ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10447783

END-TO-END LEARNING OF GAUSSIAN MIXTURE PROPOSALS USING DIFFERENTIABLE PARTICLE FILTERS AND NEURAL NETWORKS

Benjamin Cox[⋆]*, Sara Pérez-Vieites*[†]*, Nicolas Zilberstein*[‡]*, Martin Sevilla*[‡]*, Santiago Segarra*[‡]*, Víctor Elvira*[⋆]

[⋆] School of Mathematics, University of Edinburgh, United Kingdom
[†] IMT Nord Europe, Institut Mines-Télécom, Centre for Digital Systems, France
[‡] Electrical and Computing Engineering, Rice University, USA

ABSTRACT

We introduce a new method, named PropMixNN, that uses a neural network to learn the proposal distribution of a particle filter. The optimal proposal distribution is approximated as a multivariate Gaussian mixture, so the proposed method aims at learning the means and covariance matrices of the S components that characterise the mixture. This unsupervised method is trained to target the log-likelihood, which does not require knowledge of the hidden state. The performance of the method is assessed in a stochastic Lorenz 96 model, which presents a non-linear chaotic behaviour. The proposed method reduces estimation errors in comparison with the state-of-the-art, showing greater improvement in highly non-linear scenarios.

***Index Terms*—** machine learning, particle filtering, neural networks, differentiable particle filter, proposal distribution, Gaussian mixture

1. INTRODUCTION

Challenges of many fields of science need the estimation of the evolution of dynamical systems. Some examples can be found in engineering [1], finance [2], epidemiology [3], ecology [4], and meteorology [5]. The behaviour of these systems can be described mathematically by state-space models (SSMs). They represent the system with a sequence of state vectors, that are associated with a sequence of noisy observation vectors. Classical filtering techniques aim at computing the posterior distribution of the state given the previous observations, providing not only point estimates of the state but also quantifying their uncertainty. Among these techniques, the Kalman filter (KF) [6] stands out, since it provides the optimal solution for linear Gaussian systems. Several extensions and generalisations of the KF, e.g., the extended Kalman filter (EKF) [7] and the unscented Kalman filter (UKF) [8], have been proposed to deal with non-linear systems. However, these methods approximate the state posterior with a Gaussian, which may not be appropriate. The particle filter (PF) or sequential Monte Carlo (SMC) [9, 10] are alternative solvers for general that approximate distributions using Monte Carlo samples or particles.

The diversity of the particles in the PF is crucial to ensure an adequate representation of the underlying dynamics of the system. This diversity depends directly on the sampling or *proposal* distribution. The simplest approach is to use the system's transition model, as in the bootstrap particle filter (BPF) [9]. However, this approach is not optimal, e.g., it does not incorporate all the observations up to the current time, missing potential information. Several methods aim at approximating the optimal proposal distribution, such as the auxiliary particle filter (APF) [11, 12, 13, 14], which adjusts the particle weights based on both the transition dynamics and the likelihood of the new observation. In the last few years, deep learning methods have been used within the SMC and PF framework to learn the optimal proposal distribution. Some examples are the neural adaptive SMC [15] and the variational SMC [16], which focus on minimising the Kullback-Leibler (KL) from the posterior to the proposal distribution. Furthermore, the advances in differentiable particle filters (DPFs) [17, 18, 19, 20] allow learning parameters and components of the PFs, such as the proposal distribution, using gradient methods.

Contribution. In this paper, we propose PropMixNN, a method to approximate the proposal distribution of a particle filter as an adaptive Gaussian mixture. This approximation allows us to estimate complex distributions, becoming more expressive as the number of mixture components increases. PropMixNN learns the mean and covariance parameters of the components of a multivariate Gaussian mixture as the output of a dense neural network. In order to train this network, we utilise the DPF framework of [17], allowing gradient propagation through the resampling step of the particle filter. In addition, PropMixNN is unsupervised since it targets the estimated log-likelihood, which does not require knowledge of the hidden states, only the sequence of observations and models.

We introduce SSMs and PF background in Section 2. In Section 3 we present our methodology for learning the proposal of a generic non-linear SSM. We provide numerical validation in Section 4, and give closing remarks in Section 5.

B.C. acknowledges support from the *Natural Environment Research Council* of the UK through a SENSE CDT studentship (NE/T00939X/1). S. P.-V. and V. E. acknowledge support from the Agence Nationale de la Recherche of France under PISCES (ANR-17-CE40-0031-01) project. The work of V. E. is also supported by the Leverhulme Research Fellowship (RF-2021-593), and by ARL/ARO under grants W911NF-20-1-0126 and W911NF-22-1-0235. In addition, we acknowledge support from the 2022 University of Edinburgh – Rice University Strategic Collaboration Award.

2. BACKGROUND

2.1. State-space models (SSMs)

We are interested in state-space dynamical systems that can be described by the equations

$$\begin{aligned}\mathbf{x}_t &= f(\mathbf{x}_{t-1},\mathbf{v}_t;\boldsymbol{\theta}),\\ \mathbf{y}_t &= g(\mathbf{x}_t,\mathbf{r}_t;\boldsymbol{\theta}),\end{aligned} \tag{1}$$

where $t=1,...,T$ denotes discrete time, $\mathbf{x}_t \in \mathbb{R}^{d_x}$ is the state of the system, $\mathbf{y}_t \in \mathbb{R}^{d_y}$ is the observation vector, $\mathbf{v}_t$ and $\mathbf{r}_t$ are distributed as the state and observation noises respectively, $\boldsymbol{\theta}$ is a set of parameters, and the functions f and g represent the behaviour of the system. In terms of probability density functions (pdfs), $p(\mathbf{x}_0|\boldsymbol{\theta})$ is the prior distribution of the state, $p(\mathbf{x}_t|\mathbf{x}_{t-1};\boldsymbol{\theta})$ is the conditional density of the state $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$, and $p(\mathbf{y}_t|\mathbf{x}_t;\boldsymbol{\theta})$ is the conditional pdf of the observation $\mathbf{y}_t$ given the hidden state $\mathbf{x}_t$. The initial value of the state is distributed as $\mathbf{x}_0 \sim p(\mathbf{x}_0|\boldsymbol{\theta})$. Note that, the state sequence $\{\mathbf{x}_t\}_{t\geq 0}$ is hidden and therefore not observed, while the related sequence of $\{\mathbf{y}_t\}_{t\geq 1}$ is observed.

2.2. Particle filtering

Filtering methods aim at estimating the state of the systems described in Eqs. (1) by approximating the posterior pdf of the state $\mathbf{x}_t$ given all the observations $\mathbf{y}_{1:t}$, i.e., $p(\mathbf{x}_t|\mathbf{y}_{1:t};\boldsymbol{\theta})$. PFs approximate this pdf using a set of K Monte Carlo samples (particles) and their associated weights, $\{\mathbf{x}_t^{(i)},\tilde{w}_t^{(i)}\}_{i=1}^K$. Thus, the posterior pdf can be written as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t};\boldsymbol{\theta}) \approx \sum_{i=1}^{K} \tilde{w}_t^{(i)} \delta_{\mathbf{x}_t^{(i)}}. \tag{2}$$

A commonly used algorithm for particle filtering is the sequential importance resampling (SIR) algorithm, given in Alg. 1. At every time step t, the particles and weights, $\{\mathbf{x}_t^{(i)},\tilde{w}_t^{(i)}\}_{i=1}^K$, are computed. First, K particles, $\mathbf{x}_t^{(i)}$, are drawn from the proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$ (line 4). With a new observation $\mathbf{y}_t$, we compute their weights, $w_t^{(i)}$, in line 5. After obtaining the normalised weights, $\tilde{w}_t^{(i)}$, we can perform the resampling step (line 7), which samples the particle set according to their relative probability masses. This step is vital to avoid the degeneracy of the filter, i.e., to ensure diversity in the particle set and obtain more accurate approximations of the posterior distribution, $p(\mathbf{x}_t|\mathbf{y}_{1:t};\boldsymbol{\theta})$.

The selection of the proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$ is crucial to ensure the particles are drawn in regions of the state space where the probability is high, so as to improve the efficiency and accuracy of the PF. In the BPF [9], the proposal distribution is set equal to the transition model of the SSM, $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}) = p(\mathbf{x}_t|\mathbf{x}_{t-1};\boldsymbol{\theta})$. This approach is intuitive but omits significant observational information from $\mathbf{y}_t$. The optimal proposal distribution incorporatesthe observation at the current time step t, i.e., $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}) = p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$. However, this distribution is typically intractable or unknown. [21, 22] proposed different parametric neural network architectures to learn the proposal distribution $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$, such as modelling it as a Gaussian distribution or utilising normalising flows.

Algorithm 1 Sequential importance resampling (SIR)

1: Draw $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0|\boldsymbol{\theta})$, for $i=1,...,K$.
2: Set $\tilde{w}_0^{(i)} = 1/K$, for $i=1,...,K$.
3: **for** $t=1,...,T$ and $i=1,...,K$ **do**
4: Draw $\mathbf{x}_t^{(i)} \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$.
5: Compute $w_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)};\boldsymbol{\theta})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)};\boldsymbol{\theta})}{\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})}$.
6: Compute $\tilde{w}_t^{(i)} = \tilde{w}_{t-1}^{(i)} w_t^{(i)} / \sum_i \tilde{w}_{t-1}^{(i)} w_t^{(i)}$.
7: Perform resampling over $\mathbf{x}_t^{(i)}$ with weights $\tilde{w}_t^{(i)}$.
8: **end for**

3. PROPOSED ALGORITHM

3.1. Mixture proposal

In this section we describe the PropMixNN method, which learns the proposal distribution for a particle filter using a deep neural network. PropMixNN conditions the proposal distribution on the previous state value and the current measurement, as is the case for the intractable optimal proposal distribution. We parameterise $\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$ as an equally weighted mixture of S multivariate Gaussian distributions with diagonal covariances, with the S-th component given by

$$\pi_s(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}) = \mathcal{N}(\mu_s(\mathbf{x}_{t-1},\mathbf{y}_t),\Sigma_s(\mathbf{x}_{t-1},\mathbf{y}_t)), \tag{3}$$

with $s=1,...,S$, where $\boldsymbol{\theta}$ are the parameters of the neural network and $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The overall distribution is therefore

$$\pi(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}) := S^{-1} \sum_{s=1}^{S} \pi_s(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}). \tag{4}$$

Mixtures of multivariate Gaussians can represent a wide range of possible distributions, offering flexibility and expressiveness that is beneficial for modelling complex systems [23].

3.2. Network architecture and learning

We propose to learn the $\mu_i(\cdot)$ and $\Sigma_i(\cdot)$ functions using a single dense neural network $\mathrm{NN}(\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$. This network has the same parameters $\boldsymbol{\theta}$ for all t, simplifying learning. We use the previous particle value $\mathbf{x}_{t-1}$ and the observation $\mathbf{y}_t$ as input, as these are also inputs to the (intractable) optimal proposal [24].

The network $\mathrm{NN}(\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta})$ is made up of L layers, where

$$\mathrm{NN}(\mathbf{x}_{t-1},\mathbf{y}_t;\boldsymbol{\theta}) = \mathbf{z}_L, \qquad \mathbf{z}_l = \rho_l(\mathbf{A}_l \mathbf{z}_{l-1} + \mathbf{b}_l), \tag{5}$$

for $l = 1, \dots, L$. This gives $\boldsymbol{\theta} = \{\mathbf{A}_1, \mathbf{b}_1, \dots, \mathbf{A}_L, \mathbf{b}_L\}$ as our learned parameters. Each layer applies a transformation by left-multiplying the d_{l-1}-dimensional vector $\mathbf{z}_{l-1}$ by the $d_l \times d_{l-1}$ matrix $\mathbf{A}_l$ and then summing the result with the d_l-dimensional vector $\mathbf{b}_l$, followed by applying the activation function ρ_l. The initial value $\mathbf{z}_0 = [\mathbf{x}_{t-1}^\intercal, \mathbf{y}_t^\intercal]^\intercal$, is the concatenation of the previous state $\mathbf{x}_{t-1}$ and the current observation $\mathbf{y}_t$. Thus, $\mathbf{z}_0$ has dimension $d_0 = d_x + d_y$. The dimension of

$\mathbf{z}_L = [\boldsymbol{\mu}^{(1)\intercal}, \mathbf{c}^{(1)\intercal}, ..., \boldsymbol{\mu}^{(S)\intercal}, \mathbf{c}^{(S)\intercal}]^\intercal$ is $d_L := 2Sd_x$, as for each of the S components we have a d_x-dimensional mean vector $\boldsymbol{\mu}_t^{(n)}$ and a d_x-dimensional covariance scale vector $\mathbf{c}_t^{(n)}$. We construct the proposal distribution from $\mathrm{NN}(\cdot;\boldsymbol{\theta})$ by

$$\mathrm{makedist}(\mathrm{NN}(\cdot;\boldsymbol{\theta})) = \sum_{s=1}^{S} S^{-1} \mathcal{N}(\boldsymbol{\mu}^{(s)}, \mathrm{diag}(\mathbf{c}^{(s)})^2). \quad (6)$$

We learn the values of the $\mathbf{A}_l$ and $\mathbf{b}_l$ parameters through optimisation, with the hyperparameters ρ_l, L, and d_l fixed by design.

To train the network we maximise the log-likelihood, given by

$$\ell(\boldsymbol{\theta}|\mathbf{y}_{1:T}) = \sum_{t=1}^{T} \left(\sum_{k=1}^{K} \log\left(w_t^{(k)} \cdot \tilde{w}_{t-1}^{(k)} \right) \right), \quad (7)$$

where $w_t^{(k)}$ and $\tilde{w}_{t-1}^{(k)}$ are the unnormalised and normalised weights of the PF in Alg. 1 [24, Chapter 12]. Note that the weights are dependent on $\boldsymbol{\theta}$ through their computation in Alg. 1. The log-likelihood is maximised when all weights are equal to $1/K$, which is a desirable outcome as it reduces weight degeneracy over time [24]. Furthermore, maximising the log-likelihood does not require knowledge of the true value of the hidden state, which is often unavailable, needing only the observations.

3.3. PropMixNN algorithm

The proposed PropMixNN method is described in Alg. 2. The parameters of the NN are initialised (line 2). Then, it proceeds in B iterations, each of them processing an increasingly large batch of observations, i.e., $\mathbf{y}^{(b)} = \mathbf{y}_{1:\lceil bT/B \rceil}$, for $b = 1, ..., B$. We do this to avoid numerical errors, since the first sampled trajectories often have an extremely small log-likelihood, which compounds numerical errors when computing the weights in Alg. 1 [25].

For each batch b, we perform J optimisation steps. For the j-th optimisation step of batch b, we run a particle filter with proposal distribution $\pi = \pi^{(b,j-1)}$ and observations $\mathbf{y}^{(b)} := \mathbf{y}_{1:\lceil bT/B \rceil}$ (line 8). The PF provides a log-likelihood estimate, $\ell\left(\boldsymbol{\theta}_{b,j-1}|\mathbf{y}^{(b)}\right)$, of the parameter $\boldsymbol{\theta}_{b,j-1}$, and compute the gradient $\nabla\ell\left(\boldsymbol{\theta}_{b,j-1}|\mathbf{y}^{(b)}\right)$ with respect to $\boldsymbol{\theta}_{b,j-1}$ (line 9). We then update the parameters $\boldsymbol{\theta}_{b,j-1}$ to obtain $\boldsymbol{\theta}_{b,j}$ (line 10), e.g., using ADAM [26] or RADAM [27]. From the updated parameters, we construct the updated network $\mathrm{NN}(\cdot;\boldsymbol{\theta}_{b,j})$, which outputs the parameters of the distribution $\pi^{(b,j)}$, when imputing $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$ (line 11). At the final iteration of the last batch, we return the fully adapted proposal distribution $\pi^{(B,J)}$ (line 14). In order to estimate quantities of the hidden state, we run the PF (Alg. 1) with $\pi^{(B,J)}$ as the proposal distribution.

3.4. Discussion

Mixtures of multivariate Gaussians are capable of representing unknown distributions, and often do so more reliably than alternative methods such as normalizing flows [21, 22, 28, 29]. These methods, while powerful, are less intuitive than Gaussian mixtures and are more susceptible to overfitting in this context [22, 21]. We note that the weights w_t depend on the samples $\mathbf{x}_{0:t}$, which are drawn from the proposal distribution that we are learning. Therefore, we require a way to propagate gradients through stochastic processes, namely particle resampling and sampling the mixture distribution. We use the resampling method of [17] to propagate gradients through the resampling step of the particle filter. By including resampling in the training using this method, we remove bias in our gradients [17, 18], improving the convergence characteristics of PropMixNN and allowing application to more complex systems, as weight degeneracy is combated in training.

In order to sample from the mixture distribution π, we draw from a categorical distribution to select the mixture component, and then from a multivariate Gaussian distribution. We reparameterise the categorical distribution using a Gumbel-Softmax distribution [30], which allows gradient propagation though categorical sampling. We propagate gradients through the multivariate Gaussian sampling using the reparametrisation trick [31]. Hence, we can compute the gradient of Eq. (7) with respect to our parameters $\boldsymbol{\theta} = \{\mathbf{A}_1, \mathbf{b}_1, ..., \mathbf{A}_L, \mathbf{b}_L\}$, and therefore train the proposal model using gradient methods, such as [26, 27].

Algorithm 2 PropMixNN

1: Choose number of batches B and steps per break J.
2: Initialise $\boldsymbol{\theta}_{0,J} := \{\mathbf{A}_1, \mathbf{b}_1, ..., \mathbf{A}_L, \mathbf{b}_L\}$.
3: Initialise $\pi^{(0,J)} := \mathrm{makedist}(\mathrm{NN}(\cdot;\boldsymbol{\theta}_{0,J}))$.
4: Set $\mathbf{y}^{(b)} := \mathbf{y}_{1:\lceil bT/B \rceil}$ for $b = 1, ..., B$.
5: **for** $b = 1, ..., B$ **do**
6: Set $\boldsymbol{\theta}_{b,0} := \boldsymbol{\theta}_{b-1,J}$ and $\pi^{(b,0)} := \pi^{(b-1,J)}$
7: **for** $j = 1, ..., J$ **do**
8: Run Alg. 1 with $\pi := \pi^{(b,j-1)}$ and observations $\mathbf{y}^{(b)}$.
9: Obtain ℓ and $\nabla\ell$ from above using Eq. (7).
10: Set $\boldsymbol{\theta}_{b,j} := \mathrm{update}(\boldsymbol{\theta}_{b,j-1}, \nabla\ell)$.
11: Set $\pi^{(b,j)} := \mathrm{makedist}(\mathrm{NN}(\cdot;\boldsymbol{\theta}_{b,j}))$.
12: **end for**
13: **end for**
14: **return** $\pi^{(B,J)}$.

4. NUMERICAL EXPERIMENTS

4.1. Lorenz 96 model

We consider a stochastic version of the Lorenz 96 model [32], a system of differential equations known to exhibit chaotic behaviour. The deterministic version of the system is given by

$$\frac{\mathrm{d}x_{i,t}}{\mathrm{d}t} = x_{i-1,t}(x_{i+1,t} - x_{i-2,t}) - x_{i,t} + F, \quad (8)$$

for $i = 1, ..., d_x$, where we define $x_{-1} := x_{d_x-1}, x_0 := x_{d_x}$, and $x_{d_x+1} := x_1$. The variable F is a forcing constant; we use $F = 8$.

We use a forward Euler integrator to approximately solve this system. We denote k iterations of the integrator by I_k,

$$I_1(x_{i,t}, \Delta t) := x_{i,t} + \Delta t \frac{\mathrm{d}x_{i,t}}{\mathrm{d}t}, \quad (9)$$

$$I_j(x_{i,t}, \Delta t) := I_1(I_{j-1}(x_{i,t}, \Delta t), \Delta t), \quad (10)$$

where Δt is the integration timestep. The dynamical system that we use for testing is given by

$$\begin{aligned} x_{i,t+1} &= I_j(x_{i,t}, \Delta t) + v_{i,t+1}, \\ y_{i,t+1} &= x_{i,t+1} + r_{i,t+1}, \end{aligned} \tag{11}$$

for $i = 1, \dots, d_x$ and $t = 0, \dots, T$, where $v_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)$, $r_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_r)$, and I_j is defined by Eq. (10).

4.2. Simulation setup

We set $j = 5$ and $\Delta t = 0.001$ in eq (11). By default, the dimension of the system is $d_x = d_y = 20$, and we set $\mathbf{\Sigma}_v = 0.25\boldsymbol{I}_{20}$ and $\mathbf{\Sigma}_r = 0.1\boldsymbol{I}_{20}$. Every simulation runs until $T = 100$. We initialise the hidden state at $\mathbf{x}_0 = \mathbf{0}_{20}$. We assess the proposed method in terms of relative mean square error (MSE), showing the accuracy of the method as a fraction of the MSE obtained with the BPF [9]. We compare our method with the IAPF [14], to illustrate the performance of a standard improved proposal. The MSE compares the weighted mean of the samples (the estimated state) with the true underlying hidden state. Note that the MSE is not targeted for optimisation. We compute the MSE for 200 independent runs of the filter, and plot the mean and symmetric 95% intervals.

We test the proposed method using a variable number of mixture components, with $S \in \{1, 6, 10\}$. All variants utilise the same network architecture, with 3 layers of output sizes $d_1 = 128, d_2 = 256, d_3 = 2Sd_x$. For the activation function ρ_l in eq. (5) we set $\rho_{1:2}(x) = \text{relu}(x) = \max(0, x)$, and $\rho_3(x) = x$. We train PropMixNN using the Rectified Adam optimiser [27], using a fixed learning rate of $3 \cdot 10^{-3}$, and setting the parameters of Alg. 2 to $B = \lceil T/5 \rceil, J = 50$. We train the method using a series of observations distinct from those on which we test PropMixNN; however, all series are instances of the Lorenz 96 system.

4.3. Numerical results

Variable number of particles. We test PropMixNN for a variable number of particles K, with $K \in \{30, 50, 100, 200\}$. We use a fixed series length $T = 100$. Fig. 1a shows that PropMixNN outperforms the BPF for all given values of K, obtaining at most 0.8 times the MSE of the BPF. PropMixNN with $S = 10$ components suffers with few particles, as few samples are taken from each component, making training less reliable. Our method outperforms the IAPF at all tested numbers of particles, by an increasing large margin as the number of particles increases.

Variable series length. We also test PropMixNN with a variable series length T, with $T \in \{30, 60, 100, 200, 500\}$. In this case we fix the number of particles $K = 100$. We show in Fig. 1b that the proposed method obtains lower values of MSE than the BPF and IAPF for all given values of T. The $S = 10$ component method slightly outperforms the $S = 6$ component method, which significantly outperforms the $S = 1$ component method.

Variable state noise. Finally, we test PropMixNN for a variable state noise $\mathbf{\Sigma}_v = \sigma_v^2 \boldsymbol{I}_{20}$, with $\sigma_v^2 \in \{0.05, 0.1, 0.25, 0.5, 1\}$. In this case, we fix the number of particles $K = 100$ and the series length $T = 100$. Fig. 1c shows that PropMixNN is superior to the BPF for all given values of σ_v^2. The improvement in accuracy is lesser for small noise variances, as small perturbations give a concentrated distribution of the state values at the next time step. However, the performance improves for larger values of σ_v^2. This is due to the chaotic behaviour of the system, which leads to multimodal distributions for the next state, as the state follows one of several diverging paths. The use of the multiple-component mixture in the proposed method captures this behaviour, outperforming both the BPF and the IAPF.

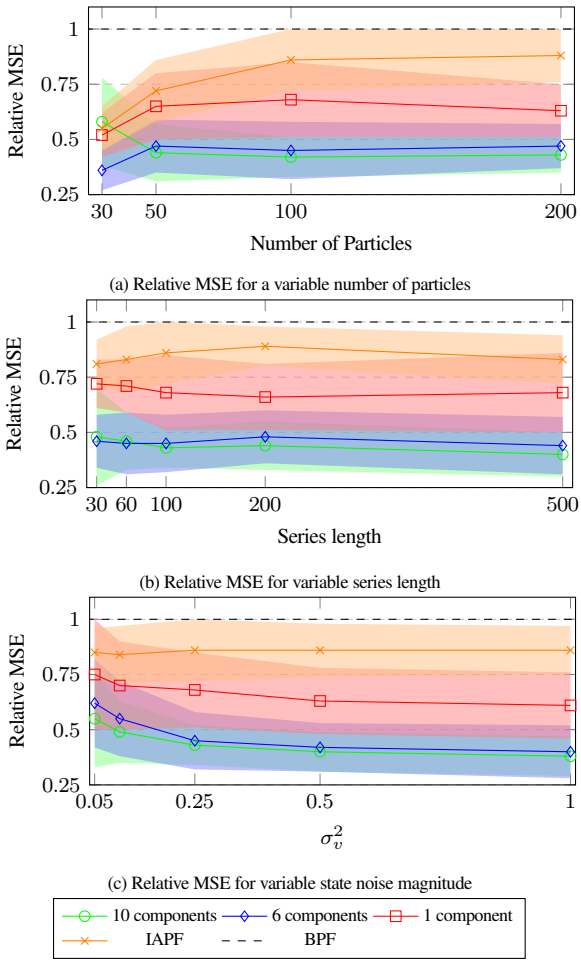

Fig. 1: Comparison of PropMixNN with the BPF and IAPF. Quantities are divided by the MSE of the corresponding BPF.

5. CONCLUSION

This work proposes a novel method, called PropMixNN, to learn the proposal distribution of a particle filter (PF). We employ a dense neural network to learn the inputs to a parametric mixture, from which we propose the hidden states. The proposed method targets the log-likelihood and does not require knowledge of the hidden states or particles of the PF. We show some numerical results for a stochastic Lorenz 96 model, which has highly chaotic behaviour. PropMixNN performs better in terms of estimation error, outperforming standard choices of the proposal distribution and a comparable state-of-the-art method.

6. REFERENCES

[1] X. Wang, T. Li, S. Sun, and J. M. Corchado, "A survey of recent advances in particle filters and remaining challenges for multitarget tracking," *Sensors*, vol. 17, no. 12, pp. 2707, 2017.

[2] A. Virbickaitė, H. F. Lopes, M. C. Ausín, and P. Galeano, "Particle learning for Bayesian semi-parametric stochastic volatility model," *Econometric Reviews*, 2019.

[3] T. A. Patterson, A. Parton, R. Langrock, P. G. Blackwell, L. Thomas, and R. King, "Statistical modelling of individual animal movement: an overview of key methods and a discussion of practical challenges," *AStA Advances in Statistical Analysis*, vol. 101, pp. 399–438, 2017.

[4] K. Newman, R. King, V. Elvira, P. de Valpine, R. S. McCrea, and B. J. Morgan, "State-space models for ecological time-series data: Practical model-fitting," *Methods in Ecology and Evolution*, vol. 14, no. 1, pp. 26–42, 2023.

[5] A. M. Clayton, A. C. Lorenc, and D. M. Barker, "Operational implementation of a hybrid ensemble/4d-Var global data assimilation system at the Met Office," *Quarterly Journal of the Royal Meteorological Society*, vol. 139, no. 675, pp. 1445–1461, 2013.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[7] M. I. Ribeiro, "Kalman and extended Kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, no. 46, pp. 3736–3741, 2004.

[8] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE, 2000, pp. 153–158.

[9] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation," *IEE Proceedings-F Radar and Signal Processing*, vol. 140, pp. 107–113, 1993.

[10] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE signal processing magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[11] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American statistical association*, vol. 94, no. 446, pp. 590–599, 1999.

[12] V. Elvira, L. Martino, M. F. Bugallo, and P. M. Djuric, "Elucidating the auxiliary particle filter via multiple importance sampling [lecture notes]," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 145–152, 2019.

[13] N. Branchini and V. Elvira, "Optimized auxiliary particle filters: adapting mixture proposals via convex optimization," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 1289–1299.

[14] V. Elvira, L. Martino, M. F. Bugallo, and P. M. Djurić, "In search for improved auxiliary particle filters," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 1637–1641.

[15] S. S. Gu, Z. Ghahramani, and R. E. Turner, "Neural adaptive sequential Monte Carlo," *Advances in neural information processing systems*, vol. 28, 2015.

[16] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential Monte Carlo," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 968–977.

[17] A. Corenflos, J. Thornton, A. Doucet, and G. Deligiannidis, "Differentiable particle filtering via entropy-regularized optimal transport," *arXiv preprint arXiv:2102.07850*, 2021.

[18] X. Chen and Y. Li, "An overview of differentiable particle filters for data-adaptive sequential Bayesian inference," *arXiv preprint arXiv:2302.09639*, 2023.

[19] A. Ścibior and F. Wood, "Differentiable particle filtering without modifying the forward pass," *arXiv preprint arXiv:2106.10314*, 2021.

[20] W. Li, X. Chen, W. Wang, V. Elvira, and Y. Li, "Differentiable bootstrap particle filters for regime-switching models," *arXiv preprint arXiv:2302.10319*, 2023.

[21] F. Gama, N. Zilberstein, R. G. Baraniuk, and S. Segarra, "Unrolling particles: Unsupervised learning of sampling distributions," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 5498–5502.

[22] F. Gama, N. Zilberstein, M. Sevilla, R. Baraniuk, and S. Segarra, "Unsupervised learning of sampling distributions for particle filters," *arXiv preprint arXiv:2302.01174*, 2023.

[23] M. F. Bugallo, V. Elvira, L. Martino, D. Luengo, J. Miguez, and P. M. Djuric, "Adaptive importance sampling: The past, the present, and the future," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 60–79, 2017.

[24] S. Särkkä, *Bayesian filtering and smoothing*, Cambridge University Press, 1st edition, 2013.

[25] A. Doucet, A. M. Johansen, et al., "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, pp. 3, 2009.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[27] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[28] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[29] B. L. Trippe and R. E. Turner, "Conditional density estimation with bayesian normalising flows," *arXiv preprint arXiv:1802.04908*, 2018.

[30] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[32] E. N. Lorenz, "Predictability: A problem partly solved," in *Proc. Seminar on predictability*. ECMWF, 1996.