Quasi-Bayesian Nonparametric Density Estimation via Autoregressive Predictive Updates(Supplementary material)

Sahra Ghalebikesabi[1] **Chris Holmes**[2] **Edwin Fong**[*2] **Brieuc Lehmann**[*3]

[1]University of Oxford
[2]Novo Nordisk
[3]University College London

A DERIVATIONS

A.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].

A.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

$$h_1(x, x_1) = \frac{E\left[f_G(x)\, f_G(x_1)\right]}{p_0(x)\, p_0(x_1)} \tag{1}$$

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

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

*equal contribution

where $w_k = v_k \prod_{j<k}\{1-v_j\}$, $v_k \overset{iid}{\sim} \mathrm{Beta}(1,a)$ 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 (1) 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

$$\frac{E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]}{p_0(x)\, p_0(x_1)} = \prod_{j=1}^{d} \frac{E\left[K(x^j \mid \theta^{*j})\, K(x_1^j \mid \theta^{*j})\right]}{p_0(x^j)\, p_0(x_1^j)}. \tag{2}$$

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 cumulative distribution function (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.

A.1.2 With Autoregression (AR-BP)

For the derivation of the AR-BP update, we can follow the arguments in the previous section until (2) where the factorised kernel assumption applies for the first time. For AR-BP, we instead have

$$\frac{E\left[K(x \mid \theta^*)\, K(x_1 \mid \theta^*)\right]}{p_0(x)\, p_0(x_1)} = \prod_{j=1}^{d} \frac{E\left[K\{x^j \mid \theta^{*j}(x^{1:j-1})\}\, K\{x_1^j \mid \theta^{*j}(x^{1:j-1})\}\right]}{p_0(x^j)\, p_0(x_1^j)}. \tag{3}$$

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 $\theta(\cdot) \sim \mathrm{GP}\{0, \tau^{-1}k\}$. 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 $\rho_0 = 1/(1+\tau)$.

A.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.

A.2.1 First Update Step

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

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

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

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

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 $p_0(y \mid x) = \mathcal{N}(y; 0, \sigma^2 + \tau^{-1})$.

A.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

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

where μ^y_{i-1} is the posterior mean of the GP at x and likewise for $\mu^{y_{i-1}}_{i-1}$. 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.

A.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

$$p_i(y \mid x) = \left[1-\alpha_i(x,x_i)+\alpha_i(x,x_i)\, c\left\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i);\rho\right\}\right] p_{i-1}(y \mid x), \tag{4}$$

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 $P_{i-1}(y \mid x) = \mathcal{N}(y \mid \mu_{i-1}(x), \sigma^2_y)$. This is an accurate approximation if the truth is normal and we have observed sufficient observations. Without loss of generalizability, we assume that $\sigma^2_y = 1$. 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 (4), we get

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

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

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

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

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

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.

A.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 $y_{1:n} \in \mathbb{R}^n$.

A.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

$$f_{G_x}(y) = \int \mathcal{N}(y \mid \theta, 1)\, dG_x(\theta), \quad G_x = \sum_{l=1}^{\infty} w_l(x)\, \delta_{\theta_l^*(x)}. \tag{5}$$

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,

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

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 A.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

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

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

$$\beta_i(x, x_i) = \frac{\alpha_i \prod_{j=1}^{d} c\left\{\Phi\left(x^j\right), \Phi\left(x_i^j\right); \rho_i^j(x^{1:j-1})\right\}}{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left\{\Phi\left(x^j\right), \Phi\left(x_i^j\right); \rho_i^j(x^{1:j-1})\right\}}. \tag{6}$$

A.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

$$b\{q_{i-1}, r_{i-1}; \rho_i(x)\} = \begin{cases} 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} \wedge r_{i-1}}{q_{i-1}\, r_{i-1}} & \text{if } y = y_i \\[2ex] 1 - \rho_i(x) + \rho_i(x)\, \dfrac{q_{i-1} - \{q_{i-1} \wedge (1 - r_{i-1})\}}{q_{i-1}\, r_{i-1}} & \text{if } y \neq y_i. \end{cases}$$

B METHODOLOGY

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

B.1 GENERATIVE MODELLING

First, we consider three approaches to generative modelling

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

B.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

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

B.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 $z_1, \dots, z_B$ from the initial predictive p_0.
2. Re-sampling $z_1, \dots, z_B$ with replacement based on the weights $w_1 = p_n(z_1)/p_0(z_1), \dots, w_B = p_n(z_B)/p_0(z_B)$.

B.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 1, 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.

B.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 A.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 $\{f_x(y) : x \in \mathbb{R}^d\}$.

B.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 Dirichlet Process Mixture Model (DPMM):

$$f_{G_x}(y) = \int \mathcal{N}(y \mid \theta, 1)\, dG_x(\theta), \quad G_x = \sum_{k=1}^{\infty} w_k(x)\, \delta_{\theta_k^*}, \tag{7}$$

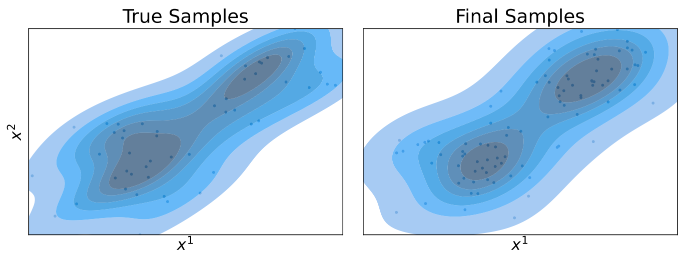
where $w_k(\mathbf{x})$ follows an x-dependent stick-breaking process. Our contribution is to assume an autoregressive factorisation of the kernel and independent GP priors on θ_k^*. See Supplement A.4.1 for the derivation of the predictive density update that is now given by

$$p_i(y \mid x) = [1 - \beta_i(x, x_i) + \beta_i(x, x_i)\, c\,\{P_{i-1}(y \mid x), P_{i-1}(y_i \mid x_i); \rho_i(x)\}]\; p_{i-1}(y \mid x), \tag{8}$$

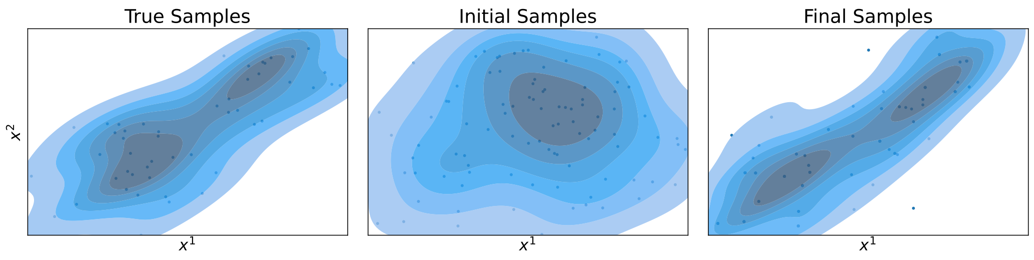
where $\rho_i(x) = \rho_i^{d+1}(x)$ and β as in (6).

B.2.2 Classification

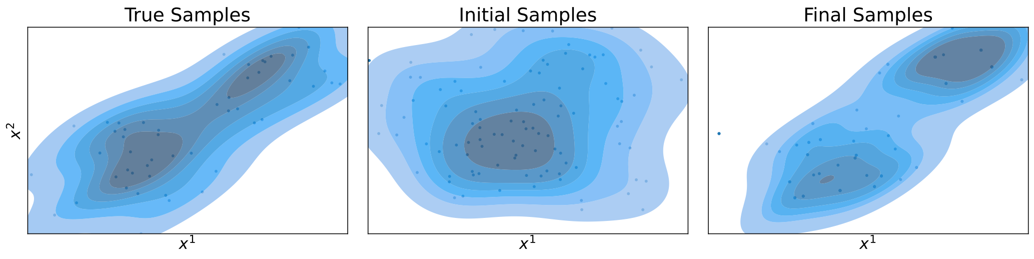
For $y_i \in \{0, 1\}$, Fong et al. [2021] assume a beta-Bernoulli mixture. As explained in Supplement A.4.2 and Fong et al. [2021], this gives the same update as in the regression setting with the difference that the copula c in (8) is



(a) Inverse Sampling



(b) Importance Sampling



(c) Sequential Monte Carlo

Figure 1: 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.

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 $\rho_y \in (0,1)$.

B.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

$$u_i^j = \left\{ (1-\alpha_i) u_{i-1}^j + \alpha_i H\left(u_{i-1}^j, v_{i-1}^j; \rho_i^j\right) \prod_{r=1}^{k-1} c\left(u_{i-1}^r, v_{i-1}^r; \rho_i^r\right) \right\} \frac{p_{i-1}\left(x^{1:k-1}\right)}{p_i\left(x^{1:k-1}\right)}, \tag{9}$$

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:**
$p_n(x_{n+1}), \ldots p_n(x_{n+n'})$: density of test points

```
1: procedure FULL__DENSITY__ESTIMATION
2:     Compute optimal bandwidth parameters                     ▷ $\mathcal{O}(Mn_\rho^2 d \cdot \#\text{gradient steps})$
3:     Compute $v_i^{j,(m)}$ for $i \in \{1,\ldots,n\}, j \in \{1,\ldots,d\}, m \in \{1,\ldots,M\}$     ▷ $\mathcal{O}(Mn^2 d)$
4:     Evaluate density at test observations $x_{n+1:n+n'}$     ▷ $\mathcal{O}(Mnn'd)$
5: 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

```
1: procedure OPTIMAL__BANDWIDTH__AND__LENGTHSCALES
2:     Subsample $\{x'_1, \ldots, x'_{n_\rho}\}$ from $x_{1:n}$
3:     for $s \leftarrow 1$ to maxiter do
4:         _, $\{p_{i-1}^{(m)}(x'_i)\}_{i,m} \leftarrow$ FIT__CONDITIONAL__PREDICTIVE__CDF(
                 $\mathcal{R}^{(s-1)}, \{x'_1, \ldots, x'_{n_\rho}\}, M$, fit_density=True)
5:         Compute $L(x'_1, \ldots, x'_{n_\rho}) = -\sum_{m=1}^{M} \sum_{i=1}^{n_\rho} \log p_{i-1}^{(m)}(x'_i)$
6:         $\mathcal{R}^{(s)} \leftarrow$ ADAM__STEP($\mathcal{R}^{(s-1)}$, $L$)
7:     end for
8:     return $\mathcal{R}^{(s)}$
9: 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;
    $u^j_{i-1}(z)$: predictive CDF for $z$;
    $v^{j,(m)}_{i-1}$: prequential CDF;
    $\rho^j_i(z^{1:j-1})$=None: bandwidth;
    $\mathcal{R}$=None: bandwidth parameters;
Output:
    $u_i(z)$

1: procedure CDF_UPDATE
2:     Compute
           $$\rho^j_i(z^{1:j-1}) \leftarrow \rho_0 k_{\mathcal{R}}\left(z^{1:j-1}, x^{1:j-1}_i\right)$$
       where $k_{\mathcal{R}}$ denotes the user-defined kernel if $\rho$ =None
3:     Compute the bivariate Gaussian copula density
           $$c\{u^j_{i-1}(z), v^{j,(m)}_{i-1}; \rho_j\} \leftarrow \frac{\mathcal{N}_2\left\{\Phi^{-1}(u^j_{i-1}(z)), \Phi^{-1}(v^{j,(m)}_{i-1}) \mid 0, 1, \rho^j_i(z^{1:j-1})\right\}}{\mathcal{N}\{\Phi^{-1}(u^j_{i-1}(z)) \mid 0,1\}\mathcal{N}\{\Phi^{-1}(v^{j,(m)}_{i-1}) \mid 0,1\}}$$
4:     Compute the conditional Gaussian copula CDF
           $$H\left\{u^j_{i-1}(z), v^{j,(m)}_{i-1}; \rho^j_i(z^{1:j-1})\right\} \leftarrow \Phi\left\{\frac{\Phi^{-1}(u^j_{i-1}(z)) - \rho^j_i(z^{1:j-1})\Phi^{-1}(v^{j,(m)}_{i-1})}{\sqrt{1-\rho^j_i(z^{1:j-1})^2}}\right\}$$
5:     Compute $\alpha_i = (2 - \frac{1}{i})\frac{1}{i+1}$
6:     Compute $u^j_i(z) = P^j_i(z|z^{1:j-1})$ by
           $$u^j_i(z) \leftarrow \left\{(1-\alpha_i)u^j_{i-1}(z) + \alpha_i H\left(u^j_{i-1}(z), v^{j,(m)}_{i-1}; \rho^j_i(z)\right)\prod_{l=1}^{j-1} c\left(u^l_{i-1}(z), v^{l,(m)}_{i-1}; \rho^j_i(z)\right)\right\}$$
           $$\cdot 1 \Big/ \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u^j_{i-1}(z), v^{j,(m)}_{i-1}; \rho^j_i(z)\right)\right\}$$
7:     return $u_i(z)$
8: 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 $\{v_{i-1}^{j,(m)}\}_{i,j,m}$

```
 1: procedure FIT_CONDITIONAL_PREDICTIVE_CDF
 2:     for $m \leftarrow 1$ to $M$ do
 3:         Sample permutation $\pi_1 \in \Pi(n), \pi_2 \in \Pi(d)$
 4:         Change the ordering of the training observations $\{x_1^{(m)}, \ldots, x_n^{(m)}\} \leftarrow$
            $\{\pi_1(x_1), \ldots, \pi_1(x_n)\}$ and the features $x \leftarrow [\pi_2(x^1), \ldots, \pi_2(x^d)]$    ▷ For
            simplicity we will drop the superscript in the following
 5:         for $j \leftarrow 1$ to $d$ do
 6:             for $k \leftarrow 1$ to $n$ do
 7:                 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
 8:             end for
 9:         end for
10:         for $i \leftarrow 1$ to $n$ do
11:             Set $v_{i-1}^{j,(m)} \leftarrow u_{i-1}^j(x_i)$ for $j \leftarrow 1$ to $d$
12:             for $k \leftarrow 1$ to $i$ do
13:                 for $j \leftarrow 1$ to $d$ do
14:                     $u_i^j(x_k)$=CDF_UPDATE($x_k$, $x_i$, $i$, $j$, $u_{i-1}^j(x_k)$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)
15:                 end for
16:                 if compute_density then
17:
                        $$p_i^{(m)}(x_k) \leftarrow \left\{1 - \alpha_i + \alpha_i \prod_{j=1}^{d} c\left(u_{i-1}^j(x_k), v_{i-1}^{j,(m)}; \rho_i^j(x_k)\right)\right\} p_{i-1}^{(m)}(x_k)$$
18:                 end if
19:             end for
20:         end for
21:     end for
22:     return $\{v_{i-1}^{j,(m)}\}_{i,j,m}, \{p_{i-1}^{(m)}(x_i)\}_{i,m}$ if compute_density else $\{v_{i-1}^{j,(m)}\}_{i,j,m}$
23: end procedure
```

Algorithm 5 Evaluate density at test observations

```
Input:
    $\mathcal{R}$: bandwidth parameters
    $x_{n+1:n+n'}$: test observations;
    $\{\{x_1^{(1)}, ..., x_n^{(1)}\}, \dots, \{x_1^{(M)}, ..., x_n^{(M)}\}\}$: 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:
    $\{p_n(x_{n+1}), \dots, p_n(x_{n+n'})\}$

  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 $u_0^j(x_{n+k}) \leftarrow \Phi(x_{n+k}^j)$
              end for
          end for

          for $i \leftarrow 1$ to $n$ do
              for $k \leftarrow 1$ to $n'$ do
                  for $j \leftarrow 1$ to $d$ do
                      $u_i^j(x_k)$=CDF__UPDATE($x_{n+k}$, $x_i$, $i$, $j$, $u_{i-1}^j(x_{n+k})$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)
                  end for
                  Compute density

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

              end for
          end for
      end for

      ▷ Average density over permutations
      for $i \leftarrow n+1$ to $n+n'$ do
          $p_n(x_i) \leftarrow \frac{1}{M}\sum_{m=1}^M p_n^{(m)}(x_i)$
      end for
      return $\{p_n(x_{n+1}), \dots, p_n(x_{n+n'})\}$
  end procedure
```

Algorithm 6 Sample new observations

Input:

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

Output:

$\{z_1^{[n]}, \dots, z_B^{[n]}\}$ and $\{p_n(z_1^{[n]}), \dots, p_n(z_B^{[n]})\}$

```
1:  procedure SAMPLE
2:      for $m \leftarrow 1$ to $M$ do
3:          for $k \leftarrow 1$ to $B$ do
4:              for $j \leftarrow 1$ to $d$ do
5:                  Initialise $u_0^j(z_k^{[0]}) \leftarrow \Phi(z_k^{[0]})$
6:              end for
7:              Initialise $w_k^{[0]} \leftarrow p_0(z_k^{[0]})/q(z_k^{[0]})$
8:          end for
9:      end for
10:     for $i \leftarrow 1$ to $n$ do
11:         for $k \leftarrow 1$ to $B$ do
12:             for $m \leftarrow 1$ to $M$ do
13:                 for $j \leftarrow 1$ to $d$ do
14:                     $u_i^j(x_k)$=CDF_UPDATE($z_k^{[i-1]}$, $x_i$, $i$, $j$, $u_{i-1}^j(z_k^{[i-1]})$, $v_{i-1}^{j,(m)}$, $\mathcal{R}$)
15:                 end for
16:                 Compute density
                    $$p_i^{(m)}\left(z_k^{[i-1]}\right) \leftarrow \left\{1-\alpha_i+\alpha_i\prod_{j=1}^d c\left(u_{i-1}^j(x_{n+k}), v_{i-1}^{j,(m)}; \rho_i^j(z_k^{[i-1]})\right)\right\} p_{i-1}^{(m)}\left(z_k^{[i-1]}\right)$$
17:             end for
18:             $p_i(z_k^{[i-1]}) \leftarrow \frac{1}{M}\sum_{m=1}^M p_i^{(m)}(z_k^{[i-1]})$
19:             $w_k^{[i]} \leftarrow w_k^{[i-1]} \cdot p_i\left(z_k^{[i-1]}\right)/p_{i-1}\left(z_k^{[i-1]}\right)$
20:         end for
21:         ESS $\leftarrow \left(\sum_k w_k^{[i]}\right)^2 / \left(\sum_k {w_k^{[i]}}^2\right)$
22:         if ESS $< 0.5 \cdot B$ then
23:             $\{z_k^{[i]}\}_k \leftarrow$RESAMPLE_WITH_REPLACEMENT$\left(\{z_k^{[i-1]}\}_k, \{w_k^{[i]}\}_k\right)$
24:             $w_k^{[i]} \leftarrow 1$ for $k = 1, \dots, B$
25:         else
26:             $z_k^{[i]} \leftarrow z_k^{[i-1]}$ for $k = 1, \dots, B$
27:         end if
28:     end for
29:     return $\{z_k^{[i]}\}_k$
30: end procedure
```

C EXPERIMENTS

C.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.

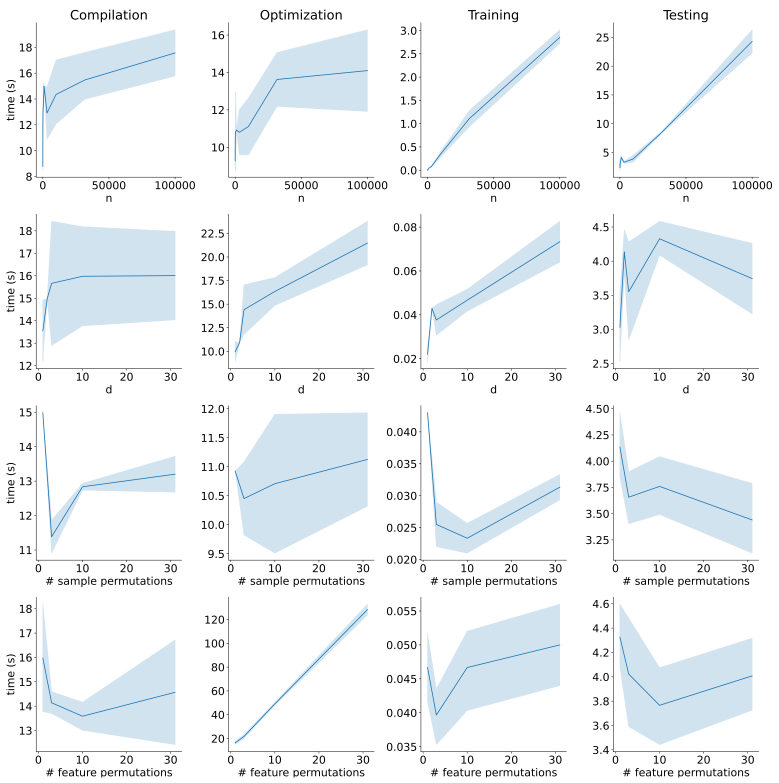
Data pre-processing For each dataset, we standardized each of the attributes by mean-centering and rescaling to have a sample standard deviation of one. Following Papamakarios et al. [2017], we eliminated discrete-valued attributes. To avoid issues arising from collinearity, we also eliminated one attribute 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 kernel density estimations (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 masked autoregressive flows (MAFs) and rational-quadratic neural spline flows (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 1 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 Gaussian process (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 $\{64, 128, 256\}$.

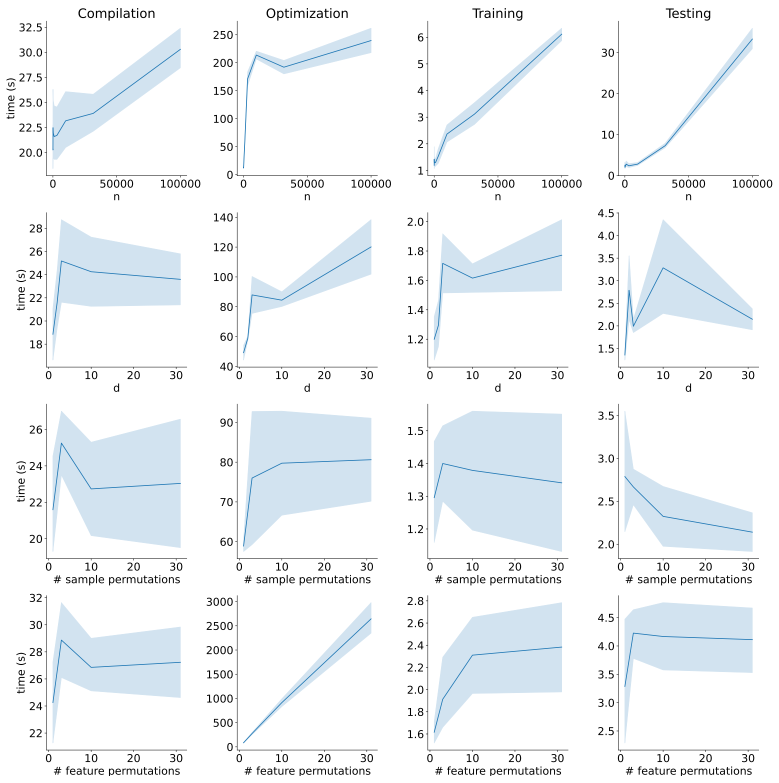
Compute We run all BP and neural network experiments on a single Tesla V100 GPU, as provided in the internal cluster of our department. In total, these experiments required compute of approximately 4000 GPU hours. The remaining experiments were run on a single core of an Intel(R) Xeon(R) Gold 6240 CPU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

C.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 2, we plot the compute in elapsed seconds w.r.t changes in these parameters.



(a) AR-BP



(b) ARnet-BP

Figure 2: 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.

Table 1: 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

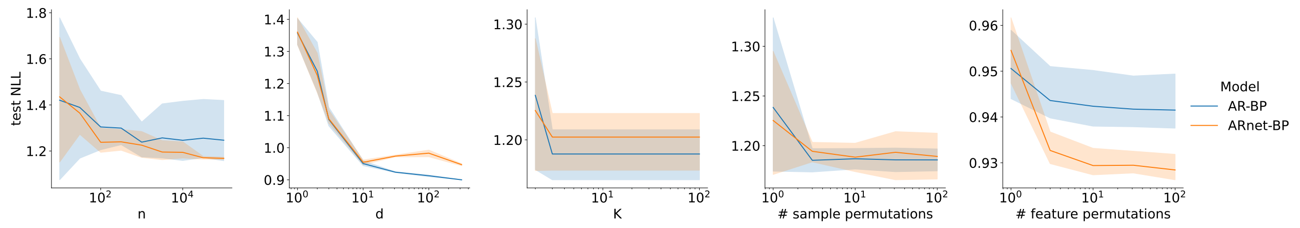


Figure 3: Sensitivity analyis: Average test NLL over 5 runs reported with standard error for a simple GMM example over a range of simulation and parameter settings.

Sensitivity analysis For the sensitivity study, we consider the same simulated GMM data as in the computational study, and plot the results in Figure 3. As expected, we observe that the test negative log-likelihood (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 3 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 2, we also show results on the small UCI datasets for:

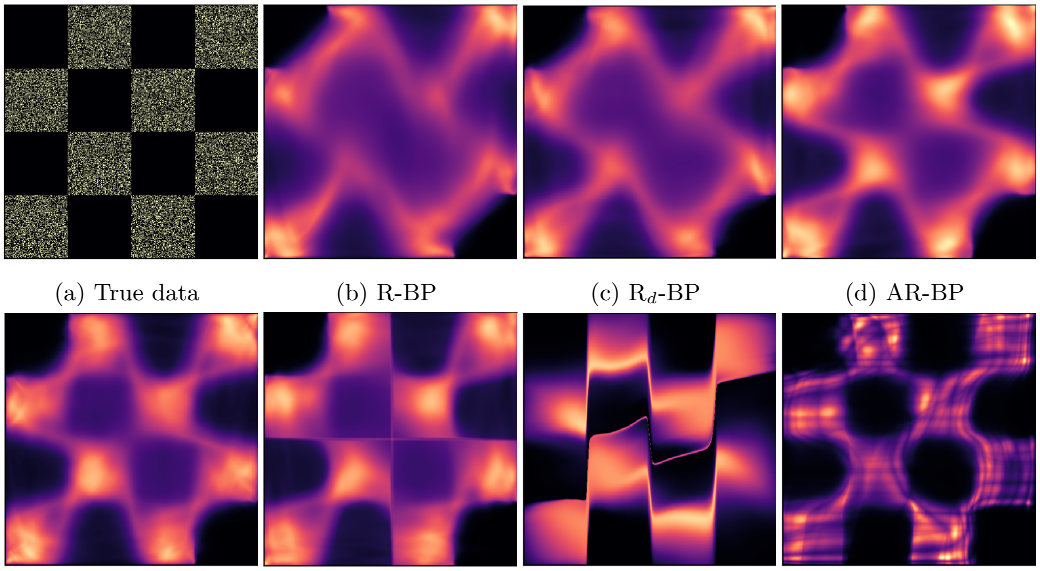
- a different choice of covariance function, namely a rational quadratic covariance function, defined by $k(x, x_i) = \left(1 + \frac{||x-x_i||_2^2}{2\gamma\ell^2}\right)^{-\gamma}$, where $\ell, \gamma > 0$ and
- a different choice of initial distribution, namely a uniform distribution (unif).

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 5, we present more results for density estimation on the complete data set in Table 3. 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

Table 2: 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	**13.22**$_{\pm 0.04}$	**6.11**$_{\pm 0.04}$	**7.21**$_{\pm 0.12}$	**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)	$13.36_{\pm 0.04}$	$6.18_{\pm 0.03}$	$7.85_{\pm 0.08}$	$20.25_{\pm 0.09}$	**−20.41**$_{\pm 1.28}$
AR$_d$-BP (unif)	$-5.18_{\pm 0.04}$	$-15.51_{\pm 0.11}$	$-16.58_{\pm 0.06}$	$-47.77_{\pm 3.77}$	$-10.73_{\pm 1.63}$


(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 4: 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 : $2.05_{\pm 0.0}$.

more than 800,000 observations which is why we omitted those results here.

Table 3: 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	$29.39_{\pm 0.00}$	**−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	**0.00**$_{\pm 0.01}$	$-6.41_{\pm 0.14}$	**19.46**$_{\pm 0.08}$	**12.51**$_{\pm 0.19}$	**2.44**$_{\pm 0.56}$

Image examples We provide preliminary results on two image datasets, digits and MNIST, in Table 4. 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 4 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 this behaviour on another

Table 4: Image datasets: average test NLL over five runs displayed with standard error

	DIGITS	MNIST
MAF	$\mathbf{-8.76}_{\pm\mathbf{0.10}}$	$-7.14_{\pm 0.48}$
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	$-7.72_{\pm 0.28}$	$\mathbf{-9.20}_{\pm\mathbf{0.10}}$

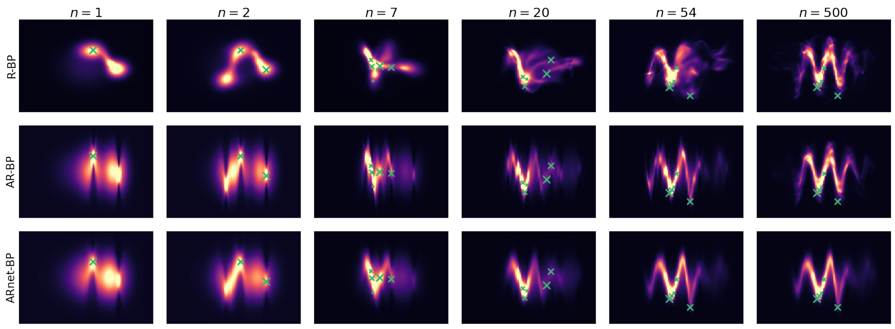


Figure 5: Illustration of the importance of an autoregressive kernel. We trained the models on 500 data points sampled according to a sine wave distribution (given in Figure 6). We visualise the predictive density after observing a different number, n, of observations, highlighting the last five points with $\times$. 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.

toy example is also given in Figure 5. Figure 6 shows density estimates from AR-BP on a number of complex distributions.

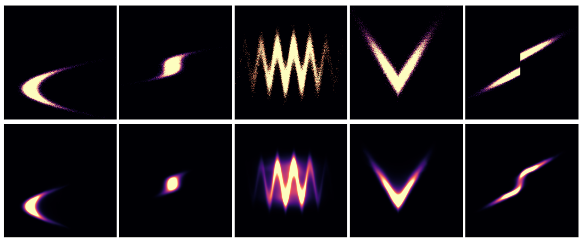


Figure 6: Scatter plots of 60,000 samples from different data distributions in the first row, and corresponding autoregressive predictive density estimates in the second row.

REFERENCES

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

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

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

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.

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`.

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.

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.

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