5th Symposium on Advances in Approximate Bayesian Inference, 2023 1–33

Quasi-Bayesian Density Estimation via Autoregressive Predictives

Anonymous Authors
Anonymous Institution

Abstract

Bayesian methods are a popular choice for statistical inference in small-data regimes due to the regularization effect induced by the prior. In the context of density estimation, the standard nonparametric Bayesian approach is to target the posterior predictive of the Dirichlet process mixture model. In general, direct estimation of the posterior predictive is intractable and so methods typically resort to approximating the posterior distribution as an intermediate step. The recent development of quasi-Bayesian predictive copula updates, however, has made it possible to perform tractable predictive density estimation without the need for posterior approximation. Although these estimators are computationally appealing, they struggle on non-smooth data distributions. This is due to the comparatively restrictive form of the likelihood models from which the proposed copula updates were derived. To address this shortcoming, we consider a Bayesian nonparametric model with an autoregressive likelihood decomposition and a Gaussian process prior. While the predictive update of such a model is typically intractable, we derive a quasi-Bayesian update that achieves state-of-the-art results in small-data regimes.

1. Introduction

Modelling the joint distribution of multivariate random variables with density estimators is a central topic in modern unsupervised machine learning research (Durkan et al., 2019; Papamakarios et al., 2017). For density estimation, the typical Bayesian approach is to target the *Bayesian predictive density*, $p_n(x) = \int f(x|\theta)\pi_n(\theta)d\theta$, where π_n denotes the posterior density of the model parameters θ after observing $x_1, \ldots, x_n$, and f denotes the likelihood function. Methods to estimate $p_n(x)$ typically resort to first approximating the posterior distribution $\pi_n(\theta)$ as an intermediate step. Approximating the posterior, however, is often computationally intensive. Here, we focus on the computationally faster approach of one-step-ahead predictive updates $p_{i-1}(x) \to p_i(x)$ based on bivariate copulas, which were first introduced by Hahn et al. (2018) for univariate data, and extended by Fong et al. (2021) to the multivariate setting and to regression analyses. These updates are inspired by Bayesian models and retain many desirable Bayesian properties, such as coherence and regularization. However, copula updates do not correspond exactly, nor approximately, to a traditional Bayesian likelihood-prior model, and are referred to as *quasi-Bayesian* (Fortini and Petrone, 2020).

2. Background

2.1. Univariate Predictive Density Updates

To compute predictive densities quickly, Hahn et al. (2018) propose an iterative approach. For $x \in \mathbb{R}$, any sequence of Bayesian posterior predictive densities $p_i(x)$ with likelihood f and posterior π_i, conditional on $x_{1:i}$, can be expressed as

$$p_i(x) = \int f(x|\theta)\pi_i(\theta)d\theta = p_{i-1}(x)h_i(x, x_i), \tag{1}$$

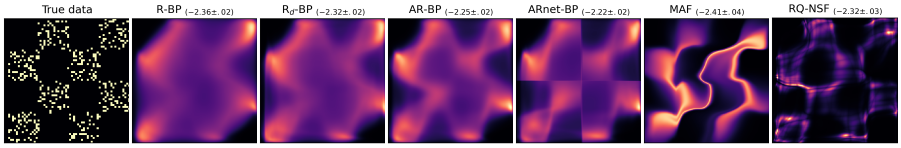


Figure 1: Density estimates of 600 observations from a chessboard distribution, reported with mean and standard deviation of test log likelihoods. For larger training sizes, see Supplement D.2. Our methods, AR-BP and ARnet-BP, outperform R-BP and AR neural networks.

for some bivariate function $h_i(x, x_i)$ (Hahn et al., 2018). Rearranging for h_i, we have

$$h_i(x, x_i) = \frac{p_i(x)}{p_{i-1}(x)} \overset{(a)}{=} \frac{p_{i-1}(x|x_i)}{p_{i-1}(x)} \overset{(b)}{=} \frac{p_{i-1}(x, x_i)}{p_{i-1}(x)p_{i-1}(x_i)} \tag{2}$$

where (a) holds by definition, and (b) $p_{i-1}(x, x_i) = p_{i-1}(x|x_i)p_{i-1}(x_i) = p_i(x)p_{i-1}(x_i)$ holds by Bayes' law. Hahn et al. (2018) show that $h_i(x, x_i)$ is the transformation of a bivariate copula density. A *bivariate copula* is a bivariate cumulative distribution function (CDF) $C : [0, 1]^2 \to [0, 1]$ with uniform marginal distributions that is used to characterise the dependence between two random variables independent of their marginals. According to Sklar's theorem (Sklar, 1959), for any bivariate density $f(y_1, y_2)$ with continuous marginal CDFs, $F_1(y_1)$ and $F_2(y_2)$, and marginal densities $f_1(y_1)$ and $f_2(y_2)$, there exists a unique bivariate copula C with density c such that $f(y_1, y_2) = c\{F_1(y_1), F_2(y_2)\} f_1(y_1)f_2(y_2)$.

Applying the copula factorization from Sklar's theorem to (2) yields that there exists some bivariate copula density c_i such that $p_{i-1}(x, x_i) = c_i\{P_{i-1}(x), P_{i-1}(x_i)\}p_{i-1}(x)p_{i-1}(x_i)$, and thus $h_i(x, x_i) = c_i\{P_{i-1}(x), P_{i-1}(x_i)\}$, where P_{i-1} is the CDF corresponding to the predictive density p_{i-1}. Given prior π and likelihood f, Equation 2 suggests that the update function can be written as

$$h_i(x, x_i) = \frac{\int f(x|\theta)f(x_i|\theta)\pi_{i-1}(\theta)d\theta}{\int f(x|\theta)\pi_{i-1}(\theta)d\theta \int f(x_i|\theta)\pi_{i-1}(\theta)d\theta}.$$

For each Bayesian model, there is thus a unique sequence of symmetric copula densities $c_i(u, v) = c_i(v, u)$. This sequence has the property that $c_n(\cdot, \cdot) \to 1$ converges to a constant function as $n \to \infty$, ensuring that the predictive density converges asymptotically with sample size n.

In general, the above equation is intractable due to the posterior so it is not possible to compute the iterative update in (1) for fully Bayesian models. Alternatively, we will consider sequences of h_i that match the Bayesian model for $i = 1$, but not for $i > 1$. As mentioned above, this copula update no longer corresponds to a Bayesian model, nor are the resulting predictive density estimates approximations to a Bayesian model. Nevertheless, if the copula updates are *conditionally identically distributed*, they still exhibit desirable Bayesian characteristics such as coherence and regularization, and are hence referred to as *quasi-Bayes*. Please refer to Berti et al. (2004) for details.

2.2. Multivariate Predictive Density Updates

The above arguments cannot directly be extended to multivariate $x \in \mathbb{R}^d$ since h_i cannot necessarily be written as $c_i\{P_{i-1}(x), P_{i-1}(x_i)\}$ for $d > 1$. However, (2) still holds, and recursive predictive updates with bivariate copulas as building blocks can be derived explicitly with Dirichlet Process

Mixture Models (DPMMs) as a general-use nonparametric model (Fong and Lehmann, 2022). The DPMM (Escobar, 1988; Escobar and West, 1995) can be written as

$$f(x|G) = \int_{\Theta} K(x|\theta)\, dG(\theta), \text{ with } G \sim \text{DP}(c, G_0) \tag{3}$$

where $\theta \in \Theta = \mathbb{R}^d$ are parameter vectors, the prior assigned to G is a Dirichlet process (DP) prior with base measure G_0 and concentration parameter $c > 0$ (Ferguson, 1973), and $K(x|\theta)$ is a user-specified kernel. In particular, Fong et al. (2021) consider the base measure $G_0 = \mathcal{N}(0, \tau^{-1} I_d)$ for some precision parameter $\tau \in \mathbb{R}_{>0}$, and the factorized kernel $K(x|\theta) = \mathcal{N}(x|\theta, I_d)$ where I_d is the d-dimensional identity matrix. The likelihood is then

$$f(x|G) = \int \prod_{j=1}^{d} \mathcal{N}\left(x^j \mid \theta^j, 1\right) dG(\theta), \tag{4}$$

where the dimensions of x are conditionally independent given θ. Following Hahn et al. (2018), we denote the dimension j of a vector y with y^j. This model inspires the following recursive predictive density update $p_i(x) = h_i(x, x_i) p_{i-1}(x)$ for which the first $d' \in \{1, \ldots, d\}$ marginals are

$$\frac{p_i(x^{1:d'})}{p_{i-1}\left(x^{1:d'}\right)} = 1 - \alpha_i + \alpha_i \prod_{j=1}^{d'} c\left(u_{i-1}^j(x^j), v_{i-1}^j; \rho_0\right), \tag{5}$$

where $u_{i-1}^j(x^j) := P_{i-1}\left(x^j \mid x^{1:j-1}\right), v_{i-1}^j := P_{i-1}\left(x_i^j \mid x_i^{1:j-1}\right)$, $c(u, v; \rho_0)$ is the bivariate Gaussian copula density with correlation $\rho_0 = 1/(1+\tau)$, p_0 can be any chosen prior density, and $\alpha_i = \left(2 - \frac{1}{i}\right) \frac{1}{i+1}$ (see Supplement A and Fong et al. (2021)). Note that the above update requires a specific ordering of the feature dimensions, and the Gaussian copula follows from the Gaussian distribution in the kernel and G_0 for the DPMM. While ρ_0 is a scalar here, Fong et al. (2021) also consider the setting with a distinct bandwidth parameter for each dimension. We refer to these recursive Bayesian predictives as R_d-BP, or simply R-BP if the dimensions share a single bandwidth.

3. AR-BP: Autoregressive Bayesian Predictives

For smooth data distributions, the recursive update defined in (5) generates density estimates that are highly competitive against other popular density estimation procedures such as kernel density estimation (KDE) and DPMM (Fong et al., 2021). Moreover, the iterative updates provide a fast estimation alternative to fitting the full DPMM through Markov chain Monte Carlo (MCMC). When considering more structured data, however, performance suffers due to the choices of the factorized kernel $K(\cdot|\theta) = \mathcal{N}(\cdot|\theta, I_d)$ and simple base measure $G_0 = \mathcal{N}(0, \tau^{-1} I_d)$ in the DPMM as these choices induce a priori independence between the data dimensions.

3.1. Bayesian Model Formulation

We therefore propose employing more general kernels and base measures in the DPMM and show that these inspire a more general tractable recursive predictive update. In particular, we allow the kernel to take on an autoregressive structure

$$K(x|\theta) = \prod_{j=1}^{d} \mathcal{N}\left(x^j \mid \theta^j\left(x^{1:j-1}\right), 1\right), \tag{6}$$

where $\theta^j : \mathbb{R}^{j-1} \to \mathbb{R}$ is now an unknown mean *function*, and not scalar, for dimension x^j, which we allow to depend on the previous $j-1$ dimensions of x. Thus, specifying our DPMM requires a base measure supported on the function space in which $(\theta^1, \ldots, \theta^d)$ is valued. We specify this base measure as a product of independent Gaussian process (GP) priors on the functional parameters

$$\theta^j \sim \text{GP}(0, \tau^{-1} k^j) \text{ for } j = 1, ..., d \tag{7}$$

where $k^j : \mathbb{R}^{j-1} \times \mathbb{R}^{j-1} \to \mathbb{R}$ and k^j can be any given covariance function that takes as input a pair of $x^{1:j-1}$ values. In practice, we use the same functional form of k for each j, so we will drop the superscript j. For convenience, we have written the scaling term τ^{-1} explicitly. We highlight that for $j = 1$, $\theta^1 \sim \mathcal{N}(0, \tau^{-1})$. Under this choice, the mean of the normal kernels in the DPMM for each dimension j is thus a flexible function of the first $j-1$ dimensions $x^{1:j-1}$, on which we elicit independent GP priors. The conjugacy of the GP with the Gaussian DPMM kernel in (6) is crucial for deriving a tractable density update. The proposed DPMM kernel in (6) is in fact more flexible than a general multivariate kernel, $K(x \mid \theta) = \mathcal{N}(x \mid \theta, \Sigma)$ because the multivariate kernel also implies an auto-regressive (AR) form like (6) but with parameters θ^j restricted to be linear in $x^{1:j-1}$.

3.2. Iterative Predictive Density Updates

Computing the Bayesian posterior predictive density induced by the DPMM with kernel given by (6) and base measure given by (7) through posterior estimation is *intractable* and requires MCMC. However, as before, we can utilize the model to derive tractable iterative copula updates. In Supplement B.1, we derive the corresponding recursive predictive density update $p_i(x) = h_i(x, x_i) p_{i-1}(x)$ for the first d' marginals and show that it takes on the form

$$\frac{p_i(x^{1:d'})}{p_{i-1}(x^{1:d'})} = 1 - \alpha_i + \alpha_i \cdot \prod_{j=1}^{d'} c\left(u_{i-1}^j(x^j), v_{i-1}^j; \rho^j(x^{1:j-1}, x_i^{1:j-1})\right), \tag{8}$$

with $u_{i-1}^j(x^j), v_{i-1}^j$ defined as in (5), $\alpha_i = \left(2 - \frac{1}{i}\right) \frac{1}{i+1}$, and the bandwidth given by

$$\rho^j(x^{1:j-1}, x_i^{1:j-1}) = \rho_0 k\left(x^{1:j-1}, x_i^{1:j-1}\right), \tag{9}$$

for $\rho_0 = 1/(1+\tau)$, and $\rho_i^1 = \rho_0$. Where appropriate, we henceforth drop the argument x for brevity. The conditional CDFs u_{i-1}^j can also be computed through an iterative closed form expression similarly to (8) (Supplement C.3). Note that the estimation is identical to the update given in (5) induced by the factorized DPMM kernel, except for the main difference that the bandwidth ρ is *no longer a constant*, but is now *data-dependent*. More precisely, the bandwidth for dimension j is a transformation of the GP covariance function k on the first $j-1$ dimensions. The additional flexibility afforded by the inclusion of k enables us to capture more complex dependency structures, as we do not enforce a-priori independence between the dimensions of the parameter θ. Similarly to the extension of R-BP to R_d-BP, we can define AR_d-BP by introducing dimension dependence in ρ_0.

3.3. Bandwidth Parameterisation

The choice of covariance function in (7) provides substantial modelling flexibility in our AR-BP framework. Moreover, the additional parameters associated with the covariance function allow us to

tune the implied covariance structure according to the observed data. This formulation enables us to draw upon the rich literature on the choice of covariance functions for Gaussian processes (Williams and Rasmussen, 2006). For simplicity we only consider the most popular such choice here, but study the more flexible rational-quadratic covariance in Supplement D.2. The radial basis function (RBF) covariance function is defined as $k_\ell(x^{1:j-1}, x'^{1:j-1}) = \exp[-\sum_{\kappa=1}^{j-1}\{(x^\kappa - x'^\kappa)/\ell^\kappa\}^2]$, where $\ell \in \mathbb{R}_{>0}^{d-1}$ is the length scale.

Neural parameterisation As we saw in the motivating example of the density estimation of a chessboard distribution in Figure 1, the RBF kernel can restrict the capacity of the predictive density update to capture intricate nonlinearities if the training data size is not sufficient. While the parameterization of the bandwidth in (9) was initially derived via the first predictive update for a DPMM, all we require is that the bandwidth function $\rho^j : \mathbb{R}^{j-1} \times \mathbb{R}^{j-1} \to \mathbb{R}$ lies in (0,1). We would also like $\rho^j(x^{1:j-1}, x'^{1:j-1})$ to take larger values when $x^{1:j-1}$ and $x'^{1:j-1}$ are 'close' in some sense. Motivated by this observation, we now consider more expressive bandwidth functions that can lead to increased predictive performance. In particular, we formulate an AR neural network $f_w : \mathbb{R}^d \to \mathbb{R}^{d \times d'}$ for $d' \in \mathbb{N}$ with the property that the j^{th} row of the output depends only on the first $j-1$ dimensions of the input. Let $Z = f_w(x)$ and denoting z^j to be the j^{th} row of the matrix Z, the covariance function is then computed as $\rho^j(x^{1:j-1}, x'^{1:j-1}) = \rho_0 \exp(-\sum_{\kappa=1}^{j-1} ||z^\kappa - z'^\kappa||_2^2)$.

Numerous AR neural network models have been extensively used for density estimation (Dinh et al., 2014; Huang et al., 2018; Kingma et al., 2016). In our experiments, we use a relatively simple model with parameter sharing inspired by NADE, an AR neural network designed for density estimation (Larochelle and Murray, 2011). More advanced properties like the permutation invariance of MADE (Papamakarios et al., 2017) create an additional overhead that cannot be used in the copula formulation as the predictive update is not permutation-invariant. We refer to Bayesian predictive densities estimated using AR neural networks as *ARnet Bayesian predictives* (ARnet-BP).

Table 1: Average NLL with standard error over five runs on data sets analysed by Fong et al. (2021).

	WINE	BREAST	PARKIN	IONO	BOSTON
KDE	$13.69_{\pm 0.00}$	$10.45_{\pm 0.24}$	$12.83_{\pm 0.27}$	$32.06_{\pm 0.00}$	$8.34_{\pm 0.00}$
DPMM (Diag)	$17.46_{\pm 0.6}$	$16.26_{\pm 0.71}$	$22.28_{\pm 0.66}$	$35.30_{\pm 1.28}$	$7.64_{\pm 0.09}$
DPMM (Full)	$32.88_{\pm 0.82}$	$26.67_{\pm 1.32}$	$39.95_{\pm 1.56}$	$86.18_{\pm 10.22}$	$9.45_{\pm 0.43}$
MAF	$39.60_{\pm 1.41}$	$10.13_{\pm 0.40}$	$11.76_{\pm 0.45}$	$140.09_{\pm 4.03}$	$56.01_{\pm 27.74}$
RQ-NSF	$38.34_{\pm 0.63}$	$26.41_{\pm 0.57}$	$31.26_{\pm 0.31}$	$54.49_{\pm 0.65}$	$-2.20_{\pm 0.11}$
R-BP	$13.57_{\pm 0.04}$	$7.45_{\pm 0.02}$	$9.15_{\pm 0.04}$	$21.15_{\pm 0.04}$	$4.56_{\pm 0.04}$
R_d-BP	$13.32_{\pm 0.01}$	$6.12_{\pm 0.05}$	$7.52_{\pm 0.05}$	$19.82_{\pm 0.08}$	$-13.50_{\pm 0.59}$
AR-BP	$13.45_{\pm 0.05}$	$6.18_{\pm 0.05}$	$8.29_{\pm 0.11}$	$17.16_{\pm 0.25}$	$-0.45_{\pm 0.77}$
AR_d-BP	$\mathbf{13.22_{\pm 0.04}}$	$\mathbf{6.11_{\pm 0.04}}$	$\mathbf{7.21_{\pm 0.12}}$	$16.48_{\pm 0.26}$	$\mathbf{-14.75_{\pm 0.89}}$
ARnet-BP	$14.41_{\pm 0.11}$	$6.87_{\pm 0.23}$	$8.29_{\pm 0.17}$	$\mathbf{15.32_{\pm 0.35}}$	$-5.71_{\pm 0.62}$

4. Experiments

We demonstrate the benefits of AR-BP, AR_d-BP and ARnet-BP for density estimation and prediction tasks in an experimental study with five baseline approaches and 13 different data sets. We compared our models against KDEs (Parzen, 1962), DPMMs (Rasmussen, 1999), masked autoregressive flows (MAFs) (Papamakarios et al., 2017) and rational-quadratic neural spline flows (RQ-NSFs) (Durkan et al., 2019). The hyperparameters of the baselines were tuned with cross-validation. Unless otherwise specified, we use respectively 10 permutations over samples and features to average the

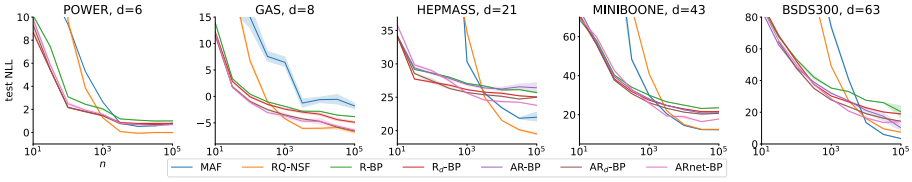


Figure 2: Average NLL and standard errors over 10 runs for training sets of different size. Our models outperform neural methods for data sets up to 10,000 samples.

quasi-Bayesian estimates. We did not see substantial improvements with more permutations. We use the same few hyperparameters (initialisation of $\rho_0, l_1, \ldots, l_d$, number of permutations, neural network architecture, and learning rate) on all data sets as our method is robust to their choice. See Supplements D and D.1 for further experiments and information.

Data sets analysed by Fong et al. (2021) See Table 1 for the negative log-likelihood (NLL) estimated on five UCI data sets (Asuncion and Newman, 2007) of small size with up to 506 samples, as investigated by Fong et al. (2021). Our proposed methods display highly competitive performance: AR$_d$-BP achieved the best test NLL on four of the data sets, while ARnet-BP prevailed on ionosphere.

Table 2: Average NLL over five runs reported with standard error for supervised tasks

	Regression			Classification		
	BOSTON	CONCR	DIAB	IONO	PARKIN	MNIST01
Linear	$0.87_{\pm 0.03}$	$0.99_{\pm 0.01}$	$1.07_{\pm 0.01}$	$0.33_{\pm 0.01}$	$0.38_{\pm 0.01}$	$0.003_{\pm 0.000}$
GP	$0.42_{\pm 0.08}$	$0.36_{\pm 0.02}$	$1.06_{\pm 0.02}$	$0.30_{\pm 0.02}$	$0.42_{\pm 0.02}$	$0.035_{\pm 0.000}$
MLP	$1.42_{\pm 1.01}$	$2.01_{\pm 0.98}$	$3.32_{\pm 4.05}$	$0.26_{\pm 0.05}$	$0.31_{\pm 0.02}$	$\mathbf{0.003_{\pm 0.000}}$
R-BP	$0.76_{\pm 0.09}$	$0.87_{\pm 0.03}$	$1.05_{\pm 0.03}$	$0.26_{\pm 0.01}$	$0.37_{\pm 0.01}$	$0.015_{\pm 0.001}$
R$_d$-BP	$0.40_{\pm 0.03}$	$0.42_{\pm 0.00}$	$1.00_{\pm 0.02}$	$0.34_{\pm 0.02}$	$0.27_{\pm 0.03}$	$0.018_{\pm 0.001}$
AR-BP	$0.52_{\pm 0.13}$	$0.42_{\pm 0.01}$	$1.06_{\pm 0.02}$	$0.21_{\pm 0.02}$	$0.29_{\pm 0.02}$	$0.015_{\pm 0.001}$
AR$_d$-BP	$\mathbf{0.37_{\pm 0.10}}$	$0.39_{\pm 0.01}$	$\mathbf{0.99_{\pm 0.02}}$	$\mathbf{0.20_{\pm 0.02}}$	$0.28_{\pm 0.03}$	$0.017_{\pm 0.001}$
ARnet-BP	$0.45_{\pm 0.11}$	$\mathbf{-0.03_{\pm 0.00}}$	$1.41_{\pm 0.07}$	$0.24_{\pm 0.04}$	$\mathbf{0.26_{\pm 0.04}}$	$0.014_{\pm 0.001}$

Data sets analysed by Papamakarios et al. (2017) A number of UCI data sets have become the standard evaluation benchmark for deep AR models (Durkan et al., 2019; Huang et al., 2018; Papamakarios et al., 2017). These include low-dimensional data sets with up to 63 features, but at least 29,000 with up to 10^6 samples. In many circumstances, data sets of such a data size are not available. Thus, we trained the models on subsets of the full data set.

Supervised Learning R-BP methods, including AR-BP, can be used for prediction tasks such as regression and classification (Fong et al., 2021). In short, this is achieved by estimating the conditional predictive density $p_n(y|x)$ of the labels y directly by assuming a dependent Dirichlet process likelihood. See Supplement C.2 for details. Again, we follow the experimental set-up of Fong et al. (2021), and additionally report results on the MNIST data set, restricted to digits of class 0 and 1. We report the conditional test NLL $-\frac{1}{n'}\sum_i \log p_n(y_i^*|x_i^*)$ for a test set $\{(x_1^*, y_1^*), \ldots, (x_{n'}^*, y_{n'}^*)\}$. We compared our models against a GP, a linear Bayesian model (Linear), and a one-hidden-layer multilayer perceptron (MLP) on several classification and regression tasks. To get a distribution over the predicted outcome in the regression case, we trained an ensemble over 10 MLPs. Our proposed methods were again highly competitive (Table 2).

References

Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

Patrizia Berti, Luca Pratelli, and Pietro Rigo. Limit theorems for a class of identically distributed random variables. *The Annals of Probability*, 32(3):2029–2052, 2004.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

A Philip Dawid. Prequential analysis. *Encyclopedia of Statistical Sciences*, 1:464–470, 1997.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Michael D Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588, 1995.

Michael David Escobar. *Estimating the means of several normal populations by nonparametric estimation of the distribution of the means*. PhD thesis, Yale University, 1988.

Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

Edwin Fong and Brieuc Lehmann. A predictive approach to bayesian nonparametric survival analysis. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 6990–7013. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/fong22a.html.

Edwin Fong, Chris Holmes, and Stephen G Walker. Martingale posterior distributions. *To appear at the Journal of the Royal Statistical Society: Series B (with discussion)*, 2021.

Sandra Fortini and Sonia Petrone. Quasi-bayes properties of a procedure for sequential learning in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82 (4):1087–1114, 2020.

David Gunawan, Khue-Dung Dang, Matias Quiroz, Robert Kohn, and Minh-Ngoc Tran. Subsampling sequential monte carlo for static bayesian models. *Statistics and Computing*, 30(6):1741–1758, 2020.

P Richard Hahn, Ryan Martin, and Stephen G Walker. On recursive Bayesian predictive distributions. *Journal of the American Statistical Association*, 113(523):1085–1093, 2018.

Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

Carl Rasmussen. The infinite gaussian mixture model. *Advances in neural information processing systems*, 12, 1999.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879, 2008.

M Sklar. Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8: 229–231, 1959.

Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. *Advances in Neural Information Processing Systems*, 30, 2017.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Number 3 in 2. MIT press Cambridge, MA, 2006.

Appendix A. Additional Information on AR-BP

A.1. Motivation for Autoregressive Factorisation

Our approach can be viewed as a Bayesian version of an online KDE procedure. To see this, note that a KDE trained on $i-1$ observations – yielding the density estimate $q_{i-1}(x)$ – can be updated after observing the i^{th} observation x_i via $q_i(x) = (1-\alpha_i)q_{i-1}(x) + \alpha_i d(x, x_i)$, where $\alpha_i = 1/i$ and $d(\cdot,\cdot)$ denotes the kernel of the KDE. Rather than adding a weighted kernel term directly, AR-BP instead adds an adaptive kernel that depends on a notion of distance between x and x_i based on the predictive CDFs conditional on $x_{1:i-1}$.

To better understand the importance of the data-dependent bandwidth, we compare the conditional predictive mean of R-BP and AR-BP in the bivariate setting $X \times Y$. Under the simplifying assumption of Gaussian predictive densities, we show in Supplement B.3 that the conditional mean of $Y \mid X$ is given by

$$\mu_i(x) = \mu_{i-1}(x) + \alpha_i(x, x_i)\rho(x, x_i)(y_i - \mu_{i-1}(x_i)),$$
$$\alpha_i(x, x_i) = \frac{\alpha_i c(P_{i-1}(x), P_{i-1}(x_i); \rho)}{1 - \alpha_i + \alpha_i c(P_{i-1}(x), P_{i-1}(x_i); \rho)}.$$

Note that $\rho(x, x_i) = \rho_0$ for R-BP. Intuitively, the updated mean is the previous mean plus a residual term at y_i scaled by some notion of distance between x and x_i. For R-BP, this distance between x and x_i depends only on their predictive CDF values through $\alpha_i(x, x_i)$. This can result in undesirable behaviour as shown in the upper plot in Figure 3(a), where the peak of $\alpha_i(x, x_i)$, as a function of x, is not centred at x_i. Counterintuitively, there is thus an $x > x_i$ where $\mu_i(x)$ is updated more than at the actual observed $x = x_i$. This follows from the lack of focus on *conditional* density estimates for R-BP, which is alleviated by AR-BP. In the AR case, $\rho(x, x_i)$ takes into account the Euclidean distance between x and x_i in the data space. We see in the lower plot in Figure 3(a) that the peak is closer to x_i. Figure 3(b) further demonstrates this difference on another toy example - we see that R-BP struggles to fit a linear conditional mean function for $n = 4$, focussing density in data sparse regions, while AR-BP succeeds to assign significant density only to points on the data manifold.

Remark 1 *The data-dependent bandwidth also appears when starting from other Bayesian non-parametric models, such as dependent DPs and GPs (see Supplement B.2.2 for the derivation).*

A.2. Training the Update Parameters

In order to compute the predictive density $p_n(x^*)$, we require the vector of conditional CDFs $[v_1^j, \ldots, v_{n-1}^j]$ where $v_i^j = P_i(x_{i+1}^j \mid x_{i+1}^{1:j-1})$. Given a bandwidth parameterization, obtaining this vector thus amounts to model-fitting, and each v_i^j requires $i-1$ iterations (Supplement C.3), for $i \in \{1, \ldots, n\}$. We note that the order of samples and dimensions influences the prediction performance in AR density estimators (Vinyals et al., 2015). In practice, averaging over different permutations of these improves performance (Supplement C.3). Full implementation details can be found in Supplement C.

A.3. Computational Complexity

The above procedure results in a computational complexity of $\mathcal{O}(Mdn^2)$ at the training stage where M is the number of permutations. At test time, we have already obtained the necessary conditional

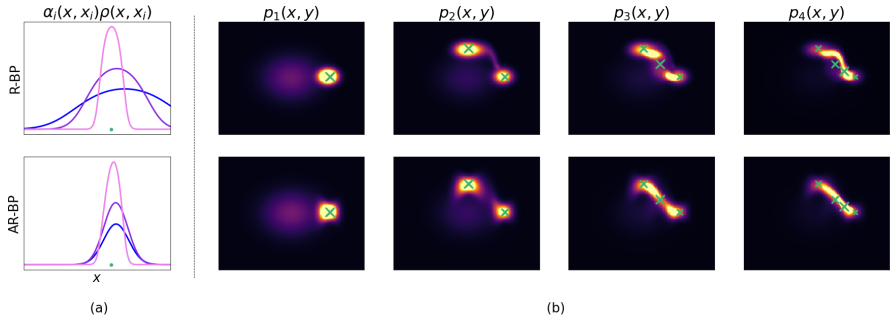


Figure 3: (a) Plots of $\alpha_i(x, x_i)\rho(x, x_i)$ for R-BP and AR-BP for $\rho_0 \in \{0.5, 0.7, 0.95\}$ (—,—,—) with new observation x_i (•). Note that $\rho(x, x_i) = \rho_0$ for R-BP, and $\ell = 1$ for AR-BP. (b) Density plots for R-BP and AR-BP trained on 4 sequential data points (×). Both figures show that the update of R-BP, unlike AR-BP, is not centred around the new datum.

(a) **Train:** Estimate $v_i^j = P_{i-1}(x_i^{1:j})$ for each i

Initialise $u_0^j(x_i) \leftarrow \Phi(x_i^j)$

For each preceding observation x_k with $k < i$:

For each feature j:

Compute *data-dependent* bandwidth $\rho^j(x_i^{1:j}, x_k^{1:j})$ (9)

Update conditional CDF $u_i^j(x_k) := P_i^j(x_k)$ based on the similarity between $u_{i-1}^j(x_k)$ and v_{i-1}^j (18)

Set $v_i^j \leftarrow u_i^j(x_i)$ for all j

(b) **Test:** Estimate predictive at test point $p_1(z)$

Initialise $u_0^j(z) \leftarrow \Phi(z^j)$

For each train observation x_i:

For each feature j:

Compute *data-dependent* bandwidth $\rho^j(x_i^{1:j}, z^{1:j})$ (9)

Update conditional CDF $u_i^j(z) := P_i^j(z)$ based on the similarity between $u_{i-1}^j(z)$ and v_{i-1}^j (18)

Update predictive density $p_{i-1}(z) \to p_i(z)$ (8)

Figure 4: Simplified summary of AR-BP. We repeat the training update for each train datum x_i to estimate $v_i^j = P_{i-1}(x_i^{1:j})$. These are needed at test time to update from $p_{i-1}(z) \to p_i(z)$. All steps are averaged over different feature and sample permutations. The main step that induces autoregression in the observations is highlighted pink. See Supplement C.3 for detailed algorithms.

prequential CDFs v_n^j in computing the prequential log-likelihood above. As a result, we have a computational complexity $\mathcal{O}(Mdn)$ for each test observation. Note that the introduction of a data-dependent bandwidth does not increase the computational complexity at train or test time relative to R-BP and only adds a negligible factor to the computational time for the calculation of the bandwidth.

Appendix B. Derivations

B.1. Derivation of AR-BP

For illustration purposes, we first start by summarising the derivation of the update without autoregression, closely following Appendix E.1.2 in Fong et al. (2021).

Tuning the bandwidth function Recall that the bandwidths $\rho_i(\cdot,\cdot)$ are parameterised by ρ_0 and the parameters of the chosen covariance functions or neural embedders. For AR-BP, these are the length scales ℓ of the RBF covariance function, while for ARnet-BP, these are the parameters w of the AR neural network. We fit these tunable parameters in a data-driven approach by maximising the prequential (Dawid, 1997) log-likelihood $\sum_{i=1}^{n} \log p_{i-1}(x_i)$ which is analogous to the Bayesian marginal likelihood – the tractable predictive density allows us to compute this exactly, and this approach is analogous to empirical Bayes. Specifically, we use gradient descent optimisation with Adam, sampling a different random permutation of the training data at each optimisation step (Supplement C.3).

B.1.1. NO AUTOREGRESSION (R-BP)

The multivariate DPMM with factorized kernel has the form

$$f_G(x) = \int \prod_{j=1}^{d} \mathcal{N}(x^j \mid \theta^j, 1)\, dG(\theta), \quad G \sim \mathrm{DP}\,(a, G_0)\,, \quad G_0(\theta) = \prod_{j=1}^{d} \mathcal{N}(\theta^j \mid 0, \tau^{-1}).$$

Given

$$p_i(x) = p_{i-1}(x) h_i(x, x_i),$$

Hahn et al. (2018) and Fong et al. (2021) derive the predictive density updates for R-BP by initally only considering the first step update h_1

$$p_1(x) = p_0(x) h_1(x, x_1)\cdot$$

From

$$h_i(x, x_i) = \frac{\int f(x|\theta) f(x_i|\theta) \pi_{i-1}(\theta) d\theta}{\int f(x|\theta)\pi_{i-1}(\theta)d\theta \int f(x_n|\theta)\pi_{i-1}(\theta)d\theta},$$

it follows that

$$h_1(x, x_1) = \frac{E\left[f_G(x)\, f_G(x_1)\right]}{p_0(x)\, p_0(x_1)} \tag{10}$$

where the expectation is over G coming from the prior. Following the stick-breaking representation of the DP, Fong et al. (2021) write G as

$$G = \sum_{k=1}^{\infty} w_k\, \delta_{\theta_k^*}$$

where $w_k = v_k \prod_{j<k}\{1-v_j\}$, $v_k \overset{iid}{\sim} \text{Beta}(1,a)$ and $\theta^*_k \overset{iid}{\sim} G_0$. Fong et al. (2021) then derive the numerator as

$$E\left[\sum_{j=1}^{\infty}\sum_{k=1}^{\infty} w_j\, w_k\, K(x \mid \theta^*_j)\, K(x_1 \mid \theta^*_k)\right]$$
$$= \left(1 - E\left[\sum_{k=1}^{\infty} w_k^2\right]\right) E\left[K(x \mid \theta^*)\right] E\left[K(x_1 \mid \theta^*)\right] + E\left[\sum_{k=1}^{\infty} w_k^2\right] E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]$$

where they have used the fact that $\sum_{k=1}^{\infty} w_k = 1$ almost surely. As $p_0(x) = E\left[K(x \mid \theta^*)\right]$, it follows that (10) can be expressed as

$$1 - \alpha_1 + \alpha_1 \frac{E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]}{p_0(x)\, p_0(x_1)}.$$

for some fixed α_1. For R-BP, the kernel K factorises with independent priors on each dimension, and $p_0(x) = \prod_{j=1}^{d} p_0(x^j) = \prod_{j=1}^{d} \mathcal{N}(x^j \mid 0, 1+\tau^{-1})$, so

$$\frac{E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]}{p_0(x)\, p_0(x_1)} = \prod_{j=1}^{d} \frac{E\left[K(x^j \mid \theta^{*j})\, K(x_1^j \mid \theta^{*j})\right]}{p_0(x^j)\, p_0(x_1^j)}. \tag{11}$$

Fong et al. (2021) then show that each univariate term corresponds to the bivariate Gaussian copula density,

$$c(u, v; \rho) = \frac{\mathcal{N}_2\left\{\Phi^{-1}(u), \Phi^{-1}(v) \mid 0, 1, \rho\right\}}{\mathcal{N}\left\{\Phi^{-1}(u) \mid 0, 1\right\} \mathcal{N}\left\{\Phi^{-1}(v) \mid 0, 1\right\}},$$

where Φ is the normal CDF, and $\mathcal{N}_2$ is the standard bivariate density with correlation parameter $\rho = 1/(1+\tau)$. They then suggest an alternative sequence h_i which iteratively repeats h_1, with the key feature that $\alpha_i = (2 - \frac{1}{i})\frac{1}{i+1}$. See Appendix E.1.1. in Fong et al. (2021) for a derivation of this sequence α_i.

B.1.2. WITH AUTOREGRESSION (AR-BP)

For the derivation of the AR-BP update, we can follow the arguments in the previous section until (11) where the factorised kernel assumption applies for the first time. For AR-BP, we instead have

$$\frac{E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]}{p_0(x)\, p_0(x_1)} = \prod_{j=1}^{d} \frac{E\left[K\{x^j \mid \theta^{*j}(x^{1:j-1})\}\, K\{x_1^j \mid \theta^{*j}(x^{1:j-1})\}\right]}{p_0(x^j)\, p_0(x_1^j)}. \tag{12}$$

The factorisation of the denominator follows from

$$p_0(x) = E\left[\prod_{j=1}^{d} K\{x^j \mid \theta^{*j}(x^{1:j-1})\}\right] = \prod_{j=1}^{d} E\left[K\{x^j \mid \theta^{*j}(x^{1:j-1})\}\right]$$

as we have independent GP priors on each function θ^{*j}. For notational convenience we write $\{y, x\}$ in place of $\{x^j, x^{1:j-1}\}$ in the following. With the autoregressive kernel assumption, there is the additional complexity

$$E\left[\mathcal{N}\{y \mid \theta(x), 1\}\, \mathcal{N}\{y_1 \mid \theta(x_1), 1\}\right]$$

where $\theta(\cdot) \sim \text{GP}\{0, \tau^{-1}k\}$. The marginal distribution of the GP is normal, so we have

$$[\theta(x), \theta(x_1)]^{\mathrm{T}} \sim \mathcal{N}_2(x, x_1 \mid 0, \Sigma_{x,x_1})$$

where

$$\Sigma_{x,x_1} = \begin{bmatrix} \tau^{-1} & \tau^{-1}k(x, x_1) \\ \tau^{-1}k(x, x_1) & \tau^{-1} \end{bmatrix}.$$

Again from the conjugacy of the normal, we can show that

$$E\left[\mathcal{N}\{y \mid \theta(x), 1\}\mathcal{N}\{y_1 \mid \theta(x_1), 1\}\right] = \mathcal{N}(y, y_1 \mid 0, K_{x,x_1})$$

where

$$K_{x,x_1} = \begin{bmatrix} 1 + \tau^{-1} & \tau^{-1}k(x, x_1) \\ \tau^{-1}k(x, x_1) & 1 + \tau^{-1} \end{bmatrix}.$$

Here $p_0(y) = E[\mathcal{N}(y|\theta(x))]$ is the same as above, since marginally $\theta(x) \sim \mathcal{N}(0, \tau^{-1})$. Plugging in $y = P_0^{-1}\{\Phi(z)\}$ again gives us the Gaussian copula density with correlation parameter

$$\rho_1(x) = \rho_0 k(x, x_1)$$

for $\rho_0 = 1/(1 + \tau)$.

B.2. Derivation of Gaussian Process Posterior

In this section, we derive the copula sequence for the Gaussian Process, which is fully tractable. This section is mostly for insight, but it would however be interesting to investigate any potential avenues for methodological development.

B.2.1. FIRST UPDATE STEP

We consider a univariate regression setting with $\{y, x\}$. For the GP, we have the model

$$f_\theta(y \mid x) = \mathcal{N}(y \mid \theta(x), \sigma^2), \quad \theta(\cdot) \sim \text{GP}(0, \tau^{-1}k).$$

Like in the above, we can derive the function $h_1(x, x_1)$. Following a similar argument to the AR-BP derivation, the first step GP copula density is

$$\frac{\mathcal{N}_2\left(y, y_1 \mid 0, K_2 + \sigma^2 I\right)}{p_0(y \mid x)p_0(y_1 \mid x_1)}$$

where K_i is the $i \times i$ Gram matrix, with kernel

$$k(x, x') = \tau^{-1} \exp\left\{-0.5(x - x')^2/\ell\right\}.$$

Writing in terms of P_0, we have

$$c\left\{P_0(y \mid x), P_0(y_1 \mid x_1); \rho_1(x)\right\}$$

where c is again the Gaussian copula density, but we have the correlation parameter as

$$\rho_1(x) = \frac{\exp\{-0.5(x - x_1)^2/\ell\}}{1 + \tau\sigma^2}.$$

From this, we can derive the first step of the update scheme:

$$p_1(y \mid x) = c\{P_0(y \mid x), P_0(y_1 \mid x_1); \rho_1(x)\}\, p_0(y \mid x)$$

where $c(u, v; \rho)$ is again the Gaussian copula density, and $p_0(y \mid x) = \mathcal{N}(y; 0, \sigma^2 + \tau^{-1})$.

B.2.2. ALL UPDATE STEPS

We can even derive the copula update scheme for $i > 1$, as the Gaussian process posterior is tractable. After observing $i - 1$ observations, we have

$$\pi(\theta_x, \theta_{x_i} \mid y_{1:i-1}, x_{1:i-1}) = \mathcal{N}(\mu_{i-1}, \Sigma_{i-1})$$

where each element of Σ_{i-1} has the entry

$$k_{i-1}(x, x') = k(x, x') - k(x, x_{1:i-1}) \left[K_{i-1} + \sigma^2 I\right]^{-1} k(x_{1:i-1}, x')$$

where the subscript $i - 1$ indicates it is the posterior kernel and μ_{i-1} is the posterior mean vector of the GP at x and x_i. Marginally, the GP copula after $i - 1$ data points is

$$\frac{\mathcal{N}_2\left(y, y_i; \mu_{i-1}, \Sigma_{i-1} + \sigma^2 I\right)}{\mathcal{N}\left\{y; \mu_{i-1}^{y}, k_{i-1}(x, x) + \sigma^2\right\} \mathcal{N}\left\{y_{i+1}; \mu_{i-1}^{y_{i-1}}, k_{i-1}(x_i, x_i) + \sigma^2\right\}}$$

where μ_{i-1}^{y} is the posterior mean of the GP at x and likewise for $\mu_{i-1}^{y_{i-1}}$. This is equivalent to the bivariate Gaussian copula density $c(u, v; \rho_i(x))$, where as before $u = P_{i-1}(y \mid x)$ and $v = P_{i-1}(y_{i+1} \mid x_{i+1})$. The correlation parameter is now

$$\rho_i(x) = \frac{k_{i-1}(x, x_i)}{\sqrt{\{k_{i-1}(x, x) + \sigma^2\}\{k_{i-1}(x_i, x_i) + \sigma^2\}}}$$

In summary, we have the update

$$p_i(y \mid x) = c\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho_i(x)\}\, p_{i-1}(y \mid x).$$

This gives the same predictives as fitting a full GP. While this update form does not offer any computational gains, it gives us insight into the GP update. The copula update corresponds to the regular normal update (Hahn et al., 2018) with a data-dependent bandwidth $\rho_i(x)$ which measures the distance between x and x_i based on the posterior kernel. A potential interesting direction of research is to seek approximations of the expensive $\rho_i(x)$ to aid with the computation of the GP.

B.3. Intuition for AR Copula

As in the main paper, we consider bivariate data, (x, y). As shown in Fong et al. (2021), the update for the conditional density for R-BP takes the form

$$p_i(y \mid x) = [1 - \alpha_i(x, x_i) + \alpha_i(x, x_i)\, c\,\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho\}]\, p_{i-1}(y \mid x), \tag{13}$$

where

$$\alpha_i(x, x_i) = \frac{\alpha_i c\{P_{i-1}(x), P_{i-1}(x_i); \rho\}}{1 - \alpha_i + \alpha_i c\{P_{i-1}(x), P_{i-1}(x_i); \rho\}}.$$

To show the effect of the AR update, we make simplifying assumptions to derive the update for the conditional mean function, $\mu_i(x) = \int y\, p_i(y \mid x) dy$. Let us assume that our predictive densities are normally distributed, that is $P_{i-1}(y \mid x) = \mathcal{N}(y \mid \mu_{i-1}(x), \sigma_y^2)$. This is an accurate approximation if the truth is normal and we have observed sufficient observations. Without loss of generalizability, we assume that $\sigma_y^2 = 1$. This then gives the form $P_{i-1}(y \mid x) = \Phi(y - \mu_{i-1}(x))$, which will help us in the calculation of the bivariate Gaussian copula. If we multiply by y and integrate on both sides of (13), we get

$$\mu_i(x) = [1 - \alpha_i(x, x_i)]\mu_{i-1}(x) + \alpha_i(x, x_i) \int c\,(P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho)\; y\, p_{i-1}(y \mid x)\, dy.$$

Plugging in $P_{i-1}(y \mid x) = \Phi\{y - \mu_{i-1}(x)\}$ (and similarly for the density) to the above gives

$$\int c\,(P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho)\; y\, dy = \int \frac{\mathcal{N}(y, y_i \mid [\mu_{i-1}(x), \mu_{i-1}(x_i)], 1,\ \rho)}{\mathcal{N}(y_i \mid \mu_{i-1}(x_i), 1)}\; y\, dy.$$

The above is simply the expectation of a conditional normal distribution, giving us

$$\int c\,(P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho)\; y\, dy = \mu_{i-1}(x) + \rho(y_i - \mu_{i-1}(x_i)).$$

Putting it all together, we thus have

$$\mu_i(x) = \mu_{i-1}(x) + \alpha_i(x, x_i)\rho(y_i - \mu_{i-1}(x_i)).$$

In the autoregressive case, we have

$$\mu_i(x) = \mu_{i-1}(x) + \alpha_i(x, x_i)\rho(x, x_i)(y_i - \mu_{i-1}(x_i)),$$

where we use the notations $\rho_i(x) = \rho(x, x_i)$ interchangeably to highlight the dependence of ρ on the distance between x and x_i. Further assuming $P_{i-1}(x) = \mathcal{N}(x \mid 0, 1)$ returns a tractable form for $\alpha_i(x, x')$, giving us Figure 3 in the main paper.

B.4. Derivation of Copula Update for Supervised Learning

We now derive the predictive density update for supervised learning tasks, closely following the derivations of Fong et al. (2021) for the conditional methods in Supplements E.2 and E.3. We assume fixed design points $x_{1:n} \in \mathbb{R}^{n \times d}$ and random response $y_{1:n} \in \mathbb{R}^n$.

B.4.1. CONDITIONAL REGRESSION WITH DEPENDENT STICK-BREAKING

We follow Appendix E.2.2 in Fong et al. (2021), and derive the regression copula update inspired by the dependent DP. Consider the general covariate-dependent stick-breaking mixture model

$$f_{G_x}(y) = \int \mathcal{N}(y \mid \theta, 1)\, dG_x(\theta), \quad G_x = \sum_{l=1}^{\infty} w_l(x)\, \delta_{\theta_l^*(x)}. \tag{14}$$

For the weights, we elicit the stick-breaking prior $w_l(x) = v_l(x) \prod_{j<l}\{1 - v_j(x)\}$ where $v_l(x)$ is a stochastic process on $\mathcal{X}$ taking values in $[0, 1]$, and is independent across l. For the atoms, which are now dependent on x, we assume they are independently drawn from a Gaussian process,

$$\theta_l^*(\cdot) \overset{iid}{\sim} \text{GP}(0, \tau^{-1}k),$$

where k is the covariance function. Once again, we want to compute

$$\frac{E\left[f_{G_x}(y)\, f_{G_{x_1}}(y_1)\right]}{p_0(y \mid x)\, p_0(y_1 \mid x_1)}.$$

Following the stick-breaking argument as in Section B.1.1, we can write the numerator as

$$\{1 - \beta_1(x, x_1)\}\, E\left[K\{y \mid \theta^*(x)\}\right] E\left[K\{y_1 \mid \theta^*(x_1)\}\right] + \beta_1(x, x_1) E\left[K\{y \mid \theta^*(x)\}\, K\{y_1 \mid \theta^*(x_1)\}\right]$$

where

$$K\{y \mid \theta^*(x)\} = \mathcal{N}\{y \mid \theta^*(x), 1\}, \quad \theta^*(\cdot) \sim \text{GP}(0, \tau^{-1}k),$$

and

$$\beta_1(x, x_1) = \sum_{k=1}^{\infty} E\left[w_k(x) w_k(x_1)\right].$$

As before, we have

$$\frac{E\left[f_{G_x}(y)\, f_{G_{x_1}}(y_1)\right]}{p_0(y \mid x)\, p_0(y_1 \mid x_1)} = c\left\{P_0(y \mid x), P_0(y_1 \mid x_1); \rho_1(x)\right\}$$

where $\rho_1(x) = \rho_0 k(x, x_1)$ and $\rho_0 = 1/(1 + \tau)$. We thus have the copula density as a mixture of the independent and Gaussian copula density. This then implies the copula update step of the form

$$p_i(y \mid x) = \left[1 - \beta_i(x, x_i) + \beta_i(x, x_i)\, c\left\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho_i(x)\right\}\right] p_{i-1}(y \mid x),$$

where we write $\rho_i(x) = \rho_i^{d+1}(x)$. As in Fong et al. (2021), we turn to the multivariate update for inspiration where we do not update $P_n(x)$ and instead keep it fixed at $P_0(x) = \Phi(x)$ (for each dimension). This gives us

$$\beta_i(x, x_i) = \frac{\alpha_i \prod_{j=1}^{d} c\left\{\Phi\left(x^j\right), \Phi\left(x_i^j\right); \rho_i^j(x^{1:j-1})\right\}}{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left\{\Phi\left(x^j\right), \Phi\left(x_i^j\right); \rho_i^j(x^{1:j-1})\right\}}. \tag{15}$$

B.4.2. CLASSIFICATION WITH BETA-BERNOULLI COPULA UPDATE

In the classification setting (Appendix E.3.1 in Fong et al. (2021)), Fong et al. (2021) assume a beta-Bernoulli mixture for $y_i \in \{0, 1\}$. As the derivation is written w.r.t ρ, we simply replace ρ with our definition of $\rho_i^j(x^{1:j-1})$, giving the update

$$p_i(y \mid x) = (1 - \beta_i(x, x_i) + \beta_i(x, x_i)\, b\,\{q_{i-1}, r_{i-1}; \rho_i(x)\})\, p_{i-1}(y \mid x)$$

where $q_{i-1} = p_{i-1}(y \mid x), r_i = p_{i-1}(y_i \mid x_i)$, $\rho_i(x)$ as in Equation 9, $\beta_i(x, x_i)$ similarly as in (15), and finally the copula-like function b given by

$$b\{q_{i-1}, r_{i-1}; \rho_i(x)\} = \begin{cases} 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} \wedge r_{i-1}}{q_{i-1}\, r_{i-1}} & \text{if } y = y_i \\ 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} - \{q_{i-1} \wedge (1 - r_{i-1})\}}{q_{i-1}\, r_{i-1}} & \text{if } y \neq y_i. \end{cases}$$

Appendix C. Methodology

In this section, we provide more details on the methodology referred to in the main part of the paper.

C.1. Generative Modelling

First, we consider three approaches to generative modelling

1. Inverse sampling
2. Importance sampling
3. Sequential Monte Carlo (SMC)

C.1.1. INVERSE SAMPLING

Univariate setting As noted by Fong et al. (2021), we can sample from $x^* \sim P_n(x)$ by inverse sampling, that is

$$u \sim \mathcal{U}[0, 1], \quad x^* \sim P_n^{-1}(u).$$

As we cannot evaluate $P_n^{-1}(u)$ directly, we instead solve an optimisation problem

$$x^* = \underset{x}{\operatorname{argmin}} |P_n(x) - u|$$

Multivariate setting The univariate procedure can be repeated iteratively in the multivariate setting given the conditional distribution

$$\begin{aligned} u^1 &\sim \mathcal{U}[0, 1],\ x^1 = P_n^{-1}(u^1) \\ u^2 &\sim \mathcal{U}[0, 1],\ x^2 = P_n^{-1}(u^2 \mid x^1) \\ &\dots \\ u^d &\sim \mathcal{U}[0, 1],\ x^d = P_n^{-1}(u^d \mid x^{1:d-1}) \end{aligned}$$

C.1.2. IMPORTANCE SAMPLING

In practice, inverse sampling is unstable and is highly dependent on the performance of the optimization. An alternative approach to data generation is importance sampling. This includes two steps

1. Sampling a set of particles $z_1, \ldots, z_B$ from the initial predictive p_0.
2. Re-sampling $z_1, \ldots, z_B$ with replacement based on the weights $w_1 = p_n(z_1)/p_0(z_1), \ldots, w_B = p_n(z_B)/p_0(z_B)$.

C.1.3. SEQUENTIAL MONTE CARLO

Importance sampling will perform poorly if p_n and p_0 are far apart. Instead, we propose a SMC procedure. A similar SMC sampling scheme has been proposed for univariate imputation of censored survival data by Fong and Lehmann (2022). Here, the goal is parameter inference, and thus only requires *implicit* sample observations by drawing the marginal CDF u_n^j from a uniform distribution. In our case, we generate new *explicit* data directly by sampling from the data space. Please see Algorithm 6 for a complete overview. As this sampling approach is similar to evaluating the density at test data points (Algorithm 5), we highlighted the differences in blue. In short,

1. We sample a set of particles $z_1, \ldots, z_B$ from the initial predictive p_0, and set the particle weights to $w_k^{[0]} = 1$ for all $k = 1, \ldots, B$
2. We update the predictive $p_{i-1} \to p_i$, and the particle weights $w_k^{[i]} = w_k^{[i-1]} \cdot p_i\left(z_k^{[i-1]}\right) / p_{i-1}\left(z_k^{[i-1]}\right)$ for each training observation
3. If the effective sample size (ESS) is smaller than half of the number of particles, we resample $z_1, \ldots, z_B$ and $w_1^{[i]}, \ldots, w_1^{[B]}$ based on their weights.

Note that particle diversity can be improved by introducing move steps, for example using Markov kernels Chopin (2002); Gunawan et al. (2020).

In Figure 5, we see that inverse sampling struggles on a simple GMM example. On the other hand, importance sampling and SMC provide reasonable samples. Similar sampling schemes have been proposed for Restricted Boltzmann Machines (Larochelle and Murray, 2011; Salakhutdinov and Murray, 2008) where samples can only be drawn from the model approximately by Gibbs sampling.

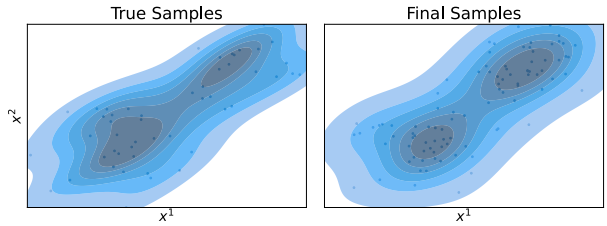
C.2. Supervised Learning

We briefly recap how joint density estimation can be extended to conditional supervised learning (regression and classification), as outlined by Fong et al. (2021). Please see Supplement B.4 for the derivation. Given fixed design points $x_{1:n}$ and random response $y_{1:n}$, the problem at hand is to infer a family of conditional densities $\{f_x(y) : x \in \mathbb{R}^d\}$.

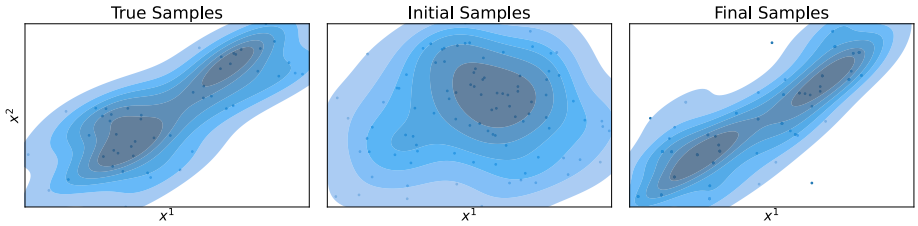
C.2.1. REGRESSION

For the regression case, Fong et al. (2021) posit a Bayesian model with the nonparametric likelihood being a covariate-dependent stick-breaking DPMM:

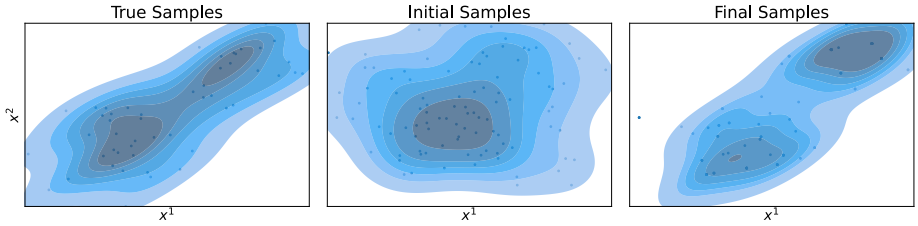
$$f_{G_x}(y) = \int \mathcal{N}(y \mid \theta, 1)\, dG_x(\theta), \quad G_x = \sum_{k=1}^{\infty} w_k(x)\, \delta_{\theta_k^*}, \tag{16}$$



(a) Inverse Sampling



(b) Importance Sampling



(c) Sequential Monte Carlo

Figure 5: 100 samples generated from AR_d-BP trained on 50 samples from a GMM with 4 components. All three sampling approaches manage to preserve the multi-modal data distribution.

where $w_k(\mathbf{x})$ follows an x-dependent stick-breaking process. Our contribution is to assume an autoregressive factorisation of the kernel and independent GP priors on θ_k^*. See Supplement B.4.1 for the derivation of the predictive density update that is now given by

$$p_i(y \mid x) = [1 - \beta_i(x, x_i) + \beta_i(x, x_i)\, c\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho_i(x)\}]\; p_{i-1}(y \mid x), \quad (17)$$

where $\rho_i(x) = \rho_i^{d+1}(x)$ and β as in (15).

C.2.2. CLASSIFICATION

For $y_i \in \{0, 1\}$, Fong et al. (2021) assume a beta-Bernoulli mixture. As explained in Supplement B.4.2 and Fong et al. (2021), this gives the same update as in the regression setting with the difference that the copula c in (17) is replaced with

$$b\{q_{i-1}, r_{i-1}; \rho_i(x)\} = \begin{cases} 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} \wedge r_{i-1}}{q_{i-1}\, r_{i-1}} & \text{if } y = y_i \\ 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} - \{q_{i-1} \wedge (1 - r_{i-1})\}}{q_{i-1}\, r_{i-1}} & \text{if } y \neq y_i, \end{cases}$$

where $\rho_i(x) = \rho_i^{d+1}(x)$, $q_{i-1} = p_{i-1}(y \mid \mathbf{x})$, $r_{i-1} = p_{i-1}(y_i \mid \mathbf{x}_i)$ and $\rho_y \in (0, 1)$.

C.3. Implementation Details

Please see Algorithm 1 for the full estimation procedure, Algorithm 2 for the optimisation of the bandwidth parameters, Algorithm 4 for the fitting procedure of the predictive density updates, and eventually Algorithm 5 for the steps during test-time inference. All algorithms are written for one specific permutation of the dimensions, and are repeated for different permutations.

Note that at both training time and test time, we need to consider the updates on the scale of the CDFs, that is for the terms such as $u_i^j(x^j)$, which appear in the update step (8). Given

$$u_i^j(x^j) = P_i(x^j | x^{1:j-1}) = \int_{-\infty}^{x^j} p_i(x^{1:j-1}, x'^j) / p_i(x^{1:j-1}) dx'^j,$$

and (8), the CDFs $u_i^j(x^j)$ take on the tractable update

$$u_i^j = \left\{ (1 - \alpha_i) u_{i-1}^j + \alpha_i H\left(u_{i-1}^j, v_{i-1}^j; \rho_i^j\right) \prod_{r=1}^{k-1} c\left(u_{i-1}^r, v_{i-1}^r; \rho_i^r\right) \right\} \frac{p_{i-1}\left(x^{1:k-1}\right)}{p_i\left(x^{1:k-1}\right)}, \quad (18)$$

and set $v_{i-1}^j = u_{i-1}^j(x_i)$ which holds by definition, where we dropped the argument x for simplicity from ρ_i^j and u_i^j, and $H(u, v; \rho)$ denotes the conditional Gaussian copula distribution with correlation ρ, that is

$$H(u, v; \rho) = \int_0^u c(u', v; \rho) du' = \Phi\left\{ \frac{\Phi^{-1}(u) - \rho \Phi^{-1}(v)}{\sqrt{1 - \rho^2}} \right\}.$$

The Gaussian copula density $c(u, v; \rho)$ is given by

$$c(u, v; \rho) = \frac{\mathcal{N}_2\left\{\Phi^{-1}(u), \Phi^{-1}(v) \mid 0, 1, \rho\right\}}{\mathcal{N}\{\Phi^{-1}(u) \mid 0, 1\} \mathcal{N}\{\Phi^{-1}(v) \mid 0, 1\}},$$

where Φ is the normal CDF, and $\mathcal{N}_2$ is the standard bivariate density with correlation $\rho \in (0, 1)$.

Ordering Note that the predictive density update depends on the ordering of both the training data and the dimensions. This permutation dependence is not an additional assumption on the data generative process, and the only implication is that the subset of ordered marginal distributions continue to satisfy (5) (main paper). In the absence of a natural ordering of the training samples or the dimensions, we take multiple random permutations, observing in practice that the resulting averaged density estimate performs better. More precisely, for a given permutation of the dimensions, we first tune the bandwidth parameters, and then calculate density estimates based on multiple random permutations of the training data. We then average over each of the resulting estimates to obtain a single density estimate for each dimension permutation, and subsequently take the average across these estimates to obtain the final density estimate. Importantly, our method is parallelizable over permutations and thus able to exploit modern multi-core computing architectures.

Algorithm 1 Full density estimation pipeline

Input:

$x_{1:n}$: training observations;
$x_{n+1:n+n'}$: test observations;
M: number of permutations over samples and features to average over;
n_ρ: number of train observations used for the optimisation of bandwidth parameters; **Output:**
$p_1(x_{1+1}), \dots p_1(x_{1+n'})$: density of test points

procedure FULL˙DENSITY˙ESTIMATION
    Compute optimal bandwidth parameters ▷ $\mathcal{O}(Mn_\rho^2 d \cdot \#\text{gradient steps})$
    Compute $v_i^{j,(m)}$ for $i \in \{1, \dots, 1\}, j \in \{1, \dots, d\}, m \in \{1, \dots, M\}$ ▷ $\mathcal{O}(M1^2 d)$
    Evaluate density at test observations $x_{n+1:n+n'}$ ▷ $\mathcal{O}(M1n'd)$
end procedure

Algorithm 2 Estimate optimal bandwidth parameters

Input:

$x_{1:n}$: training observations;

M: number of permutations over samples and features to average over;

n_ρ: number of train observations used for the optimisation of bandwidth parameters;

`maxiter`: number of iterations;

$\mathcal{R}^{(0)}$: initialisation of bandwidth parameters:

-$\mathcal{R}^{(0)} = \{\rho_0^{(0)}, l_1^{(0)}, \ldots, l_{d-1}^{(0)}\}$ (by default, $\rho_0^{(0)} \leftarrow 0.9, l_1^{(0)} \leftarrow 1, \ldots, l_{d-1}^{(0)} \leftarrow 1$) for AR-BP,

-$\mathcal{R}^{(0)} = \{\rho_0^{(0)}, w\}$ (by default, $\rho_0^{(0)} \leftarrow 0.9$, and w initialised as implemented in `Haiku` by default) for ARnet-BP

Output:

$\mathcal{R}^{(\texttt{maxiter})}$: optimal bandwidth parameters

procedure OPTIMAL˙BANDWIDTH˙AND˙LENGTHSCALES
  Subsample $\{x'_1, \ldots, x'_{n_\rho}\}$ from $x_{1:n}$
  for $s \leftarrow 1$ **to** `maxiter` **do**
    $\dot{}, \{p_{i-1}^{(m)}(x'_i)\}_{i,m} \leftarrow$ FIT˙CONDITIONAL˙PREDICTIVE˙CDF($\mathcal{R}^{(s-1)}, \{x'_1, \ldots, x'_{n_\rho}\}, M$, fit˙density=True)
    Compute $L(x'_1, \ldots, x'_{n_\rho}) = -\sum_{m=1}^{M} \sum_{i=1}^{n_\rho} \log p_{i-1}^{(m)}(x'_i)$
    $\mathcal{R}^{(s)} \leftarrow$ ADAM˙STEP($\mathcal{R}^{(s-1)}, L$)
  end for
  return $\mathcal{R}^{(s)}$
end procedure

Algorithm 3 Single copula update

Input:

z: observation to update the log density;
x_i: observation to update with;
i: sample index;
j: feature index;
$u_{i-1}^{j}(z)$: predictive CDF for z;
$v_{i-1}^{j,(m)}$: prequential CDF;
$\rho_i^j(z^{1:j-1})$=None: bandwidth;
$\mathcal{R}$=None: bandwidth parameters;

Output:

$u_i(z)$

procedure CDF˙UPDATE

Compute

$$\rho_i^j(z^{1:j-1}) \leftarrow \rho_0 k_{\mathcal{R}}\left(z^{1:j-1}, x_i^{1:j-1}\right)$$

where $k_{\mathcal{R}}$ denotes the user-defined kernel if ρ =None

Compute the bivariate Gaussian copula density

$$c\{u_{i-1}^{j}(z), v_{i-1}^{j,(m)}; \rho_j\} \leftarrow \frac{\mathcal{N}_2\left\{\Phi^{-1}(u_{i-1}^{j}(z)), \Phi^{-1}(v_{i-1}^{j,(m)}) \mid 0, 1, \rho_i^j(z^{1:j-1})\right\}}{\mathcal{N}\{\Phi^{-1}(u_{i-1}^{j}(z)) \mid 0, 1\}\mathcal{N}\{\Phi^{-1}(v_{i-1}^{j,(m)}) \mid 0, 1\}}$$

Compute the conditional Gaussian copula CDF

$$H\left\{u_{i-1}^{j}(z), v_{i-1}^{j,(m)}; \rho_i^j(z^{1:j-1})\right\} \leftarrow \Phi\left\{\frac{\Phi^{-1}(u_{i-1}^{j}(z)) - \rho_i^j(z^{1:j-1})\Phi^{-1}(v_{i-1}^{j,(m)})}{\sqrt{1-\rho_i^j(z^{1:j-1})^2}}\right\}$$

Compute $\alpha_i = (2 - \frac{1}{i})\frac{1}{i+1}$

Compute $u_i^j(z) = P_i^j(z|z^{1:j-1})$ by

$$u_i^j(z) \leftarrow \left\{(1-\alpha_i)u_{i-1}^{j}(z) + \alpha_i H\left(u_{i-1}^{j}(z), v_{i-1}^{j,(m)}; \rho_i^j(z)\right)\prod_{l=1}^{j-1} c\left(u_{i-1}^{l}(z), v_{i-1}^{l,(m)}; \rho_i^j(z)\right)\right\}$$
$$\cdot 1 \Big/ \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u_{i-1}^{j}(z), v_{i-1}^{j,(m)}; \rho_i^j(z)\right)\right\}$$

return $u_i(z)$

end procedure

Algorithm 4 Estimate prequential CDFs at train observations

Input:

$\mathcal{R}$: bandwidth parameters
$x_{1:n}$: training observations;
M: number of permutations over features to average over;
compute˙density (by default, False);

Output:

$\{v_{i-1}^{j,(m)}\}_{i,j,m}, \{p_{i-1}^{(m)}(x_i)\}_{i,m}$ if compute˙density, else $\{v_{i-1}^{j,(m)}\}_{i,j,m}$

procedure FIT˙CONDITIONAL˙PREDICTIVE˙CDF
  for $m \leftarrow 1$ **to** M **do**
    Sample permutation $\pi_1 \in \Pi(1), \pi_2 \in \Pi(d)$
    Change the ordering of the training observations $\{x_1^{(m)}, \ldots, x_1^{(m)}\} \leftarrow \{\pi_1(x_1), \ldots, \pi_1(x_1)\}$ and the features $x \leftarrow [\pi_2(x^1), \ldots, \pi_2(x^d)]$ ▷ For simplicity we will drop the superscript in the following
    for $j \leftarrow 1$ **to** d **do**
      for $k \leftarrow 1$ **to** 1 **do**
        Initialise $u_0^j(x_k) \leftarrow \Phi(x_k^j)$ ▷ u also depends on the permutation m, but since we do not reuse u after m is updated, we drop the index for simplicity
      end for
    end for
    for $i \leftarrow 1$ **to** 1 **do**
      Set $v_{i-1}^{j,(m)} \leftarrow u_{i-1}^j(x_i)$ **for** $j \leftarrow 1$ **to** d
      for $k \leftarrow 1$ **to** i **do**
        for $j \leftarrow 1$ **to** d **do**
          $u_i^j(x_k)$=CDF˙UPDATE(x_k, x_i, i, j, $u_{i-1}^j(x_k)$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)
        end for
        if compute˙density **then**

$$p_i^{(m)}(x_k) \leftarrow \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u_{i-1}^j(x_k), v_{i-1}^{j,(m)}; \rho_i^j(x_k)\right)\right\} p_{i-1}^{(m)}(x_k)$$

        end if
      end for
    end for
  end for
  return $\{v_{i-1}^{j,(m)}\}_{i,j,m}, \{p_{i-1}^{(m)}(x_i)\}_{i,m}$ **if** compute˙density **else** $\{v_{i-1}^{j,(m)}\}_{i,j,m}$
end procedure

Algorithm 5 Evaluate density at test observations

Input:

$\mathcal{R}$: bandwidth parameters

$x_{n+1:n+n'}$: test observations;

$\{\{x_1^{(1)}, ..., x_1^{(1)}\}, \ldots, \{x_1^{(M)}, ..., x_1^{(M)}\}\}$: sets of permuted train observations;

$\{v_i^{j,(m)}\}_{i,j,m}$: prequential conditional CDFs at train observations;

M: number of observations over features to average over;

Output:

$\{p_1(x_{1+1}), \ldots, p_1(x_{1+n'})\}$

procedure EVAL˙DENSITY

for $m \leftarrow 1$ **to** M **do**

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

for $k \leftarrow 1$ **to** n' **do**

Initialise $u_0^j(x_{1+k}) \leftarrow \Phi(x_{1+k}^j)$

end for

end for

for $i \leftarrow 1$ **to** 1 **do**

for $k \leftarrow 1$ **to** n' **do**

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

$u_i^j(x_k)$=CDF˙UPDATE(x_{1+k}, x_i, i, j, $u_{i-1}^j(x_{1+k})$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)

end for

Compute density

$$p_i^{(m)}(x_{1+k}) \leftarrow \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u_{i-1}^j(x_{1+k}), v_{i-1}^{j,(m)}; \rho_i^j(x_{1+k})\right)\right\} p_{i-1}^{(m)}(x_{1+k})$$

end for

end for

end for

▷ Average density over permutations

for $i \leftarrow 1 + 1$ **to** $1 + n'$ **do**

$p_1(x_i) \leftarrow \frac{1}{M} \sum_{m=1}^{M} p_1^{(m)}(x_i)$

end for

return $\{p_1(x_{1+1}), \ldots, p_1(x_{1+n'})\}$

end procedure

Algorithm 6 Sample new observations

Input:

$\mathcal{R}$: bandwidth parameters
$\{z_1^{[0]}, \ldots, z_B^{[0]}\}$: initial samples from proposal distribution;
$\{q(z_1^{[0]}), \ldots, q(z_B^{[0]})\}$: proposal density evaluated at initial samples;
$\{\{x_1^{(1)}, ..., x_1^{(1)}\}, \ldots, \{x_1^{(M)}, ..., x_1^{(M)}\}\}$: sets of permuted train observations;
$\{v_i^{j,(m)}\}_{i,j,m}$: prequential conditional CDFs at train observations;

Output:

$\{z_1^{[1]}, \ldots, z_B^{[1]}\}$ and $\{p_1(z_1^{[1]}), \ldots, p_1(z_B^{[1]})\}$

procedure SAMPLE
  for $m \leftarrow 1$ **to** M **do**
    for $k \leftarrow 1$ **to** B **do**
      for $j \leftarrow 1$ **to** d **do**
        Initialise $u_0^j(z_k^{[0]}) \leftarrow \Phi(z_k^{[0]})$
      end for
      Initialise $w_k^{[0]} \leftarrow p_0(z_k^{[0]})/q(z_k^{[0]})$
    end for
  end for

  for $i \leftarrow 1$ **to** 1 **do**
    for $k \leftarrow 1$ **to** B **do**
      for $m \leftarrow 1$ **to** M **do**
        for $j \leftarrow 1$ **to** d **do**
          $u_i^j(x_k)$=CDF˙UPDATE($z_k^{[i-1]}$, x_i, i, j, $u_{i-1}^j(z_k^{[i-1]})$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)
        end for
        Compute density

$$p_i^{(m)}\left(z_k^{[i-1]}\right) \leftarrow \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u_{i-1}^j(x_{1+k}), v_{i-1}^{j,(m)}; \rho_i^j(z_k^{[i-1]})\right)\right\} p_{i-1}^{(m)}\left(z_k^{[i-1]}\right)$$

      end for
      $p_i(z_k^{[i-1]}) \leftarrow \frac{1}{M}\sum_{m=1}^{M} p_i^{(m)}(z_k^{[i-1]})$
      $w_k^{[i]} \leftarrow w_k^{[i-1]} \cdot p_i\left(z_k^{[i-1]}\right)/p_{i-1}\left(z_k^{[i-1]}\right)$
    end for
    ESS $\leftarrow \left(\sum_k w_k^{[i]}\right)^2 / \left(\sum_k w_k^{[i]^2}\right)$
    if ESS $< 0.5 \cdot B$ **then**
      $\{z_k^{[i]}\}_k \leftarrow$RESAMPLE˙WITH˙REPLACEMENT$\left(\{z_k^{[i-1]}\}_k, \{w_k^{[i]}\}_k\right)$
      $w_k^{[i]} \leftarrow 1$ **for** $k = 1, \ldots, B$
    else
      $z_k^{[i]} \leftarrow z_k^{[i-1]}$ **for** $k = 1, \ldots, B$
    end if
  end for
  return $\{z_k^{[i]}\}_k$
end procedure

Appendix D. Experiments

D.1. Experimental Details

The UCI data sets (Asuncion and Newman, 2007) we used are: wine, breast, parkinson (PARKIN), ionosphere (IONO), boston housing (BOSTON), concrete (CONCR), diabetes (DIAB), and digits.

Code We downloaded the code for MAF and NSF from `https://github.com/bayesiains/nsf`, and the code for R-BP from `https://github.com/edfong/MP/tree/main/pr_copula`, and implemented EarlyStopping with patience 50, and 200 minimal, and 2000 maximal iterations. Note that we chose the autoregressive version of RQ-NSF over the coupling variant as the former seemed to generally outperform the latter in Durkan et al. (2019). The neural network in ARnet-BP was implemented with `Haiku` (Hennigan et al., 2020). The remaining methods are implemented in `sklearn`. For the DPMM with VI (mean-field approximation), we use both the diagonal and full covariance function, with default hyperparameters for the priors. The code used to generate these results is available as an additional supplementary directory.

Initialisation We initialise the predictive densities with a standard normal, the bandwidth parameter with $\rho_0 = 0.9$, the length scales with $l_2 = 1, ..., l_{d-1} = 1$, and the neural network weights inside ARnet-BP by sampling from a truncated normal with variance proportional to the number of input nodes of the layer.

Data pre-processing For each dataset, we standardized each of the attributes by mean-centering and rescaling to have a sample standard deviation of one. Following Papamakarios et al. (2017), we eliminated discrete-valued attributes. To avoid issues arising from collinearity, we also eliminated one attribute from each pair of attributes with a Pearson correlation coefficient greater than 0.98.

Hyperparameter tuning We average over $M = 10$ permutations over samples and features. The bandwidth of the KDEs was found by five-fold cross validation over a grid of 80 log-scale-equidistant values from $\rho = 0.1$ to 100. For the DPMM, we considered versions with a diagonal (Diag) and full (Full) covariance matrix for each mixture component. We optimized over the weight concentration prior of the DPMM by five-fold cross validation with values ranging from 10^{-40} to 1. The model was trained with variational inference using `sklearn`. The hyperparameters of MAFs and RQ-NSFs were found with a Bayesian optimisation search. For MAF and RQ-NSF, we applied a Bayesian optimisation search over the learning rate $\{3 \cdot 10^{-4}, 4 \cdot 10^{-4}, 5 \cdot 10^{-4}\}$, the batch size $\{512, 1024\}$, the flow steps $\{10, 20\}$, the hidden features $\{256, 512\}$, the number of bins $\{4, 8\}$, the number of transform blocks $\{1, 2\}$ and the dropout probability $\{0, 0.1, 0.2\}$. On each data set, the hyperparameter search ran for more than 5 days. Please see Table 3 for the optimal parameters found. For the benchmark UCI data sets, we did not tune the hyperparameters for neither MAF nor RQ-NSF but instead used the standard parameters given by Durkan et al. (2019). The kernel parameters of the GP are optimised during training, the α resp. λ intialization parameter of the linear model over the range from 1 to 2 resp. 0.01 to 0.1, and the hidden layer sizes of the MLP over the values $\{64, 128, 256\}$.

Compute We run all BP and neural network experiments on a single Tesla V100 GPU, as provided in the internal cluster of our department. In total, these experiments required compute of approximately 4000 GPU hours. The remaining experiments were run on a single core of an Intel(R) Xeon(R) Gold 6240 CPU @ 2.60GHz, using up a total of 100 hours.

Table 3: Hyperparameters for MAF and RQ-NSF

	data	batch size	learning rate	flow steps	hidden nodes	bins	transform blocks	dropout
MAF	WINE	10000	0.0003	20	512	-	1	0.2
	BREAST	10000	0.0004	20	512	-	1	0.2
	PARKINSONS	10000	0.0004	20	512	-	1	0.2
	IONOSPHERE	10000	0.0003	20	512	-	1	0.2
	BOSTON	10000	0.0003	10	512	-	1	0.2
	CONCRETE	1024	0.0003	10	512	-	1	0.2
	DIABETES	10000	0.0004	20	512	-	1	0.2
	CHECKERBOARD	10000	0.0003	20	512	-	1	0.2
RQ-NSF	WINE	10000	0.0004	20	512	8	1	0.2
	BREAST	10000	0.0005	10	512	8	1	0.2
	PARKINSONS	10000	0.0005	20	512	8	1	0.2
	IONOSPHERE	10000	0.0003	10	512	8	1	0.2
	BOSTON	10000	0.0003	10	512	8	1	0.2
	CONCRETE	1024	0.0004	20	256	8	2	0.1
	DIABETES	256	0.0004	10	512	8	2	0.2
	CHECKERBOARD	1024	0.0004	10	512	8	2	0.1

D.2. Additional Experimental Results

Computational analysis For the computational study, we consider data sampled from a Gaussian mixture model (GMM). By default, we set the number of training samples to $n = 500$, the number of test samples to $n' = 500$, the number of features to $d = 2$, the number of mixture components to $K = 2$, and the number of feature and samples permutations to 1. In Figure 6, we plot the compute in elapsed seconds w.r.t changes in these parameters.

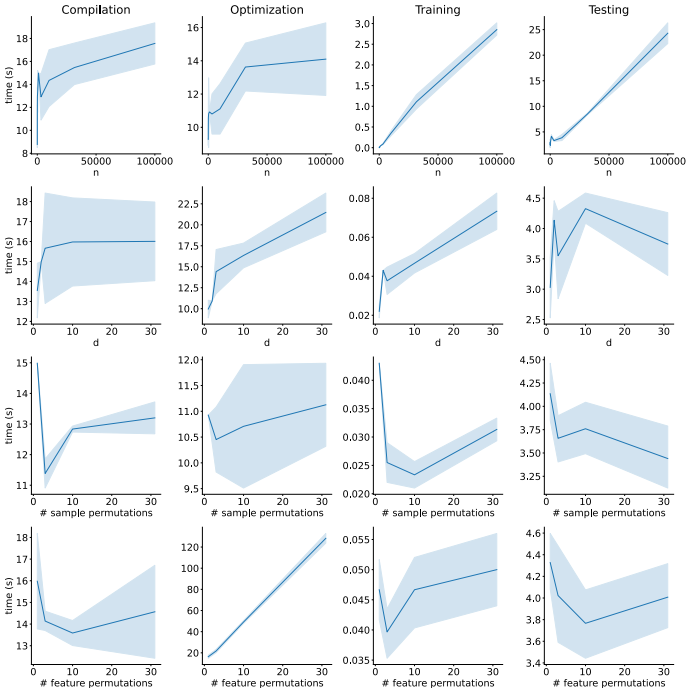
Sensitivity analysis For the sensitivity study, we consider the same simulated GMM data as in the computational study, and plot the results in Figure 7. As expected, we observe that the test NLL decreases in n, and in the number of permutations. It also decreases in the number of mixture components. One possible explanation for this is that, as noted by Hahn et al. (2018), R-BP can be interpreted as a mixture of n normal distributions. The NLL decreases in d, as the mixture components are easier to distinguish in higher dimensional covariate spaces.

Ablation study Figure 7 shows the test NLL of ARnet-BP and AR-BP for the above GMM example, as a function of the number of sample permutations, and number of feature permutations. We see that averaging over multiple permutations is crucial to the performance of AR-BP. In Table 4, we also show results on the small UCI datasets for:

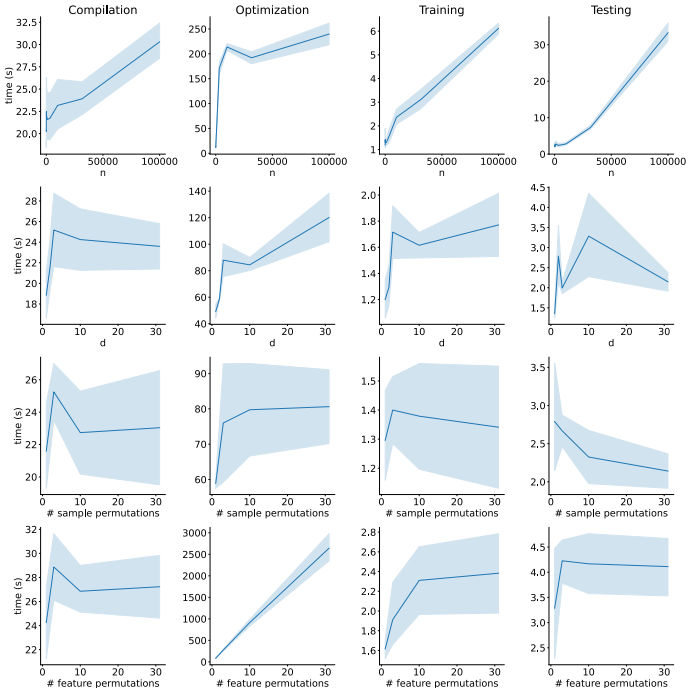
- a different choice of covariance function, namely a rational quadratic covariance function, defined by $k(x, x_i) = \left(1 + \frac{||x-x_i||_2^2}{2\gamma\ell^2}\right)^{-\gamma}$, where $\ell, \gamma > 0$ and
- a different choice of initial distribution, namely a uniform distribution (unif).

We observe that none of these ablations consistently outperforms AR_d-BP.

Benchmark UCI data sets As we only presented a subset of the results on the benchmark data sets introduced by Papamakarios et al. (2017) in Section 4, we present more results for density estimation



(a) AR-BP



(b) ARnet-BP

Figure 6: Computational study: computational time measured in elapsed seconds for a simple GMM example. Note that R-BP has the same computational complexity and only saves an indiscernible constant time factor.

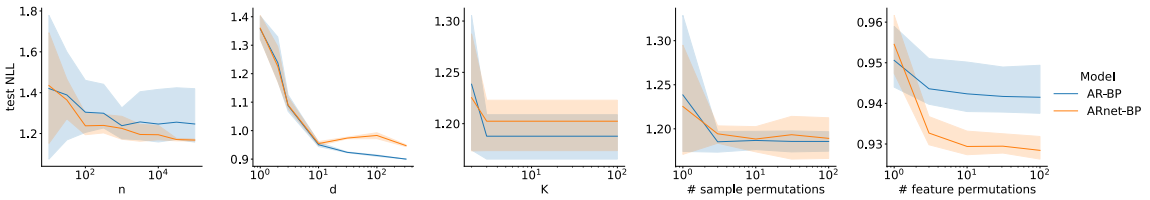


Figure 7: Sensitivity analyis: Average test NLL over 5 runs reported with standard error for a simple GMM example over a range of simulation and parameter settings.

Table 4: Average NLL with standard error over five runs on five UCI data sets of small-to-moderate size

	WINE	BREAST	PARKIN	IONO	BOSTON
n/d	89/12	97/14	97/16	175/30	506/13
AR$_d$-BP	$\mathbf{13.22_{\pm 0.04}}$	$\mathbf{6.11_{\pm 0.04}}$	$\mathbf{7.21_{\pm 0.12}}$	$\mathbf{16.48_{\pm 0.26}}$	$-14.75_{\pm 0.89}$
AR-BP (RQ)	$13.53_{\pm 0.02}$	$7.39_{\pm 0.06}$	$8.79_{\pm 0.08}$	$21.26_{\pm 0.08}$	$4.49_{\pm 0.00}$
AR$_d$-BP (RQ)	$13.36_{\pm 0.04}$	$6.18_{\pm 0.03}$	$7.85_{\pm 0.08}$	$20.25_{\pm 0.09}$	$\mathbf{-20.41_{\pm 1.28}}$
AR$_d$-BP (unif)	$-5.18_{\pm 0.04}$	$-15.51_{\pm 0.11}$	$-16.58_{\pm 0.06}$	$-47.77_{\pm 3.77}$	$-10.73_{\pm 1.63}$

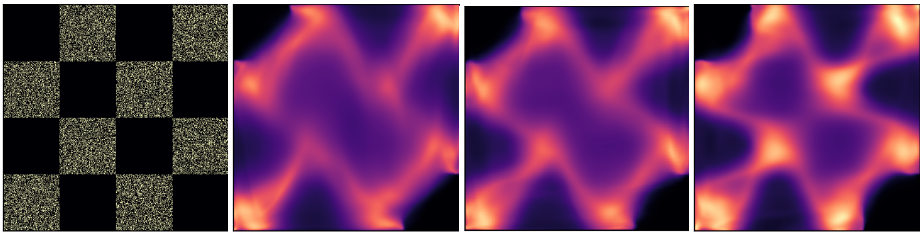
on the complete data set in Table 5. These results underscore that 1) MAF and RQ-NSF outperform any other baseline, the more data is available; 2) KDE underperforms in high-dimensional settings; 3) DPMM is not suitable for every data distribution. Note that evaluation of the R-BP variants take at least 4 days to run on any of the data sets with more than 800,000 observations which is why we omitted those results here.

Table 5: Average NLL with standard error over five runs on benchmark UCI data from Papamakarios et al. (2017)

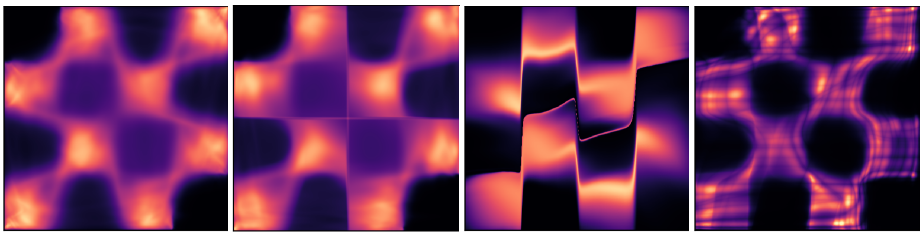
	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
n/d	1,659,917/6	852,174/8	315,123/21	29,556/43	1,000,000/ 63
Gaussian	$7.73_{\pm 0.00}$	$3.59_{\pm 0.00}$	$27.93_{\pm 0.00}$	$37.20_{\pm 0.00}$	$56.45_{\pm 0.00}$
KDE	$29.39_{\pm 0.00}$	$\mathbf{-9.61_{\pm 0.00}}$	$26.44_{\pm 0.00}$	$43.88_{\pm 7.52}$	$63.70_{\pm 10.00}$
DPMM (Diag)	$0.51_{\pm 0.01}$	$1.20_{\pm 0.02}$	$25.80_{\pm 0.00}$	$39.16_{\pm 0.01}$	$37.55_{\pm 0.02}$
DPMM (Full)	$0.33_{\pm 0.00}$	$-5.57_{\pm 0.04}$	$23.40_{\pm 0.02}$	$18.82_{\pm 0.01}$	$4.47_{\pm 0.00}$
MAF	$0.52_{\pm 0.00}$	$-2.21_{\pm 0.54}$	$21.10_{\pm 0.04}$	$12.81_{\pm 0.08}$	$2.76_{\pm 0.17}$
RQ-NSF	$\mathbf{0.00_{\pm 0.01}}$	$-6.41_{\pm 0.14}$	$\mathbf{19.46_{\pm 0.08}}$	$\mathbf{12.51_{\pm 0.19}}$	$\mathbf{2.44_{\pm 0.56}}$

Image examples We provide preliminary results on two image datasets, digits and MNIST, in Table 6. Note that the AR-BP copula updates investigated here were not designed with computer vision tasks in mind. The rich parameterization allows the model to overfit to the data leading to a prequential negative log-likelihood of at least -684 at train time while the test NLL is considerably higher. ARnet-BP, on the other hand, helps to model the complex data structure more efficiently. We expect that further extensions based on, for instance, convolutional covariance functions (Van der Wilk et al., 2017) may prove fruitful.

Toy examples Figure 8 shows density estimates for the introductory example of the checkerboard distribution in a large data regime. We observe that neural-network-based methods outperform the AR-BP alternatives. Nevertheless, AR-BP performs better than the baseline R-BP. An illustration of



(a) True data (b) R-BP (c) R_d-BP (d) AR-BP



(e) AR_d-BP (f) ARnet-BP (g) MAF (h) RQ-NSF

Figure 8: Scatter plot and density estimates of 60,000 observations sampled from a chessboard data distribution. Test log likelihoods are R-BP: $2.25_{\pm 0.0}$, R_d-BP : $2.19_{\pm 0.0}$, AR-BP: $2.21_{\pm 0.0}$, AR_d-BP: $2.10_{\pm 0.0}$, ARnet BP : $2.19_{\pm 0.0}$, MAF : $2.09_{\pm 0.0}$, RQ-NSF : $2.05_{\pm 0.0}$.

Table 6: Image datasets: average test NLL over five runs displayed with standard error

	DIGITS	MNIST
MAF	$\mathbf{-8.76_{\pm 0.10}}$	$-7.14_{\pm 0.48}$
RQ-NSF	$-6.17_{\pm 0.13}$	$-8.49_{\pm 0.03}$
R-BP	$-8.80_{\pm 0.00}$	$-9.04_{\pm 0.07}$
R_d-BP	$-7.46_{\pm 0.12}$	$-7.73_{\pm 0.07}$
AR-BP	$-8.66_{\pm 0.03}$	$-7.31_{\pm 42.54}$
AR_d-BP	$-7.46_{\pm 0.18}$	$-8.32_{\pm 61.92}$
ARnet-BP	$-7.72_{\pm 0.28}$	$\mathbf{-9.20_{\pm 0.10}}$

this behaviour on another toy example is also given in Figure 9. Figure 10 shows density estimates from AR-BP on a number of complex distributions.

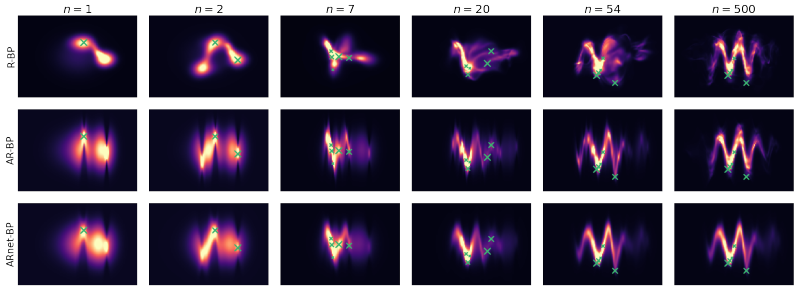


Figure 9: Illustration of the importance of an autoregressive kernel. We trained the models on 500 data points sampled according to a sine wave distribution (given in Figure 10). We visualise the predictive density after observing a different number, n, of observations, highlighting the last five points with ×. We observe that for highly non-linear relationships between x^1 and x^2, the optimal bandwidth of R-BP is quite high ($\rho = 0.93$) which results in strong overfitting. Even when we choose $\rho_0 = 0.93$ for AR-BP and ARnet-BP, we observe that these models learn the true data distribution with fewer samples than R-BP does.

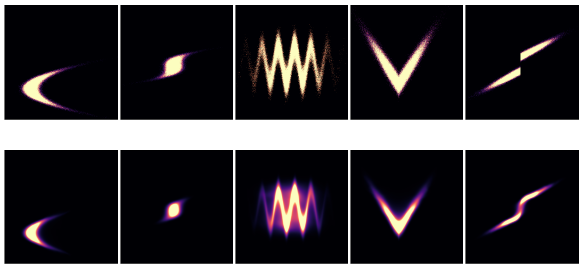


Figure 10: Scatter plots of 60,000 samples from different data distributions in the first row, and corresponding autoregressive predictive density estimates in the second row.