NEURAL NETWORKS SEQUENTIAL TRAINING USING VARIATIONAL GAUSSIAN PARTICLE FILTER

Mhd Modar Halimeh, Andreas Brendel, and Walter Kellermann

Multimedia Communications and Signal Processing
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstr. 7, D-91058 Erlangen, Germany.
{mhd.m.halimeh, andreas.brendel, walter.kellermann}@fau.de

ABSTRACT

In this paper, we propose a sequential training algorithm for feedforward neural networks based on particle filtering. The proposed algorithm uses variational learning to tailor a proposal density by minimizing the variational energy. This density is then incorporated into the Gaussian particle filter framework. The proposed algorithm and an extension to it using evolutionary resampling are compared to training a neural network using a random walk-based particle filter, an extended Kalman filter, the use of variational learning only, and the backpropagation algorithm, using a synthetic dataset generated by a time-varying random process and a real dataset, where the proposed approach resulted in a moderately lower training and testing errors and a better convergence behavior, rendering the algorithm attractive for uses such as neural networks pre-training.

1. INTRODUCTION

Given a neural network architecture for a specific task, the main challenge is to train the network by an efficient training algorithm making the best use of the available data. Due to its simplicity and computational efficiency, the backpropagation algorithm (BPA) [1], including its extensions, e.g., [2–5], stands out as the most-widely used supervised training algorithm. However, and despite its widespread use, several challenges to the BPA have been identified and extensively studied. These challenges include the vulnerability to making overconfident decisions, i.e., overfitting, by not considering any uncertainty, or equivalently confidence, measures in the network parameters estimates [6]. Moreover, significant effort is required for the parametrization of the BPA itself before the actual training [7].

A Bayesian approach to train neural networks aims at overcoming some of the challenges faced by the BPA [8] or similar gradient descent-based algorithms. Many Bayesian approaches for neural network training have been developed in the last decades, one of the earliest being the use of the Extended Kalman Filter (EKF) to train Feedforward Neural Networks (FNN) [9] while in [10], particle filtering was used to realize a hierarchal Bayesian framework for FNNs. A Hamilton dynamics-based Markov Chain Monte-Carlo method was developed in [11]. In [12], a Bayesian framework for NN was introduced using variational methods. Building upon [12], the probabilistic BPA was introduced in [7], while [13] introduced the 'Bayes by BP', also based on variational learning. In [14], adaptive importance sampling is proposed to accelerate the training of a neural probabilistic language model.

The authors would like to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting this work (contract number KE 890/4-2).

In this paper, a Bayesian approach for training FNNs sequentially is proposed. This approach is beneficial in environments where the data should be incorporated into the training while it arrives as a data stream along the time axis. To this end, we introduce a particle filter-based algorithm, where variational learning is used to tailor an importance density, from which we generate particles. This density is then incorporated into the Gaussian Particle Filter (GPF) [15] scheme as it will be used to train the FNN. The algorithm is evaluated using two datasets, a synthetic dataset generated by a time-varying random process and the real-world ABALONE dataset [16].

This paper is structured as follows: in Section 2 an overview of particles filters and the unique problem of static parameters estimation for algorithms such as particle filters is given. Section 3 introduces the proposed approach, while in Section 4 the proposed algorithm is verified by two generic experiments. Finally, in Section 5, conclusions are drawn and an outlook to future work is presented.

2. OVERVIEW

In this Section, we first give a brief overview of particle filtering (see also [17] and the references therein for recent advances in Monte Carlo methods). Then, we discuss the problem of estimating static parameters using particle filters and the challenges that are associated with this case.

2.1. Particle Filtering

Denoting the input vector measurements, e.g., features, at time instant k by $\boldsymbol{x}_k$, the output measurements, e.g., class association probabilities, by $\boldsymbol{y}_k$, and the latent random variable, i.e., the network's weights, by $\boldsymbol{w}_k = [w_{1,k}, w_{2,k}, ..., w_{N_w,k}]^{\mathrm{T}}$, where N_w denotes the number of weights in the network, a state-space model reads

$$\boldsymbol{w}_k = \boldsymbol{f}(\boldsymbol{w}_{k-1}, \boldsymbol{d}_k), \tag{1}$$

$$\boldsymbol{y}_k = \boldsymbol{g}(\boldsymbol{w}_k, \boldsymbol{x}_k) + \boldsymbol{v}_k, \tag{2}$$

where $\boldsymbol{d}_k$ and $\boldsymbol{v}_k$ are the process and measurements noise, respectively, and are assumed to be independent, white, Gaussian, and of zero-mean. $\boldsymbol{g}(\cdot)$ is the function describing the network and thus is known, while the function $\boldsymbol{f}(\cdot)$ describes the evolution of the weights vector over time. A Bayesian approach aims at obtaining the posterior density $p(\boldsymbol{w}_k|\boldsymbol{y}_{1:k})$, which encapsulates the knowledge about the weights vector $\boldsymbol{w}$ at time instant k, given all the observed output measurements $\boldsymbol{y}$ until k. The according Bayes filter reads [18]

$$p(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k}) = \frac{p(\boldsymbol{y}_k|\boldsymbol{w}_k)p(\boldsymbol{w}_k|\boldsymbol{w}_{k-1})}{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})} p(\boldsymbol{w}_{0:k-1}|\boldsymbol{y}_{1:k-1}). \tag{3}$$

When either function $\boldsymbol{f}(\cdot)$ or $\boldsymbol{g}(\cdot)$ is nonlinear, the Bayes filter becomes analytically intractable and an approximation is necessary [19]. A popular approximation technique is particle filtering, where the posterior density $p(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$ is approximated using a set of N_p particles $\{\boldsymbol{w}_{0:k}^{(i)}\}_{i=1}^{N_p}$ and their associated weights $\{q_k^{(i)}\}_{i=1}^{N_p}$, according to (see [20] for a detailed treatment of particle filters)

$$p(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{N_p} q_k^{(i)} \delta(\boldsymbol{w}_{0:k} - \boldsymbol{w}_{0:k}^{(i)}). \tag{4}$$

The particles $\{\boldsymbol{w}_{0:k}^{(i)}\}_{i=1}^{N_p}$ are drawn from an easily accessible density known as the importance density, denoted by $\pi(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$ which has a strict support over $p(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$ [21].

Since the particles are drawn from a different density, i.e., $\pi(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$ instead of $p(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$, they are weighted by

$$q_k^{(i)} = \frac{p(\boldsymbol{w}_{0:k}^{(i)}|\boldsymbol{y}_{1:k})}{\pi(\boldsymbol{w}_{0:k}^{(i)}|\boldsymbol{y}_{1:k})}. \tag{5}$$

Finally, and in order for (4) to represent a density, the weights are normalized such that $\sum_{i=1}^{Np} q_k^{(i)} = 1$ and are resampled using one of many proposed resampling schemes in the literature [22].

Despite the mild formal assumptions the density $\pi(\boldsymbol{w}_{0:k}|\boldsymbol{y}_{1:k})$ has to fulfill in order to be a valid importance density, the choice of this density is crucial and a bad choice can lead to poor performance. In [23], it has been shown that the optimal choice of $\pi(\boldsymbol{w}_k|\boldsymbol{y}_{1:k}, \boldsymbol{w}_{0:k-1})$, w.r.t. the particles weights' variance, is $p(\boldsymbol{w}_k|\boldsymbol{w}_{k-1}, \boldsymbol{y}_k)$. Unfortunately, this density is often inaccessible, and a common practical choice is simply to sample from the state transition density $p(\boldsymbol{w}_k|\boldsymbol{w}_{k-1})$, which is known as prior sampling.

2.2. Static Parameters Problem

The static nature of the parameters of interest, i.e., the time-invariance of the optimum weights vector in the case of neural networks, is problematic and imposes serious challenges to on-line Bayesian inference [24, 25], e.g., particle filters. Limiting the scope of the discussion to particle filters, a common way to address this problem is the use of *artificial* state transition models with an intuitive choice being the random walk. Despite being widely-used, random walk is a highly inefficient way of moving particles in the state-space. This inefficiency becomes more severe when the latent variable is of high dimensionality. As an alternative to the random walk, one can incorporate other estimation algorithms to update/move the particles, such as the EKF [10], the Alopex algorithm [26], or the Unscented Kalman Filter (UKF) [27].

3. THE PROPOSED METHOD

In this Section, our proposed approach for devising an importance density based on minimizing the variational energy is introduced and followed by an inference stage to weight and resample the drawn particles.

3.1. A Variational Learning-based Importance Density

In this paper, we propose the use of variational inference [19] to address the static nature of the FNN weights vector $\boldsymbol{w}_k$. The reason behind this choice is that variational learning enables us to obtain an importance density that is close to the posterior which, in the context of particle filtering, is desirable, since such importance densities result in particles with a low weights' variance, as it can be seen from (5).

Since our aim is to devise an importance density $\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)$ that is as close to the true underlying posterior as possible, we minimize the variational free energy $\mathcal{F}(\cdot)$ w.r.t. the parameters vector $\boldsymbol{\theta}_k$, which completely characterizes $\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)$ [12]

$$\mathcal{F}(\boldsymbol{\theta}_k, \boldsymbol{y}_k) = -\left\langle \ln\left[\frac{p(\boldsymbol{y}_k|\boldsymbol{w}_k)p(\boldsymbol{w}_k|\alpha)}{\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)}\right]\right\rangle_{\boldsymbol{w}_k\sim\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)}, \tag{6}$$

where $p(\boldsymbol{w}_k|\alpha)$ is the weights prior density, that is characterized by the parameter α, while $\langle g(x)\rangle_{x\sim p}$ denotes the expectation of $g(x)$ over the density p. The aim of this paper is to properly incorporate (6) into a particle filter-based framework that is used afterwards for FNN training.

Following [12], the variational free energy in (6) can be decomposed into two parts, with the first part describing the network loss, which is related to the weights likelihood

$$L^N(\boldsymbol{\theta}_k, \boldsymbol{y}_k) = -\left\langle \ln p(\boldsymbol{y}_k|\boldsymbol{w}_k)\right\rangle_{\boldsymbol{w}_k\sim\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)}, \tag{7}$$

while the second part describes the complexity loss, which quantifies the deviation of $\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)$ from the prior

$$L^C(\boldsymbol{\theta}_k, \boldsymbol{y}_k) = -\left\langle \ln\left[\frac{p(\boldsymbol{w}_k|\alpha)}{\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)}\right]\right\rangle_{\boldsymbol{w}_k\sim\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)}. \tag{8}$$

In the following, we assume that the density $\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)$ is normally distributed, with the parameter set $\boldsymbol{\theta}_k$ consisting of the mean vector and covariance matrix $\boldsymbol{\theta}_k = \{\overline{\boldsymbol{w}}_{k|k-1}, \boldsymbol{C}_{k|k-1}\}$. Obviously, Gaussianity of the posterior is a strong assumption and does not match the mutlimodality associated with densities of high dimensions that describe highly nonlinear systems. However, this assumption is motivated by the fact that using multimodal densities would result in an unrealistically high computational load rendering the training algorithm effectively unusable.

Unlike [12], we do not restrict the covariance matrix to be diagonal. This is motivated by the fact that when separable distributions, i.e., distributions representable as a product of the marginals, are used, a good approximative importance density given by minimizing the variational free energy is more concentrated than the true distribution. This has led to criticizing the use of variational learning in devising importance densities, since compact distributions are inefficient samplers for particle filters [19].

Next, we introduce the following identity [28]

$$\nabla_{\boldsymbol{\mu}}\langle \boldsymbol{h}(\boldsymbol{z})\rangle_{\boldsymbol{z}\sim\mathcal{N}} = \langle \nabla_{\boldsymbol{z}}\boldsymbol{h}(\boldsymbol{z})\rangle_{\boldsymbol{z}\sim\mathcal{N}}, \tag{9}$$

in which $\mathcal{N}$ denotes a normal distribution with the mean vector $\boldsymbol{\mu}$. To minimize the variational free energy, the mean vector $\overline{\boldsymbol{w}}_{k|k-1}$ is updated according to

$$\overline{\boldsymbol{w}}_{k|k-1} = \overline{\boldsymbol{w}}_{k-1} - \eta\nabla_{\overline{\boldsymbol{w}}}\mathcal{F}(\boldsymbol{\theta}_k, \boldsymbol{y}_k), \tag{10}$$

where η denotes the step size, and $\overline{\boldsymbol{w}}_{k-1}$ is the mean vector of the posterior's $p(\boldsymbol{w}_{0:k-1}|\boldsymbol{y}_{1:k-1})$ approximation $\mathcal{N}(\overline{\boldsymbol{w}}_{k-1}, \boldsymbol{C}_{k-1})$. This corresponds to moving the particles in a direction of a lower variational free energy (6).

In order to evaluate the update term, we employ the identity (9). Let $w_{j,k}$ denote the j-th element in the weights vector $\boldsymbol{w}_k$, and by using the Gaussianity assumption, one arrives at

$$\frac{\partial L^N(\boldsymbol{\theta}_k, \boldsymbol{y}_k)}{\partial \overline{\boldsymbol{w}}_{j,k|k-1}} = \left\langle \frac{\partial \ln p(\boldsymbol{y}_k|\boldsymbol{w}_k)}{\partial w_{j,k}}\right\rangle_{\boldsymbol{w}_k\sim\pi(\boldsymbol{w}_k|\boldsymbol{\theta}_k)} \approx \frac{1}{N_p}\sum_{i=1}^{N_p}\frac{\partial \ln p(\boldsymbol{y}_k|\boldsymbol{w}_k^{(i)})}{\partial w_{j,k}}, \tag{11}$$

where the approximation is due to the use of the necessarily finite number of particles N_p and the term $\frac{\partial \ln p(\boldsymbol{y}_k|\boldsymbol{w}_k^{(i)})}{\partial w_{j,k}}$ can be obtained using the conventional BPA.

Regarding the second component of (6), i.e., the complexity loss (8): one can chose many priors, an obvious choice being $p(\boldsymbol{w}_k|\alpha) = \mathcal{N}(\boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{prior}})$, which results in an update term similar to the L2 regularization [12]. Another possibility is to set the prior to be uniformly distributed, meaning that all weights are equally probable. Then the minimization of $\mathcal{F}(\boldsymbol{\theta}_k, \boldsymbol{y}_k)$ becomes equivalent to minimizing $L^N(\boldsymbol{\theta}_k, \boldsymbol{y}_k)$ combined with Gaussian weight noise (see [12]).

Since estimating $\boldsymbol{C}_{k|k-1}$ using variational learning will generally be computationally costly [28] and constitutes a serious handicap for a neural network training scheme, $\boldsymbol{C}_{k|k-1}$ is set to be equal to $\boldsymbol{C}_{k-1}$, which is calculated at the previous inference stage.

After obtaining the importance density $\mathcal{N}(\overline{\boldsymbol{w}}_{k|k-1}, \boldsymbol{C}_{k|k-1})$, N_p new particles are drawn to replace the old particles, and one moves to the inference stage.

3.2. Inference Stage

After updating the particles, the proposed scheme continues as a conventional GPF [15], by first evaluating the particles weights

$$q_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{w}_k^{(i)}), \tag{12}$$

which are then normalized such that $\sum_{i=1}^{N_p} q_k^{(i)} = 1$.
Afterwards, the posterior's parameters, i.e, the mean vector $\overline{\boldsymbol{w}}_k$ and the covariance matrix, are estimated according to

$$\overline{\boldsymbol{w}}_k = \sum_{i=1}^{N_p} q_k^{(i)} \boldsymbol{w}_k^{(i)}, \tag{13}$$

$$\boldsymbol{C}_k = \sum_{i=1}^{N_p} q_k^{(i)} (\boldsymbol{w}_k^{(i)} - \overline{\boldsymbol{w}}_k)(\boldsymbol{w}_k^{(i)} - \overline{\boldsymbol{w}}_k)^{\mathrm{T}}. \tag{14}$$

The density $\mathcal{N}(\overline{\boldsymbol{w}}_k, \boldsymbol{C}_k)$ is used to draw N_p equally weighted particles [15]. The parameters $\{\overline{\boldsymbol{w}}_k, \boldsymbol{C}_k\}$ are then propagated to the next time step. We refer to this approach as the Variational Gaussian Particle Filter (VGPF) and summarize it in Algorithm 1.

Additionally, an efficient extension is possible by employing the evolutionary resampling concept [29, 30]. Evolutionary resampling performs a partial resampling step instead of resampling the entire particles' population. The particles which are to be resampled are determined based on their weights. We refer to this extension as the Evolutionary VGPF (E-VGPF).

Algorithm 1 The VGPF Pseudocode

Initialize the network with $\boldsymbol{w}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_0)$
for $k = 1 : K$ **do**
  Update the mean vector $\overline{\boldsymbol{w}}_{k|k-1}$ according to (10).
  Propagate the variance $\boldsymbol{C}_{k|k-1} = \boldsymbol{C}_{k-1}$
  Draw N_p new particles $\sim \mathcal{N}(\overline{\boldsymbol{w}}_{k|k-1}, \boldsymbol{C}_{k|k-1})$.
  Evaluate the particles' weights $\{q_k^{(i)}\}_{i=1}^{N_p}$ (12).
  Calculate the mean vector $\overline{\boldsymbol{w}}_k$ and covariance matrix $\boldsymbol{C}_k$ using (13) and (14), respectively.
end for

4. SIMULATIONS

In this section, the VGPF and E-VGPF are evaluated and examined using two different datasets, one generated by a synthesized time-varying random process while the other, the ABALONE dataset [16], is a real dataset.

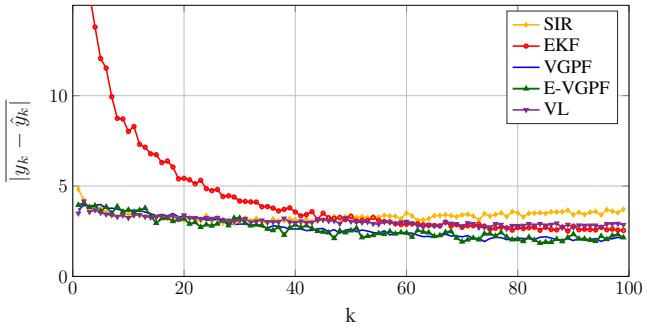

Fig. 1: Section 4.1: Average predicted absolute output error.

4.1. A Dataset Generated by A Time-Varying Process

In the first experiment, we focus on the efficiency of the proposed training scheme for a time-varying scenario resorting to the frequently used time-varying random process from [31]

$$y_k = 4\sin(x_{1,k} - 2) + 2x_{2,k}^2 + 5\cos(0.02k) + 5 + v_k. \tag{15}$$

Here, a total of 100 samples of the output data y_k are generated, where the input vector $\boldsymbol{x}_k = [x_{1,k}, x_{2,k}]^{\mathrm{T}}$ is standard normally distributed, while v_k denotes a zero mean, additive white Gaussian noise with the variance $\sigma_v^2 = 0.1$.

Unlike [31], we approximate the process using a network with a single hidden layer of 20 sigmoid neurons and a linear output layer. The excessive number of neurons in the hidden layer, compared to 5 in [31], aims at illustrating the efficiency of the proposed method in a relatively high-dimensional state-space. Networks trained with the proposed VPGF and E-VGPF are compared to networks trained using a conventional Sequential Importance Sampling/Resampling (SIR) particle filter, and an EKF, both based on the random walk state model, and to a network trained using Variational Learning (VL) [12] based on the Gaussianity assumption.

The SIR particle filter is configured with $N_p = 100$ particles, the random walk was realized by adding a white Gaussian noise with $\mathcal{N}(\boldsymbol{0}, \sigma_d^2 \boldsymbol{I})$ to the weights at each iteration, where $\boldsymbol{I}$ is the identity matrix and $\sigma_d^2 = 0.5$. The EKF is parametrized with a diagonal process noise covariance matrix $\sigma_d^2 \boldsymbol{I}$, with $\sigma_d^2 = 0.01$. The VGPF/E-VGPF algorithms use $N_p = 10$ particles, an observation noise variance for all training algorithms is $\hat{\sigma}_v^2 = 2$. The VGPF, E-VGPF and VL are trained assuming uniform priors and a step size in (10) is set to $\eta = 0.075$. For the VL, 10 samples are used to evaluate the gradient in (10) and the covariance matrix is assumed to be diagonal and is obtained as in [12].

To initialize the different networks, an initial weights vector $\boldsymbol{w}_0$ is drawn from a zero-mean Gaussian distribution with a diagonal covariance matrix variance $\sigma_0^2 \boldsymbol{I}$ with $\sigma_0^2 = 100$ for the weights connecting the input layer to the hidden layer and $\sigma_0^2 = 1$ for the weights connecting the hidden layer to the output layer, similar to [31].

Finally, to evaluate the performance of each training algorithm, we use the prediction Root Mean Square Error (RMSE), meaning that at time instant k, we use the parameters, i.e., the network weights, obtained by the training algorithm to predict the output $y(k+1)$ and compute the RMSE of it. In addition, the Standard Deviation of the Error (STDE) is also calculated. The experiment was repeated 1000 times and the resulting average RMSE and STDE values, in addition to the run times T normalized to the maximum,

	SIR	EKF	VL	VGPF	E-VGPF
RMSE	4.5	7.2	4.2	3.7	3.6
STDE	4.4	6.3	4.0	3.6	3.5
T	1	0.07	0.24	0.30	0.28

Table 1: Section 4.1 RMSE, STDE and normalized run time.

are given in Table 1. Moreover, to provide an insight into the convergence behavior, the RMSE values are averaged over the 1000 realizations and depicted in Figure 1 for the networks trained by the different algorithms. It is clear from the results, in both Table 1 and Figure 1, that networks trained by the VGPF and E-VGPF perform best, while the advantage of training via the SIR particle filter over the EKF is consistent with the results reported in [31]. Furthermore, from the Figure 1, a superior convergence behavior is observed for both the VGPF and E-VGPF, compared to the EKF. It is also obvious that the advantage of the evolutionary resampling is minimal in this scenario.

4.2. ABALONE Dataset

In this experiment, we evaluate the proposed algorithm using the ABALONE dataset provided in [16]. In this dataset, the aim is to predict the age of abalone using different physical measurements of the shell. The measurements include the diameter, length, height, shucked weights, whole weights, viscera weight, gender, and the number of rings. The problem can be formulated as a classification problem or as a regression problem. In this experiment, we opt for the regression problem. The dataset contains 4178 samples, divided into 3000 samples for training and 1178 samples for testing.
The trained FNN has a single hidden layer with 25 neurons, each with a sigmoid activation function, and an output layer with a linear activation function. The different networks are trained using an SIR particle filter, an EKF, a conventional BPA, VL, the proposed VGPF, and E-VGPF.

The SIR particle filter uses a random walk as proposal density realized by adding a white Gaussian noise with $\mathcal{N}(\mathbf{0}, \sigma_d^2\mathbf{I})$ with $\sigma_d^2 = 0.01$ to the particles, while the observation noise variance is set to be $\sigma_v^2 = 0.5$ and has $N_p = 100$ particles. The EKF is parametrized with a process noise variance $\sigma_d^2 = 0.001$ and an observation noise variance $\sigma_v^2 = 0.005$. The VL, VGPF, and E-VGPF have uniform priors. The VL uses a 100 samples to evaluate the gradient in (10), and evaluates a diagonal covariance matrix as in [12].

The observation noise for the VPGF, and E-VGPF is set similar to the EKF with $\sigma_v^2 = 0.005$. The VGPF, E-VGPF, VL, and BP are parametrized with a decaying step size $\eta_k = \frac{0.1}{k}$ and the number of particles the VGPF, and E-VGPF is $N_p = 100$. Finally, to initialize the networks, the initial weights are drawn from a Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma_0^2\mathbf{I})$ with $\sigma_0^2 = 0.5$.

The experiment is repeated 100 times, i.e., the networks are re-initialized and trained a 100 times. To evaluate each algorithm, the RMSE and STDE are calculated for the testing data subset and the average RMSE and STDE, averaged over the 100 repetitions, are presented in Table 2 in addition to the normalized run time T for each algorithm.

As it can be seen from the results in Table 2, the VGPF trained network outperforms both the SIR-trained and the EKF-trained networks by a considerable margin, while moderately outperforming the BPA-trained and the VL-trained networks by a margin of 7%. While the incorporation of the evoluationary resampling in the E-VGPF provides an improvement of another 7% relative to the VGPF.

Next, we examine the convergence behavior of the VGPF/E-VGPF and compare it to the VL and the conventional BPA to determine if the advantage w.r.t. the testing RMSE extends to the convergence behavior. The training RMSE is averaged over the 100 trials, smoothed, by averaging over a 10 samples long window for better visualization, and plotted in Figure 2.

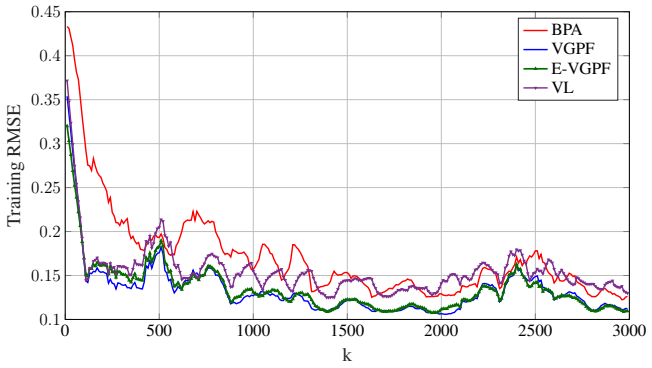

Fig. 2: Convergence behavior comparison in Section 4.2.

	SIR	EKF	BPA	VL	VGPF	E-VGPF
RMSE	0.246	0.291	0.127	0.125	0.116	0.109
STDE	0.214	0.27	0.048	0.045	0.027	0.025
T	0.45	0.12	0.06	0.68	1	0.81

Table 2: Section 4.2, test RMSE, STDE and normalized run time.

Figure 2 shows that the VGPF, E-VGPF, and VL converge significantly faster than the BPA. However, while the BPA and the VL reach a similar training error at the end of the training phase, the VGPF, and E-VGPF maintain their advantage, and provide a lower training error and a more stable behavior. It is also worth noting that, despite similar training errors, the VL generalizes slightly better than the BPA, which can be recognized by recalling the difference in the testing errors in Table 2. However, this desirable behavior of the VGPF comes at the expense of increased computational complexity, see Table 2. This computational load is eased in the E-VGPF, where the resampling is applied partially.

Finally, we observe that for a long sequence of training data, the particles' cloud starts to collapse and the posterior variance diminishes, reflecting path degeneracy and motivating the use of one of the variance introduction techniques, e.g., Markov Chain Monte Carlo moves [32] or regularization. This is in line with other observations reported in the literature regarding Monte Carlo-based approaches for estimating static parameters, e.g., [33].

5. CONCLUSION

In this paper, we have introduced the VGPF, a sequential training algorithm for FNNs. The VGPF uses variational learning to devise an importance density which is later on incorporated into the GPF scheme. The VGPF and an extension using evolutionary resampling, i.e., E-VGPF, are evaluated for two different datasets, a time-varying synthetic dataset and a real dataset. Both algorithms showed a moderately better performance and generalization, in addition to a better convergence behavior. As a consequence, the proposed algorithms are good candidates for use cases such as pre-training of FNN. Moreover, the use of evolutionary resampling resulted in the less computationally intense E-VGPF variant.

Future work will address the path degeneracy observed for long training sequences, a phenomenon that was observed and reported for competing approaches in the literature before.

6. REFERENCES

[1] D. Rumelhart *et al.*, *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1, chapter Learning internal representations by error propagation, pp. 318–362, MIT Press, Cambridge, MA, USA, 1986.

[2] P. Diederik *et al.*, "Adam: A method for stochastic optimization," in *Int. Conf. Learning Representations (ICLR)*, May 2015.

[3] S. Ruder, "An overview of gradient descent optimization algorithms," in *arXiv, 1609.04747*, 2016.

[4] J. Koushik and H. Hayashi, "Improving stochastic gradient descent with feedback," in *arXiv, 1611.01505*, 2016.

[5] F. Shang *et al.*, "Fast stochastic variance reduced gradient method with momentum acceleration for machine learning," in *arXiv, 1703.07948*, 2017.

[6] J. Lampinen and A. Vehtari, "Bayesian approach for neural networks review and case studies," *Neural Networks*, vol. 14, no. 3, pp. 257 – 274, 2001.

[7] J. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *32nd Int. Conf. Machine Learning*, 2015, pp. 1861–1869.

[8] D. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, May 1992.

[9] S. Singhal and L. Wu, *Advances in neural information processing systems 1*, chapter Training multilayer perceptrons with the extended Kalman algorithm, pp. 133–140, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[10] N. de Freitas, *Bayesian methods for neural networks*, Ph.D. thesis, Trinity College, University of Cambridge, 1999.

[11] R. Neal, *Bayesian learning for neural networks*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[12] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor *et al.*, Eds., pp. 2348–2356. Curran Associates, Inc., 2011.

[13] C. Blundell *et al.*, "Weight uncertainty in neural networks," in *Int. Conf. on Machine Learning*, 2015, vol. 37, pp. 1613–1622.

[14] Y. Bengio and J. Senecal, "Adaptive importance sampling to accelerate training of a neural probabilistic language model," *IEEE Trans. on Neural Networks*, vol. 19, no. 4, pp. 713–722, April 2008.

[15] J. Kotecha and P. Djuric, "Gaussian particle filtering," *IEEE Trans. Signal Process.*, vol. 51, no. 10, pp. 2592–2601, Oct. 2003.

[16] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," `http://archive.ics.uci.edu/ml`: retrived on 01.02.2018, 2017.

[17] S. Shao *et al.*, "Bayesian model comparison with the Hyvärinen score: computation and consistency," *arXiv e-prints*, p. arXiv:1711.00136, Oct. 2017.

[18] C. Bishop, *Pattern recognition and machine learning*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] D. MacKay, *Information theory, inference & learning algorithms*, Cambridge University Press, New York, USA, 2002.

[20] O. Cappé, S. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, May 2007.

[21] G. Casella C. Robert, *Monte Carlo statistical methods*, Springer, New York, NY, U.S., 2004.

[22] M. Gerber *et al.*, "Negative association, ordering and convergence of resampling methods," *arXiv e-prints*, p. arXiv:1707.01845, July 2017.

[23] A. Doucet *et al.*, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, July 2000.

[24] N. Kantas *et al.*, "On particle methods for parameter estimation in state-space models," *Statistical Science*, vol. 30, no. 3, pp. 328–351, Aug. 2015.

[25] V. Tadic C. Andrieu, A. Doucet, "On-line parameter estimation in general state-space models," in *IEEE Conf. on Decision and Control*, Dec. 2005, pp. 332–337.

[26] Z. Chen *et al.*, "Theory of Monte Carlo sampling-based Alopex algorithms for neural networks," in *IEEE Int. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, May 2004, vol. 5, pp. 01–04.

[27] R. Van Der Merwe *et al.*, "The unscented particle filter," in *Int. Conf. on Neural Information Process. Sys.*, Feb. 2001, vol. 13, pp. 563–569.

[28] M. Opper and C. Archambeau, "The variational Gaussian approximation revisited," vol. 21, pp. 786–92, Oct. 2008.

[29] C. Huemmer *et al.*, "The significance-aware EPFES to estimate a memoryless preprocessor for nonlinear acoustic echo cancellation," in *IEEE Global Conf. on Signal and Inform. Process. (GlobalSIP)*, Dec. 2014, pp. 557–561.

[30] M. Halimeh *et al.*, "Hybrid particle filtering based on an elitist resampling scheme," in *Sensor Array and Multichannel Signal Process. Workshop (SAM)*, Jul 2018.

[31] N. de Freitas *et al.*, "Sequential Monte Carlo methods to train neural network models," *Neural Computation*, vol. 12, no. 4, pp. 955–993, Apr. 2000.

[32] W. Gilks and C. Berzuini, "Following a moving target Monte Carlo inference for dynamic Bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, Jan. 2002.

[33] C. Andrieu *et al.*, "Sequential MCMC for Bayesian model selection," in *IEEE Signal Process. Workshop on Higher-Order Statistics*, June 1999, pp. 130–134.