A Neural Implementation for Nonlinear Filtering

Anna Kutschireiter
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
annak@ini.uzh.ch

Simone Carlo Surace
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
surace@ini.uzh.ch

Henning Sprekeler
Bernstein Center for Computational Neuroscience
Technische Universitaet Berlin
10587 Berlin, Germany
h.sprekeler@tu-berlin.de

Jean-Pascal Pfister
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
jpfister@ini.uzh.ch

Abstract

The computational task of continuous-time state estimation, nonlinear filtering and identification, i.e. parameter learning, poses a class of interesting problems, which mathematicians have been working on for over 50 years and which has received increasing attention in both machine-learning and neuroscience communities. Moreover, the question how Bayesian inference in general and nonlinear filtering in particular can be implemented in neuronal tissue might be a step towards understanding information processing in the brain. Yet possible answers to this question remain debated. Starting from the mathematical formalism of nonlinear filtering theory, we propose a stochastic rate-based network in terms of a stochastic differential equation whose activity samples the posterior dynamics. The underlying mathematical framework is flexible enough to additionally allow extensions to other tasks such as parameter learning. We show that the numerical performance of the model is adequate to account for both nonlinear filtering and identification problems. Our network may be implemented as a recurrent neuronal network in a biologically plausible manner and thus offers a concrete proposition of how neural sampling might be implemented in the brain.

1 Introduction

One of the most fascinating properties of the brain is its ability to continuously gather and process information about its environment based on noisy and hence unreliable input. There exists ample experimental evidence, e.g. in perception [3] and in decision making [2], that uncertainty is an intrinsic part of these computational processes in the brain. In general, uncertainty is represented in terms of probability distributions and there is accumulating evidence that the brain is constantly performing inference [22, 7]. However, it remains controversial how uncertainty might be represented by neuronal populations. On the one hand, it has been suggested that probability distributions are expressed as probabilistic population codes [18], in which each neuron represents a state of the encoded random variable and their activities are proportional to the probability of the corresponding state. As the name suggests, the information about a single random variable is distributed across a population of neurons. On the other hand, the neural sampling hypothesis [7] proposes an inference scheme where the activity of each neuron represents the full distribution by presenting samples from it over time.

From the mathematical point of view, the algorithmic task of continuous-time inference, commonly referred to as nonlinear filtering, is challenging even without the imperative of biological plausibility. Apart from a few exceptions for a restricted set of problems [4, 13], the Kushner equation [17] as the formal solution to the general filtering problem is infinite-dimensional and thus needs an approximation scheme. The problem becomes even more challenging if underlying model parameters have to be estimated in addition to a hidden state. Approaches based on sampling have proven to be a powerful tool to solve these tasks numerically. In principle they enable any posterior distribution to be represented with an accuracy that depends on the number of samples. On the one hand, so called particle methods (see for instance [6, 14]), generally rely on importance sampling. On the other hand, Langevin sampling techniques [23, 19] perform a noisy gradient ascent on the log likelihood or the log posterior. In particular the latter methods provide a promising ground for a biologically plausible implementation of neural or synaptic sampling [20, 15].

Here, we propose a framework of how the brain could perform core computations such as state estimation from noisy sensory stimuli by considering three distinct levels: the computational, the algorithmic and the implementation level. On the first level, the computational task of performing inference is set in the context of continuous-time continuous-state nonlinear filtering theory. More precisely, the underlying objects or features, which give rise to the sensory stimuli, are thought to correspond to hidden variables that have to be inferred, or filtered, based on noisy stimuli. Secondly, choosing to represent stimuli and filtered states as continuous stochastic variables, we derive an algorithm that performs the filtering task by evolving these variables according to a stochastic differential equation (SDE). This SDE is thought to sample from the full posterior at each time step, and its first moment corresponds to the continuously varying hidden state that is to be inferred. This framework is extended to learn underlying model parameters by a gradient ascent on the log-likelihood. The performance of this algorithm is assessed numerically. On the implementation level, we propose a recurrent neuronal network whose dynamics is governed by the posterior SDE and that solves the filtering task in a sampling-based manner.

2 The generative model

Let us first formalize the filtering task. The vector of unobserved states[1] $\mathbf{x}_t \in \mathbb{R}^n$ at time t is to be inferred based on a time series of noisy observations $\mathcal{Y}_t$, where the set $\mathcal{Y}_t = \{\mathbf{y}_s \in \mathbb{R}^m, s \in [0, t]\}$ comprises all the observations up to time t. A visual representation of this generative model is given in Figure 1a.

In our model, the hidden states $\mathbf{x}_t$ follow an Ito diffusion with any, in general nonlinear, drift term. Further, we assume that the dynamics of our observations $\mathbf{y}_t$ follows from the hidden states via a generative function $\mathbf{g}(\mathbf{x}_t)$. It is common in the filtering literature (cf. [1]) to formulate this generative model in terms of a continuous-time dynamical system that is described by the two coupled Ito stochastic differential equations (SDE):

$$\text{hidden}: \quad \mathrm{d}\mathbf{x}_\mathrm{t} = \mathbf{f}(\mathbf{x}_t)\, dt + \Sigma_x^{1/2}\, d\mathbf{w}_t, \tag{1}$$

$$\text{observation}: \quad \mathrm{d}\mathbf{y}_\mathrm{t} = \mathbf{g}(\mathbf{x}_t)\, dt + \Sigma_y^{1/2}\, d\mathbf{v}_t, \tag{2}$$

where $\mathbf{w}_t \in \mathbb{R}^n$ and $\mathbf{v}_t \in \mathbb{R}^m$ are uncorrelated vector Brownian motion processes with $\langle d\mathbf{w}_t d\mathbf{w}_s^T \rangle = \mathbb{I}^{n\times n}\delta_{ts}dt$ and $\langle d\mathbf{v}_t d\mathbf{v}_s^T \rangle = \mathbb{I}^{m\times m}\delta_{ts}dt$, respectively, where $\mathbb{I}$ denotes the unit matrix in the corresponding dimension and $\delta_{ts} = 1$ if $t = s$ and $\delta_{ts} = 0$ otherwise, respectively. The functions $\mathbf{f}(\mathbf{x}_t) : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{g}(\mathbf{x}_t) : \mathbb{R}^n \to \mathbb{R}^m$ are vector-valued functions of the hidden variable. We further consider the noise covariances Σ_x and Σ_y to be diagonal matrices such that the noise in each dimension of the stochastic processes $\mathbf{x}_t$ and $\mathbf{y}_t$ is independent.

The processes in Eqs. (1) and (2) define probability distributions $p(\mathbf{x}, t) = p(\mathbf{x}_t)$, which corresponds to the prior, and $p(\mathbf{y}, t|\mathbf{x}_t) = p(\mathbf{y}_t|\mathbf{x}_t)$ over the hidden variables and observations at each time t. In general, the evolution of the probability distribution $p(\mathbf{x}_t)$ is given by [8]

$$dp(\mathbf{x}_t) = \mathcal{L}\left[p(\mathbf{x}_t)\right] dt, \qquad \text{with} \tag{3}$$

$$\mathcal{L}\left[p(\mathbf{x}_t)\right] = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left[f_i(\mathbf{x}_t)p(\mathbf{x}_t)\right] + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}\left[\Sigma_{x,ij}p(\mathbf{x}_t)\right]. \tag{4}$$

[1] For consistency, vectors will be printed in bold face, i.e. $\mathbf{v} = (v_1, v_2, \dots)$.

where $\mathcal{L}$ denotes the Fokker-Planck operator.[2]

3 Approximate Inference with a Neural Filter

In this section we want to derive a nonlinear filter that is able to approximate the posterior state density $p(\mathbf{x}_t|\mathcal{Y}_t)$ and thus to infer the hidden states based on the observations $\mathcal{Y}_t$. Starting from a general framework known from the filtering literature (cf. for instance [10]), an Ito SDE for a random variable $\hat{\mathbf{x}}_t$ is formulated that samples an approximate posterior density. In a second step the SDE is rewritten such that it can be implemented in a recurrent neuronal network and hence meets biological constraints.

3.1 General, nonlinear filtering in continuous time

A formal solution of the filtering problem in Eqs. (1) and (2) is given by the Kushner equation [17], a stochastic partial differential equation for the posterior probability distribution over the inferred hidden variables, conditioned on the observations $\mathcal{Y}_t$:[3]

$$dp(\mathbf{x}_t|\mathcal{Y}_t) \quad = \quad \mathcal{L}\big(p(\mathbf{x}_t|\mathcal{Y}_t)\big)\,dt + p(\mathbf{x}_t|\mathcal{Y}_t)\cdot\big(d\mathbf{y}_t - \langle\mathbf{g}(\mathbf{x}_t)\rangle\big)^T\Sigma_y^{-1}\big(\mathbf{g}(\mathbf{x}_t) - \langle\mathbf{g}(\mathbf{x}_t)\rangle\big), \quad (5)$$

where the Fokker-Planck operator $\mathcal{L}(p)$ was defined in Eq. (4) and follows from the hidden dynamics. From the Kushner equation, it is possible to derive SDEs for posterior expectations $\langle\phi(\mathbf{x}_t)\rangle$:[4]

$$d\langle\phi\rangle \quad = \quad \langle\mathcal{L}^\dagger(\phi)\rangle dt + \mathrm{cov}(\phi, \mathbf{g}(\mathbf{x}_\mathrm{t})^\mathrm{T})\Sigma_\mathrm{y}^{-1}\big(\mathrm{d}\mathbf{y}_\mathrm{t} - \langle\mathbf{g}(\mathbf{x}_\mathrm{t})\rangle\mathrm{dt}\big), \quad \text{with} \quad (6)$$

$$\mathcal{L}^\dagger(\phi(\mathbf{x})) \quad = \quad \sum_i f_i(\mathbf{x})\frac{\partial}{\partial x_i}\phi(\mathbf{x}) + \frac{1}{2}\sum_{i,j}\Sigma_{x,ij}\frac{\partial^2}{\partial x_i\partial x_j}\phi(\mathbf{x}), \quad (7)$$

where $\phi(\mathbf{x}_t)$ is a twice-differentiable, scalar-valued function and $\mathcal{L}^\dagger$ is the formal adjoint of the Fokker-Planck operator (see Supplementary Information) for the hidden dynamics (Eq. 1). Note that in order to derive the evolution of the mean $\boldsymbol{\mu}_t = \langle\mathbf{x}_t\rangle$ one has to compute Eq. (6) with $\phi = x_{i,t}$ for the components of the vector $\mathbf{x}_t$, which gives

$$d\boldsymbol{\mu}_t \quad = \quad \langle\mathbf{f}(\mathbf{x}_t)\rangle dt + \mathrm{cov}\big(\mathbf{x}_\mathrm{t}, \mathbf{g}(\mathbf{x}_\mathrm{t})^\mathrm{T}\big)\Sigma_\mathrm{y}^{-1}\big(\mathrm{d}\mathbf{y}_\mathrm{t} - \langle\mathbf{g}(\mathbf{x}_\mathrm{t})\rangle\mathrm{dt}\big). \quad (8)$$

Hence, the mean $\boldsymbol{\mu}_t$ evolves first according to the drift term of the hidden dynamics and second according to the so-called innovations process $d\mathbf{n}_t = d\mathbf{y}_t - \langle\mathbf{g}(\mathbf{x}_t)\rangle dt$, i.e. the difference between expected and observed change in observations, multiplied by a gain factor. In general, the gain factor $\mathrm{cov}\big(\mathbf{x}_\mathrm{t}, \mathbf{g}(\mathbf{x}_\mathrm{t})^\mathrm{T}\big)$ depends on higher-order moments $\langle\mathbf{x}_t^n\rangle$ with $n > 1$ for a non-constant function $\mathbf{g}(\mathbf{x})$. To further illustrate the problem, consider for instance a linear function $\mathbf{g}(\mathbf{x}) = \mathbf{x}$. In this case, the gain factor depends on the second moment, i.e. $\mathrm{cov}\big(\mathbf{x}_\mathrm{t}, \mathbf{x}_\mathrm{t}^\mathrm{T}\big) = \langle\mathbf{x}_\mathrm{t}\mathbf{x}_\mathrm{t}^\mathrm{T}\rangle - \boldsymbol{\mu}_\mathrm{t}\boldsymbol{\mu}_\mathrm{t}^\mathrm{T}$. In turn, attempting to solve for the second moment employing Eq. (6) with a function ϕ quadratic in the hidden states $\mathbf{x}$ will result in an expression depending on the third moment and so on. This induces a closure problem that makes the solutions of the Kushner equation in general infinite dimensional and hence analytically intractable. Finding finite-dimensional approximations to these solutions is thus the starting point for many filtering algorithms (cf. [1, 10]).

3.2 A stochastic differential equation for the posterior

Motivated by the Kushner equation as a formal solution, we now aim at finding an approximation on the posterior distribution. For this, we consider N i.i.d. stochastic processes $\hat{\mathbf{x}}_t^{(k)},\ k = 1, \ldots, N$, conditioned on the observations $\mathcal{Y}_t$, whose trajectories are supposed to sample the posterior distribution, i.e. $p(\mathbf{x}_t|\mathcal{Y}_t) \approx \tilde{p}(\mathbf{x}_t|\mathcal{Y}_t) = \frac{1}{N}\sum_{k=1}^N \delta(\mathbf{x}_t - \hat{\mathbf{x}}_t^{(k)})$. As an ansatz for the evolution of the $\hat{\mathbf{x}}_t^{(k)} \sim \tilde{p}(\hat{\mathbf{x}}_t|\mathcal{Y}_t)$, we propose the following SDE:

$$d\hat{\mathbf{x}}_t \quad = \quad \mathbf{f}(\hat{\mathbf{x}}_t)dt + W(d\mathbf{y}_t - \mathbf{g}(\hat{\mathbf{x}}_t)dt) + \Sigma_x^{1/2}d\hat{\mathbf{w}}_t, \quad (9)$$

[2] See Supplementary Information for a more elaborate summary on how to compute the evolution of moments and underlying probability distributions from SDEs.

[3] Unless stated explicitly otherwise, triangle brackets $\langle\cdot\rangle$ denote expectations with respect to the posterior distribution $p(\mathbf{x}_t|\mathcal{Y}_t)$.

[4] For the sake of readability we dropped the arguments in $\phi(\mathbf{x}_t)$ and $f(\mathbf{x}_t)$.

where $\hat{\mathbf{w}}_t \in \mathbb{R}^n$ is an uncorrelated vector Brownian motion process with $\langle d\hat{\mathbf{w}}_t d\hat{\mathbf{w}}_s^T \rangle = \mathbb{I}^{n\times n}\delta_{ts}dt$ and W is a - possibly time-dependent - gain matrix. We consider this SDE a reasonable ansatz because firstly its drift term and thus the evolution of its first moment matches Eq. (8) if the gain factor $\mathrm{cov}\big(\mathbf{x}_\mathrm{t}, \mathbf{g}(\mathbf{x}_\mathrm{t})^\mathrm{T}\big)$ is replaced by W and if the "frozen noise" assumption holds, i.e rather than a random process, we assume the set of observations $\mathcal{Y}_T$ to be a given set of externally fixed values for all times up to a final time T. The drift term contains the dynamics of the prior as well as the innovation term $d\hat{\mathbf{n}}_t = d\mathbf{y}_t - \mathbf{g}(\hat{\mathbf{x}}_t)dt$. Second, the noise covariance of Eq. (9) in our ansatz was chosen such that the adjoint Fokker-Planck operator $\tilde{\mathcal{L}}^\dagger$ of Eq. (9) matches Eq. (6) as closely as possible (see also Supplementary Information), i.e.

$$d\langle\phi\rangle \approx \langle\tilde{\mathcal{L}}^\dagger(\phi)\rangle dt, \quad \text{with} \tag{10}$$

$$\tilde{\mathcal{L}}^\dagger(\phi(\mathbf{x})) = \sum_i \big(f_i(\mathbf{x}) + [W\big(\frac{d\mathbf{y}_t}{dt} - \mathbf{g}(\mathbf{x})\big)]_i\big)\frac{\partial}{\partial x_i}\phi(\mathbf{x}) + \frac{1}{2}\sum_{i,j}\Sigma_{x,ij}\frac{\partial^2}{\partial x_i \partial x_j}\phi(\mathbf{x}). \tag{11}$$

Consequently, in the limit of no observations the prior dynamics of the hidden variables is retrieved.

3.3 Interpretation as a recurrent neuronal network

By introducing the matrix W we are actually taking one step back from Eq. (8) to account for further constraints on the implementation level. Since we want to implement this filter as a neuronal network in a biologically plausible way, we will now consider different choices of W that significantly influence possible interpretations and implementations. Though the underlying mathematical framework is general, in the following we consider a linear observation model, that is a linear generative function $\mathbf{g}(\mathbf{x}) = J\mathbf{x}$ with a generative weight matrix J, a choice which allows for a more straightforward neuronal implementation.

Let us first consider the choice of W being an arbitrary, possibly time dependent matrix. In this case, Eq. (9) may be interpreted as a neuronal filter where the $\hat{\mathbf{x}}_t^{(k)}$ correspond to a population of filter neurons whose internal dynamics is governed by the nonlinear function $f(\mathbf{x})$ and whose neuronal activities represent samples of the posterior. These filter neurons are recurrently connected to populations of novelty neurons $\hat{\mathbf{n}}_t^{(k)}$, which evolve according to the innovation term. In this interpretation, W corresponds to the matrix of synaptic weights that connects novelty neurons $\hat{\mathbf{n}}_t^{(k)}$ to filter neurons $\hat{\mathbf{x}}_t^{(k)}$. Alternatively, the same synaptic input can be implemented in a single recurrent network of neurons that directly receive feedforward input from a set of sensory neurons $\mathbf{y}_t$ through synaptic weights W and inhibitory recurrent input from the other filter neurons through the weights WJ. At this point, the weight matrix may be assumed to be a fixed matrix or it may be plastic, i.e. changing according to some learning rule based on the network activity. We will derive a possible learning rule for W in section 4.

By contrast, consider $W = \Sigma_t J^T \Sigma_y^{-1}$, a choice consistent with Eq. (8) for a linear generative function. Since we do not know the posterior variance Σ_t, we consider it to correspond to the empirical variance computed from the random trajectories of $\hat{\mathbf{x}}_t$ at each time step instead. An explicit dependence on Σ_y^{-1} ensures Eq. (9) to be governed mainly by the prior dynamics in the case of large observation noise. In the limit of small observation noise, the innovation term dominates the dynamics and the trajectories from Eq. (9) will evolve to states that perfectly account for the observations, i.e. $0 = d\mathbf{n}_t^{(k)} = d\mathbf{y}_t - J\hat{\mathbf{x}}_t^{(k)}dt$. What is more, thinking in terms of the computational task, which is to estimate a hidden trajectory $\mathbf{x}_t$ based on the observations, it makes sense to feed back an estimate about the uncertainty given by the empirical variance Σ_t into the trajectories: The more uncertain the state estimate, the more the innovation term should be accounted for and thus incorporated into the instantaneous dynamics. We will see in section 5 numerically that using the empirical variance in W leads indeed to a satisfactory filtering performance. However, using the empirical variance may add some difficulties when implementing this architecture on a neuronal network. Thinking of W in terms of a weight matrix, the empirical variance $\Sigma_t = \langle\hat{\mathbf{x}}\hat{\mathbf{x}}^T\rangle - \boldsymbol{\mu}_t\boldsymbol{\mu}_t^T$ is the only source of plasticity. Obviously, this plasticity rule is non local, implying that each filter neuron $\hat{x}_i^{(k)}$ has to know about the state of every other filter neuron at each time. Clearly, another layer of computation would have to be added to the network in order to perform this cross computation within a neuronal network and feed it into the weight matrix.

4 Learning with diffusive processes

As mentioned in the previous section, we can think of W in Eq. (9) as a weight matrix that can be learned by an explicit learning rule. In the following we will focus on deriving such a learning rule for W by performing a gradient ascent on the likelihood for the set of observations $\mathcal{Y}_t$. For this, we first have to formalize the notion of a likelihood function in the context of nonlinear filtering in continuous time, which will then be maximized with respect to W in order to end up with a suitable online learning rule. Since the likelihood function is general, this concept can easily be applied to determine any of the other model parameters. In the context of nonlinear filtering, the estimation of model parameters is usually referred to as identification problem.

4.1 The likelihood function for diffusive processes

Consider being given a sequence of observations $\mathcal{Y}_t$ that may have been generated by two different diffusion processes, which, in turn, induce two different probability measures $\mathbb{P}$ and $\mathbb{Q}$. In order to decide which of these processes is more likely to have generated the sequence, a likelihood ratio between these models can be computed, which corresponds to the so-called Radon-Nikodym derivative between the induced probability measures [16, p. 282]:

$$\Lambda(\mathcal{Y}_t) \quad = \quad \frac{d\mathbb{Q}(\mathcal{Y}_t)}{d\mathbb{P}(\mathcal{Y}_t)}. \tag{12}$$

Loosely speaking, a large value of $\Lambda(\mathcal{Y}_t)$ provides evidence for the diffusion model inducing $\mathbb{Q}$ and vice versa. Here, we make use of the innovation process $\mathbf{n}_t$ to write an SDE for $\mathbf{y}_t$ (cf. [21]):

$$d\mathbf{y}_t \quad = \quad \langle \mathbf{g}_\theta(\mathbf{x}_t) \rangle_{P_\theta(\mathbf{x}_t|\mathcal{Y}_t)} dt + d\mathbf{n}_t. \tag{13}$$

Under the measure $\mathbb{Q}_\theta$, which corresponds to the original measure (or rather to a family of measures parametrized by θ) induced by Eqs. (1) and (2) when setting $\Sigma_y = \mathbb{I}$, $\mathbf{n}_t$ is a $\mathcal{Y}_t$-adapted Brownian motion process [1, p. 33]. Therefore, a marginalization over the hidden variables $\mathbf{x}_t$ leaves us with a measure over the observations $\mathbb{Q}_\theta(\mathcal{Y}_t)$. In order to determine how likely the observations were generated from the model in Eq. (13), we compute the Radon-Nikodym derivative of $\mathbb{Q}_\theta$ with respect to the Wiener measure $\mathbb{P}$, a measure under which $\mathbf{y}_t$ is a Brownian motion process independent of hidden variables as well as parameters.[5] The corresponding Radon-Nikodym derivative can be computed by applying Girsanov's theorem [1, cf. Eq. 3.18 on p. 52]:

$$\Lambda_\theta(\mathcal{Y}_t) = \frac{d\mathbb{Q}_\theta(\mathcal{Y}_t)}{d\mathbb{P}(\mathcal{Y}t)} \quad = \quad \exp\left(\int_0^t \langle \mathbf{g}_\theta(\mathbf{x}_s) \rangle^T d\mathbf{y}_s - \frac{1}{2} \langle \mathbf{g}_\theta(\mathbf{x}_s) \rangle^T \langle \mathbf{g}_\theta(\mathbf{x}_s) \rangle ds \right). \tag{14}$$

The objective is now to maximize this likelihood ratio, or equivalently its logarithm, for our generative model with respect to the model parameters θ. Thus, the log-likelihood function $L_t(\theta)$ for our model with linear observations[6] reads:

$$L_t(\theta) \quad = \quad \int_0^t \left(\boldsymbol{\mu}_s^T J^T \Sigma_y^{-1} d\mathbf{y}_s - \frac{1}{2} \boldsymbol{\mu}_s^T J^T \Sigma_y^{-1} J \boldsymbol{\mu}_s ds \right). \tag{15}$$

4.2 An online maximum likelihood estimate for the filter weight matrix

In order to maximize the likelihood for the set of observations $\mathcal{Y}_t$ we perform a gradient ascent on Eq. 15 with respect to the components of the weight matrix W_{ij} (for detailed computation steps, see Supplementary Information)

$$\begin{aligned} \Delta W_{ij} \quad &= \quad \eta_W \partial_{W_{ij}} L_t(W_{ij}) \\ &= \quad \eta_W \int_0^t \left(\partial_{W_{ij}} \boldsymbol{\mu}_s \right)^T J^T \Sigma_y^{-1} (d\mathbf{y}_s - J \boldsymbol{\mu}_s ds), \end{aligned} \tag{16}$$

[5] Actually, it is completely arbitrary which measure we choose as the reference measure $\mathbb{P}$ because we are trying to maximize a likelihood *ratio* (cf. [5]). Here, the Wiener measure as a choice of reference is advantageous because the Radon-Nikodym derivative is straightforward to compute.

[6] Note that we had to rescale Eq. (2) such that it has unit variance. Hence in Eq. (14) we have to set $\mathbf{g}_\theta(\mathbf{x}_t) \to \Sigma_y^{-1} J \mathbf{x}_t$ and $d\mathbf{y}_t \to \Sigma_y^{-1} d\mathbf{y}_t$

where η_W denotes the learning rate. Equation 16 gives rise to the following online learning rule

$$dW_{ij} \quad = \quad \eta_W (\partial_{W_{ij}} \boldsymbol{\mu}_t)^T J^T \Sigma_y^{-1} (d\mathbf{y}_t - J\boldsymbol{\mu}_t dt), \tag{17}$$

where we introduced the filter derivative $\partial_{W_{ij}} \boldsymbol{\mu}_t$ through which the filter explicitly enters the learning rule. For the neural filter defined in Eq. (9), we may use the empirical mean of N samples to estimate the filter derivative:

$$\partial_{W_{ij}} \boldsymbol{\mu}_t \quad = \quad \frac{1}{N} \sum_{k=1}^{N} \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{\alpha}_{kij}, \tag{18}$$

with dynamics of the tensor $\boldsymbol{\alpha}_{kij}$ determined by the neural filter equation (9)

$$d\boldsymbol{\alpha}_{kij} \quad := \quad \partial_{W_{ij}} \left(d\hat{\mathbf{x}}_t^{(k)} \right) = \left(F(\hat{\mathbf{x}}_t^{(k)}) - WJ \right) \boldsymbol{\alpha}_{kij} dt + [d\mathbf{y}_t - J\mathbf{x}_t^{(k)} dt]_j \mathbf{e}_i, \tag{19}$$

where $F_{ij} = \frac{\partial f_i}{\partial x_j}$ denotes the Jacobian of the nonlinear hidden dynamics and $\mathbf{e}_i$ denotes the unit vector in the i-th direction.

Taken together, when we assume W to be a plastic weight matrix that is learned as observations become available, at least three equations are needed to infer the hidden state at each timestep: First, Eq. (9) to evolve the states of the neurons, and second, Eqs. (17) and (19) to update the weights in the filter equation.

4.3 An online maximum likelihood estimate for the generative weight matrix

Of course, we can proceed analogously to compute any of the model parameters. As an example, we consider learning the generative weight matrix J. By again performing a gradient ascent on Eq. (15) with respect to its components J_{ij}, a suitable online learning rule for J is derived as:

$$dJ_{ij} \quad = \quad \eta_J (\partial_{J_{ij}} \boldsymbol{\mu}_t)^T J^T \Sigma_y^{-1} (d\mathbf{y}_t - J\boldsymbol{\mu}_t dt) + \eta_J [\Sigma_y^{-1} (d\mathbf{y}_t - J\boldsymbol{\mu}_t dt) \boldsymbol{\mu}_t^T]_{ij}, \tag{20}$$

where we again introduced a filter derivative for J, given by $\partial_{J_{ij}} \boldsymbol{\mu}_t = \frac{1}{N} \sum_{k=1}^{N} \partial_{J_{ij}} \hat{\mathbf{x}}_t^{(k)} = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{\beta}_{kij}$ with dynamics

$$d\boldsymbol{\beta}_{kij} \quad := \quad \partial_{J_{ij}} \hat{\mathbf{x}}_t^{(k)} = \left(F(\hat{\mathbf{x}}_t^{(k)}) - WJ \right) \boldsymbol{\beta}_{kij} dt - \hat{x}_{t,j}^{(k)} W \mathbf{e}_i dt. \tag{21}$$

Unfortunately, neither the learning rule for W nor the one for J is local, implying that the weights can only be computed when knowing the state of each filter neuron at each time. In order to learn the weights in this framework, another layer of computation for the weights would have to be introduced.

5 Numerical validation on a nonlinear system

Our approach is now validated using a one-dimensional example with a nonlinear hidden dynamics given by $f(x) = 4x(1 - x^2)$. Figure 1b shows three sample trajectories and the bimodal stationary distribution resulting from this process. As mentioned earlier, unlike in the case of a linear hidden dynamics where the solution is given by the Kalman-Bucy filter [13], a closed-form solution of the posterior does not exist in this case. Therefore, as a benchmark for this system, we use a standard particle filter (PF), which we assume to sample the posterior distribution at each time step. In addition, we compare our neural filter (NF) to the extended Kalman filter (EKF), a filter that parametrizes the posterior by mean and variance only and relies on a local linearization of the nonlinear function. In the following, we consider the three different choices for W, which were introduced in subsection 3.3 and which will be referred to as W_{learned}, W_{emp} and W_{const}, respectively.

Comparing the posterior distribution to that of the PF, we see that it is indeed well captured. For instance, for the simulation in Figure 1c with $\Sigma_y = 0.1$, $\Sigma_x = 0.1$, $J = 1$, $\eta_W = 0.1$ and $N = 1000$ it is shown that the neural filter with W_{learned} (cf. section 4) is able to capture the features of the posterior distribution. It is slightly skewed from its mean, taking into account the bimodal shape of the prior, as becomes also apparent in the simulations done with the PF. Generally, the neural filter tends to somewhat underestimate the posterior variance, but comparing the evolution of NF posterior

variance to that estimated by the PF in time, they overall agree reasonably well, while that estimated by the EKF differs substantially. Empirically estimated higher-order moments are consistent to a sufficient extent with that of the PF. That the NF is able to estimate the hidden state very well becomes apparent when looking at the MSE $E = 1/T \sum_t (x_{\mathrm{hidden,t}} - \mu_t)^2$ for the nonlinear system (Figure 1d). Though the performance of the PF as the baseline solution is always best, each of the neural filter variants perform nearly as well over the range of observation noise Σ_y that was tested. Note that for large observation noise, the variance of the filter is identical to the prior variance, implying that in this limit the filter trajectories follow the prior dynamics. By contrast, the EKF does by definition not capture higher order moments of the posterior and thus fails to reproduce its features as shown in Figure 1c. For larger observation noise, matters become even worse because the state estimate of the EKF evolves to one of the fixed points of $f(x)$ and remains there, irrespective of the real hidden state. This explains why its predictive performance is fairly poor compared to that of any of the NF variations tested.

On the next level of complication, we performed simulations where the generative weight J had to be learned following Eqs. (20) and (21) in addition to finding the hidden state. This identification problem could be solved efficiently by both the NF with W_{learned} and the NF with W_{emp} (Figure 1e, simulation with parameters as before, a true generative weight $J = 1$ and $\eta_J = 0.005$). Note that for the NF variant with W_{learned}, actually two parameters have to be learned simultaneously, which we expect not to be independent of each other by the way the model is set up (cf. Eq. 8 with Eq. 9). Nevertheless, the model is able to learn the generative weight parameter J. Values of J resulting from the learning are slightly biased towards values below the ground truth generative weight, whereas J values learned with the empirical NF variant tend to be slightly biased towards larger values, and always fluctuate within a 10% range of the true value. Additionally, upon getting closer to the true underlying parameters[7] the error is reduced and thus filter performance improves.

6 Discussion

In this paper, we formulated the computational task of inference as a generative model where the continuous-valued hidden variables and the observations evolve continuously in time. Based on the theory of nonlinear filtering, we proposed an Ito SDE for the posterior process and derived a learning rule for the parameters of this filter itself as well as for the generative weights of the underlying generative model. We then introduced a recurrent dynamical network governed by the dynamics of the posterior SDE, which may serve as a possible sampling-based implementation of Bayesian inference.

Indeed, the sampling-based framework is a central result of our proposed neural filter. Unlike the various extensions of the Kalman filter [12], which are applied to nonlinear systems, such as the EKF or the unscented Kalman filter [11], the neural filter is not restricted to approximate the posterior by a Gaussian parametrized by its first and second moment. Rather, due to its nonlinearity the network dynamics may represent any probability distribution at any given time step. We numerically validated that this distribution approximated the real posterior distribution with a numerical performance close to the standard PF. In addition, the neural filter is well in accordance with existing, sample-based filtering approaches. On the one hand it may be seen as a PF itself where all particles carry the same weight and that, therefore, avoids numerical pitfalls such as weight degeneracy. On the other hand, it is also structurally consistent with Langevin sampling, as proposed for instance in [9]. Moreover, it even generalized the Langevin-sampling approach that assumes a Gaussian prior, i.e. a linear hidden dynamics, by including nonlinear hidden dynamics and, in principle, may incorporate a nonlinear observation model.

In future work, the biological plausibility of the recurrent network model will be further addressed. First, observing that the learning rules we derived in section 4 do not fulfill the requirement of locality, which is needed for a biologically plausible learning rule (e.g. a Hebbian learning rule), the model could be enhanced such that individual neurons obey *different* rather than identical dynamics, depending on their own state and on interactions with all the other neurons in the network. Second, by including the theory of filtering and identification of point processes, it may be extended such that spike-based representation may be accounted for.

[7] Of course, we have only access to the ground truth of the generative weight J but not to the filter weight W.

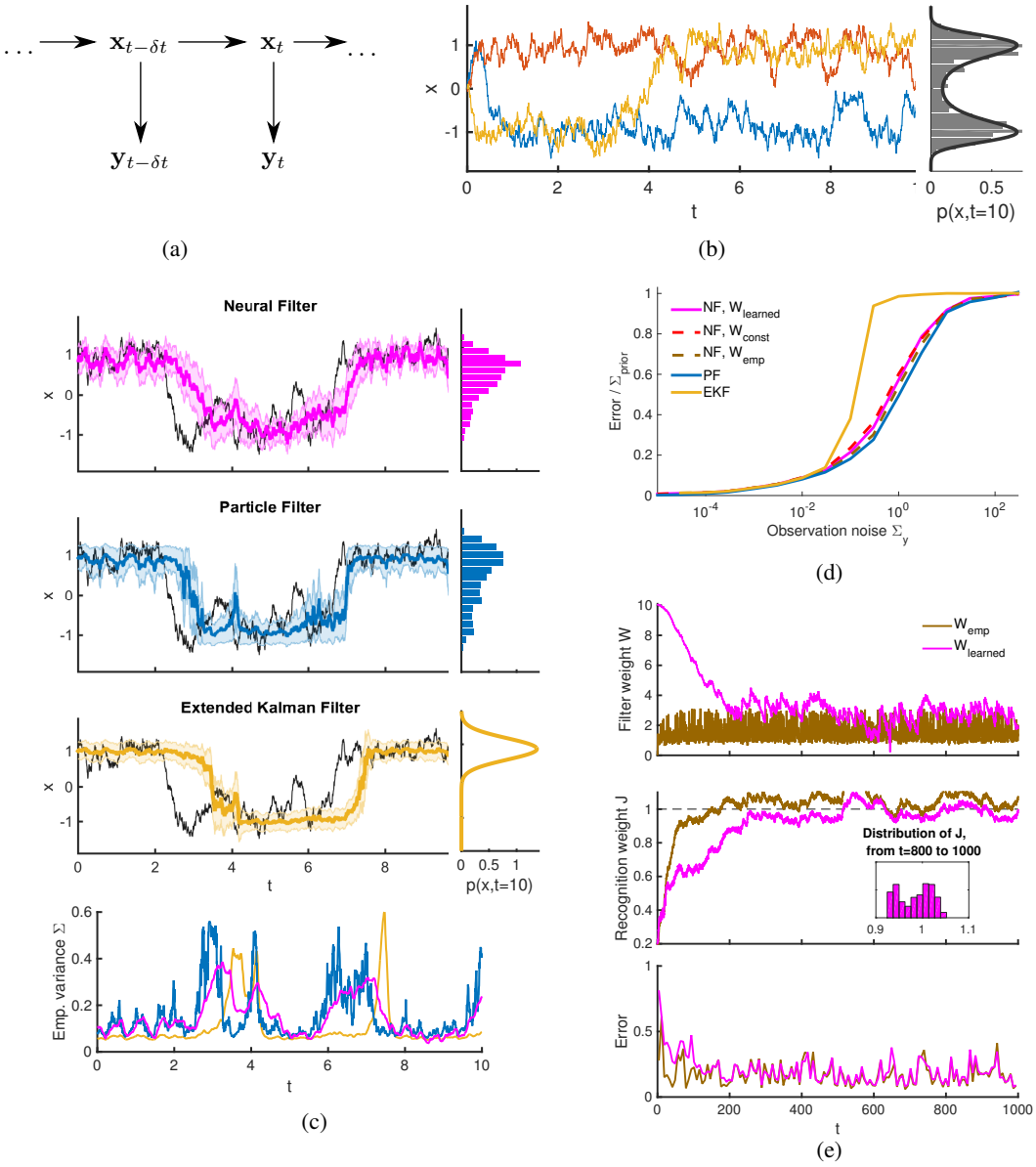

Figure 1: **(a)** The generative model. Here, we consider continuous time, hence $\delta t \to 0$. **(b)** Three sample trajectories of the prior drawn from Eq. (1) with $f(x) = -4x(1-x^2)$ and $\Sigma_x = 1$. The histogram of 1000 sample trajectories at $t = 10$ reflects the stationary distribution (thick line). **(c)** Comparison of estimated posterior trajectories. The underlying hidden trajectory is represented by the black line. Magenta, blue and yellow lines correspond to the state estimation μ_t and shaded regions to the standard deviation estimated by the neural filter (NF) with $W_{learned}$, particle filter (PF) and extended Kalman filter (EKF), respectively. Histograms depict the states of $N = 1000$ sample trajectories at $t = 10$ and are thought to correspond to the posterior density $p(x_{t=10}|\mathcal{Y}_{t=10})$. **(d)** Mean squared error $E = 1/T \sum_t (x_{\text{hidden,t}} - \mu_t)^2$ as a function of observation noise Σ_y. Here, we also compared the error of NF with constant filter weights W_{const} (red) and of NF with filter weights W_{emp} (cyan). **(e)** Parameter values of J and W as well as the MSE for both NF with W_{emp} and NF with W_{learned}. Black line denotes true generative weight. Inset: Histogram reflecting distribution of values of learned generative weights over the last 200 time steps.

References

[1] A. Bain and D. Crisan. *Fundamentals of Stochastic Filtering*. Springer, New York, 2009.

[2] A. K. Churchland, R. Kiani, R. Chaudhuri, et al. Variance as a signature of neural computations during decision making. *Neuron*, 69(4):818–831, 2011.

[3] M. M. Churchland, B. M. Yu, J. P. Cunningham, et al. Stimulus onset quenches neural variability: a widespread cortical phenomenon. *Nature Neuroscience*, 13(3):369–378, 2010.

[4] F. Daum. Nonlinear filters: Beyond the Kalman Filter. *Aerospace and Electronic Systems Magazine, IEEE*, 20(8):57–69, 2005.

[5] A. Dembo and O. Zeitouni. Parameter estimation of partially observed continuous time stochastic processes via the EM algorithm. *Stochastic Processes and their Applications*, 23(1):91–113, 1986.

[6] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[7] J. Fiser, P. Berkes, G. Orbán, and M. Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in Cognitive Sciences*, 14(3):119–130, Mar. 2010.

[8] C. W. Gardiner. *Handbook of Stochastic Methods*. Springer, Heidelberg, 2nd edition, 1985.

[9] G. Hennequin, L. Aitchison, and M. Lengyel. Fast Sampling-Based Inference in Balanced Neuronal Networks. *Advances in Neural Information Processing Systems*, pages 1–9, 2014.

[10] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

[11] S. J. Julier and J. K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. *System Identification*, 3:3–2, 1997.

[12] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[13] R. E. Kalman and R. S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95, 1961.

[14] N. Kantas, A. Doucet, S. S. Singh, J. M. Maciejowski, and N. Chopin. On Particle Methods for Parameter Estimation in General State-Space Models. *Compute*, pages 1–31, 2012.

[15] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass. Network Plasticity as Bayesian Inference. *Arxiv*, 2:1–33, 2015.

[16] F. C. Klebaner. *Introduction to Stochastic Calculus with Applications*. Imperial College Press, 2nd edition, 2005.

[17] H. Kushner. On the Differential Equations Satisfied by Conditional Probability Densities of Markov Processes, with Applications. *Journal of the Society for Industrial & Applied Mathematics, Control*, 2(1), 1962.

[18] W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget. Bayesian inference with probabilistic population codes. *Nature neuroscience*, 9(11):1432–8, Nov. 2006.

[19] D. J. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2005.

[20] R. Moreno-Bote, D. C. Knill, and A. Pouget. Bayesian sampling in visual perception. *Proceedings of the National Academy of Sciences of the United States of America*, 108(30):12491–12496, 2011.

[21] J. M. F. Moura and S. K. Mitter. Identification and Filtering: Optimal Recursive Maximum Likelihood Approach. Technical Report August, 1986.

[22] A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham. Probabilistic brains: knowns and unknowns. *Nature Neuroscience*, 16(9):1170–8, 2013.

[23] M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

A Neural Implementation for Nonlinear Filtering Supplementary information

Anna Kutschireiter
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
annak@ini.uzh.ch

Simone Carlo Surace
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
surace@ini.uzh.ch

Henning Sprekeler
Bernstein Center for Computational Neuroscience
Technische Universitaet Berlin
10587 Berlin, Germany
h.sprekeler@tu-berlin.de

Jean-Pascal Pfister
Institute of Neuroinformatics
University of Zurich, ETH Zurich
8058 Zurich, Switzerland
jpfister@ini.uzh.ch

1 Stochastic differential equations in a nutshell

For readers who are not too familiar with the concepts of Ito stochastic differential equations (SDE), we want to give a very brief overview about how to describe diffusion processes, compute underlying probability distributions and moments from an SDE, and the common discretization scheme that we used to simulate the trajectories.

1.1 Stochastic differential equations, moments and probability distributions

Consider a vector-valued random variable $\mathbf{X}_t \in \mathbb{R}^n$ that evolves according to an Ito diffusion, i.e. it can be described by the Ito SDE

$$d\mathbf{x}_t \quad = \quad \mathbf{a}(\mathbf{x}_t, t)\, dt + B(\mathbf{x}_t, t) d\mathbf{w}_t. \tag{S1}$$

Here, $\mathbf{w}_t \in \mathbb{R}^n$ is a vector Brownian motion process with $\langle d\mathbf{w}_t d\mathbf{w}_s^T \rangle = \mathbb{I}^{n\times n}\delta_{ts}dt$, where $\mathbb{I}$ denotes the unit matrix in the corresponding dimension and further $\delta_{ts} = 1$ if $t = s$ and $\delta_{ts} = 0$ otherwise. The deterministic part of Eq. (S1) is determined by the *drift term* $\mathbf{a}(\mathbf{x}_t, t)dt$ with a vector-valued function $\mathbf{a}(\mathbf{x}_t, t)$, whereas the stochastic part is determined by the *diffusion term* $B(\mathbf{x}_t, t)d\mathbf{w}_t$ with the matrix-valued noise covariance $B(\mathbf{x}_t, t)$.

The process in Eq. (S1) defines a probability distribution $p(\mathbf{X}_t = \mathbf{x}, t) = p(\mathbf{x}_t)$ over the random variable at each time t. In general, the evolution of the probability distribution $p(\mathbf{x}_t)$ is given by the Fokker-Planck equation[1]:

$$dp(\mathbf{x}_t) \quad = \quad \mathcal{L}\left[p(\mathbf{x}_t)\right] dt, \qquad \text{with} \tag{S2}$$

$$\mathcal{L}\left[p(\mathbf{x}_t)\right] \quad = \quad -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left[a_i(\mathbf{x}_t)p(\mathbf{x}_t)\right] + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}\left[B_{ij}(\mathbf{x}_t)p(\mathbf{x}_t)\right]. \tag{S3}$$

where $\mathcal{L}$ is called Fokker-Planck operator [1].

For certain drift and diffusion terms, there exists an analytical solution of Eq. (S2) for the probability distribution $p(\mathbf{x}_t)$. As an example, let us consider a stochastic process $\mathbf{x}_t$ with drift $a_i(\mathbf{x}) = a_i(x_i)$

[1] For the sake of readability, we dropped the explicit t-dependence in the arguments. This, however, does not affect the generality of Eq. (S2) and (S3).

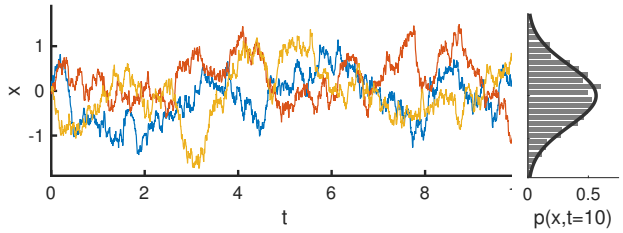

Figure S1: Three sample trajectories of the prior drawn from Eq. (S1) with $f(x) = -x$ and $\Sigma_x = 1$. The histogram of 1000 sample trajectories at $t = 10$ reflects the stationary distribution (thick line).

and noise covariance $B(\mathbf{x}_t) = \mathrm{diag}(b_i(x_i))$. For this process, the dimensions of the stochastic process are decoupled and the stationary distribution with $dp(\mathbf{x}_t) = 0$ can be computed from Eq. (S2):

$$p(\mathbf{x}_t) \quad = \quad \prod_{i=1}^{n} p(x_{t,i}) = \prod_{i=1}^{n} \frac{1}{Z_i} \exp\left(\int_{-\infty}^{x_i} \frac{a_i - \frac{1}{2}\frac{\partial}{\partial x'} b_i(x'')|_{x'}}{\frac{1}{2} b_i(x')} dx' \right), \tag{S4}$$

where Z_i denotes the normalization constant in dimension i.[2]

From the Fokker-Planck equation, it is possible to derive the evolution of the expectation of an arbitrary scalar-valued function $\phi(\mathbf{x})$:

$$\begin{aligned} d\langle \phi(\mathbf{x_t}) \rangle \quad &= \quad \int d\mathbf{x}_t \phi(\mathbf{x}_t) \left(dp(\mathbf{x}_t) \right) \\ &= \quad \left(\int d\mathbf{x}_t \phi(\mathbf{x}_t) \mathcal{L}\left[p(\mathbf{x}_t) \right] \right) dt \\ &= \quad \langle \sum_i a_i(\mathbf{x_t}) \frac{\partial}{\partial x_{t,i}} \phi(\mathbf{x_i}) + \frac{1}{2} \sum_{i,j} B_{i,j}(\mathbf{x}_t) \frac{\partial^2}{\partial x_{t,i} \partial x_{t,j}} \phi(\mathbf{x}_t) \rangle dt \\ &=: \quad \langle \mathcal{L}^\dagger \left(\phi(\mathbf{x_t}) \right) \rangle dt, \end{aligned} \tag{S5}$$

where $\mathcal{L}^\dagger$ is the adjoint Fokker-Planck operator. Note that Eq. (S2) and Eq. (S5) are contained in Eq. (5) and Eq. (6) (Kushner equation) and that the Kushner equations incorporate an additional term due to the observations. To sum up, with a given Ito SDE it is possible to set up equations for the evolution of its underlying probability distribution and for any scalar-valued function by determining the Fokker-Planck operator and its adjoint.

1.2 Numerical Implementation

For numerical simulation of the SDE in Eq. (S1) a time-discretization scheme, the so-called the Euler-Maruyama approximation, can be employed [3] to integrate an Ito SDE for small time steps $\Delta t = t_{n+1} - t_n$:

$$\mathbf{x}_{n+1} \quad = \quad \mathbf{x}_n + \mathbf{a}(\mathbf{x}_n, t_n)\Delta t + B(\mathbf{x}_n, t_n)\Delta \mathbf{w}_n, \tag{S6}$$

where the increment of the Brownian motion process can be sampled from a Gaussian with zero mean and unit variance, i.e. $\Delta \mathbf{w}_n \sim \mathcal{N}(0, \mathbb{I}\Delta t)$. In the paper, this scheme was employed to numerically integrate Eq. (9).

As an example, let us consider one-dimensional stochastic processes governed by the SDE in Eq. (S1) with $a(x) = -x$ and $b(x) = 1$[3]. The resulting stationary distribution following from Eq. (S4) is a Gaussian distribution with zero mean and variance $\sigma^2 = \frac{1}{2}$. In Figure S1 we show 3 sample trajectories from the SDE and the histogram resulting from 1000 sample trajectories at time $t = 10$. This histogram nicely reflects the stationary distribution obtained analytically (thick line).

[2] Note that depending on $\mathbf{a}(\mathbf{x})$ Eq. (S4) could for instance be non-normalizable, implying that no stationary distribution exists for this process.

[3] More precisely, this SDE describes an Ornstein-Uhlenbeck process.

2 Derivation of online learning rules

Here, we want to give the full calculations leading to Eqs. (17), (19), (20) and (21) in the paper. The log-likelihood function for the linear observation model reads:

$$L_t(\theta) = \int_0^t \left(\boldsymbol{\mu}_s^T J^T \Sigma_y^{-1} d\mathbf{y}_s - \frac{1}{2} \boldsymbol{\mu}_s^T J^T \Sigma_y^{-1} J \boldsymbol{\mu}_s ds \right). \tag{S7}$$

In order to maximize the likelihood for the set of observations $\mathcal{Y}_t$ we perform a gradient ascent on Eq. (S7) with respect to the components of the weight matrix W_{ij}:

$$\begin{aligned}
\Delta W_{ij}^{\text{offline}} &= \eta_W \partial_{W_{ij}} L_t(W_{ij}) \\
&= \eta_W \int_0^t \left(\left(\partial_{W_{ij}} \boldsymbol{\mu}_s^T \right) J^T \Sigma_y^{-1} d\mathbf{y}_s - \left(\partial_{W_{ij}} \boldsymbol{\mu}_s^T \right) J^T \Sigma_y^{-1} J \boldsymbol{\mu}_s ds \right), \\
&= \eta_W \int_0^t \left(\partial_{W_{ij}} \boldsymbol{\mu}_s \right)^T J^T \Sigma_y^{-1} (d\mathbf{y}_s - J \boldsymbol{\mu}_s ds), \qquad \text{(S8)} \\
\Rightarrow dW_{ij}^{\text{online}} &= \eta_W \left(\partial_{W_{ij}} \boldsymbol{\mu}_t \right)^T J^T \Sigma_y^{-1} (d\mathbf{y}_t - J \boldsymbol{\mu}_t dt), \qquad \text{(S9)}
\end{aligned}$$

where in Eq. (18) we defined the filter derivative $\partial_{W_{ij}} \boldsymbol{\mu}_t = \frac{1}{N} \sum_{k=1}^N \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} = \frac{1}{N} \sum_{k=1}^N \boldsymbol{\alpha}_{kij}$, with dynamics of the tensor $\boldsymbol{\alpha}_{kij}$ determined by the neural filter equation (9) with linear observation dynamics $\mathbf{g}(\mathbf{x})$:

$$\begin{aligned}
d\boldsymbol{\alpha}_{kij} &:= \partial_{W_{ij}} \left(d\hat{\mathbf{x}}_t^{(k)} \right) \\
&= \partial_{W_{ij}} \left(d\mathbf{f}(\hat{\mathbf{x}}_t^{(k)}) dt + W \left(d\mathbf{y}_t - J \hat{\mathbf{x}}_t^{(k)} dt \right) + \Sigma_x^{1/2} d\hat{\mathbf{w}}_t, \right) \\
&= F(\hat{\mathbf{x}}_t^{(k)}) \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} dt + \left[d\mathbf{y}_t - J \hat{\mathbf{x}}_t^{(k)} dt \right]_j \mathbf{e}_i - W J \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} dt \\
&= \left(F(\hat{\mathbf{x}}_t^{(k)}) - WJ \right) \boldsymbol{\alpha}_{kij} dt + [d\mathbf{y}_t - J \mathbf{x}_t^{(k)} dt]_j \mathbf{e}_i, \qquad \text{(S10)}
\end{aligned}$$

where $F_{ij} = \frac{\partial f_i}{\partial x_j}$ denotes the Jacobian of the nonlinear hidden dynamics and $\mathbf{e}_i$ denotes the unit vector in the i-th direction.

Similarly, we can perform a gradient ascent on Eq. (S7) with respect to the generative weights J. Note, that in this case, the likelihood function explicitly depends on J, in addition to an implicit dependence via the filter derivative:

$$\begin{aligned}
\Delta J_{ij}^{\text{offline}} &= \eta_J \partial_{J_{ij}} L_t(J_{ij}) \\
&= \eta_J \int_0^t \left(\partial_{J_{ij}} \boldsymbol{\mu}_s \right)^T J^T \Sigma_y^{-1} (d\mathbf{y}_s - J \boldsymbol{\mu}_s ds) \\
&\quad + \eta_J \int_0^t \left(\left[\Sigma_y^{-1} d\mathbf{y}_s \boldsymbol{\mu}_s^T \right]_{ij} - \left[\Sigma_y^{-1} J \boldsymbol{\mu}_s \boldsymbol{\mu}_s^T \right]_{ij} ds \right) \\
&= \eta_J \int_0^t (\partial_{J_{ij}} \boldsymbol{\mu}_s)^T J^T \Sigma_y^{-1} (d\mathbf{y}_s - J \boldsymbol{\mu}_t ds) \\
&\quad + \eta_J \int_0^t [\Sigma_y^{-1} (d\mathbf{y}_s - J \boldsymbol{\mu}_t ds) \boldsymbol{\mu}_s^T]_{ij}, \qquad \text{(S11)} \\
\Rightarrow dJ_{ij}(t)^{\text{online}} &= \eta_J (\partial_{J_{ij}} \boldsymbol{\mu}_t)^T J^T \Sigma_y^{-1} (d\mathbf{y}_t - J \boldsymbol{\mu}_t dt) \\
&\quad + \eta_J [\Sigma_y^{-1} (d\mathbf{y}_t - J \boldsymbol{\mu}_t dt) \boldsymbol{\mu}_t^T]_{ij}, \qquad \text{(S12)}
\end{aligned}$$

where we again introduced a filter derivative for J, given by $\partial_{J_{ij}} \boldsymbol{\mu}_t = \frac{1}{N} \sum_{k=1}^N \partial_{J_{ij}} \hat{\mathbf{x}}_t^{(k)} = \frac{1}{N} \sum_{k=1}^N \boldsymbol{\beta}_{kij}$ with dynamics

$$\begin{aligned}
d\boldsymbol{\beta}_{kij} &:= \partial_{J_{ij}} \hat{\mathbf{x}}_t^{(k)} \\
&= \partial_{J_{ij}} \left(d\mathbf{f}(\hat{\mathbf{x}}_t^{(k)}) dt + W \left(d\mathbf{y}_t - J \hat{\mathbf{x}}_t^{(k)} dt \right) + \Sigma_x^{1/2} d\hat{\mathbf{w}}_t, \right) \\
&= F(\hat{\mathbf{x}}_t^{(k)}) \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} dt - \hat{x}_{t,j}^{(k)} W \mathbf{e}_i dt - W J \partial_{W_{ij}} \hat{\mathbf{x}}_t^{(k)} dt \\
&= \left(F(\hat{\mathbf{x}}_t^{(k)}) - WJ \right) \boldsymbol{\beta}_{kij} dt - \hat{x}_{t,j}^{(k)} W \mathbf{e}_i dt. \qquad \text{(S13)}
\end{aligned}$$

3 Additional results

3.1 Numerical validation on a linear system

In addition to the nonlinear system we considered in the paper, we can also study the performance on a system with a linear hidden dynamics $f(x) = -x$, i.e. the hidden process is given by an Ornstein Uhlenbeck process. In this case the posterior density can be accessed analytically and the optimal filter for the continuous-time system is given by the Kalman-Bucy filter (KBF) [2], i.e. at every moment in time the posterior distribution $p(\mathbf{x}_t|\mathcal{Y}_t)$ is a Gaussian distribution and thus fully parametrized by its mean and variance.

Comparing the neural filter (NF) to the KBF, we oberserve that the neural filter is able to well approximate the posterior distribution. In Figure S2a we show sample trajectories as well as the empirical posterior distribution, which result from the neural filter in Eq. (9) with a W that is learned according to Eqs. (17) and (19) and with a W that is computed using the empirical variance. In this simulation, $\Sigma_y = 0.03$, $\Sigma_x = 0.1$, $J = 1$, $\eta_W = 1$ and $N = 1000$. The tendency of the neural to slightly underestimate the true posterior variance given by the KBF is due to an additional term showing up when computing the variance from Eq. (9) analytically. Empirically estimated higher order moments (up to $n = 4$ was tested) are very close to zero as expected for a Gaussian posterior - note that due to the finite number of particles, they are of course not exactly zero.

Most importantly, despite an underestimation of the posterior variance, the neural filter is able to estimate the hidden state, given by its first moment μ_t, i.e. the computational task it was set out to fulfill. To assess its ability to infer the hidden state, we compute the MSE $E = 1/T \sum_t (x_{\mathrm{hidden,t}} - \mu_t)^2$ as a function of observation noise Σ_y. Our findings in Figure S2b suggest that for a linear system, the neural filter in its varieties that were described in section 3 in the paper are all able to perform the inference task with a performance that is indistinguishable from the KBF.

In simulations where - in addition to the hidden state - the generative weight parameter J had to be learned from the observations, the neural filter with a linear hidden dynamics did not perform as well as the NF with a non-linear hidden dynamics. In principle, the model with W_{learned} was able to learn the underlying parameter (cf. Figure S2c, simulation with parameters as before, a true generative weight $J = 1$ and $\eta_J = 0.008$), but this depended strongly on the choice of initial conditions and the NF was likely to be stuck in what we assume to be a local minimum of the likelihood function upon "bad" initialization. The situation becomes slightly worse for the NF with W_{emp} because it tends to overestimate the generative weight J by 20%. This overestimation may be accounted for by the underestimation of the posterior variance, both of which contribute to the computation of the filter weight.

3.2 Comparison to particle filter and extended Kalman filter

To better illustrate how our model differs from conventional filtering methods such as particle filtering and the (extended) Kalman filter, we present a brief summary in Table 1.

References

[1] C. W. Gardiner. *Handbook of Stochastic Methods*. Springer, Heidelberg, 2nd edition, 1985.

[2] R. E. Kalman and R. S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95, 1961.

[3] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Springer, Berlin, Heidelberg, 1st edition, 1999.

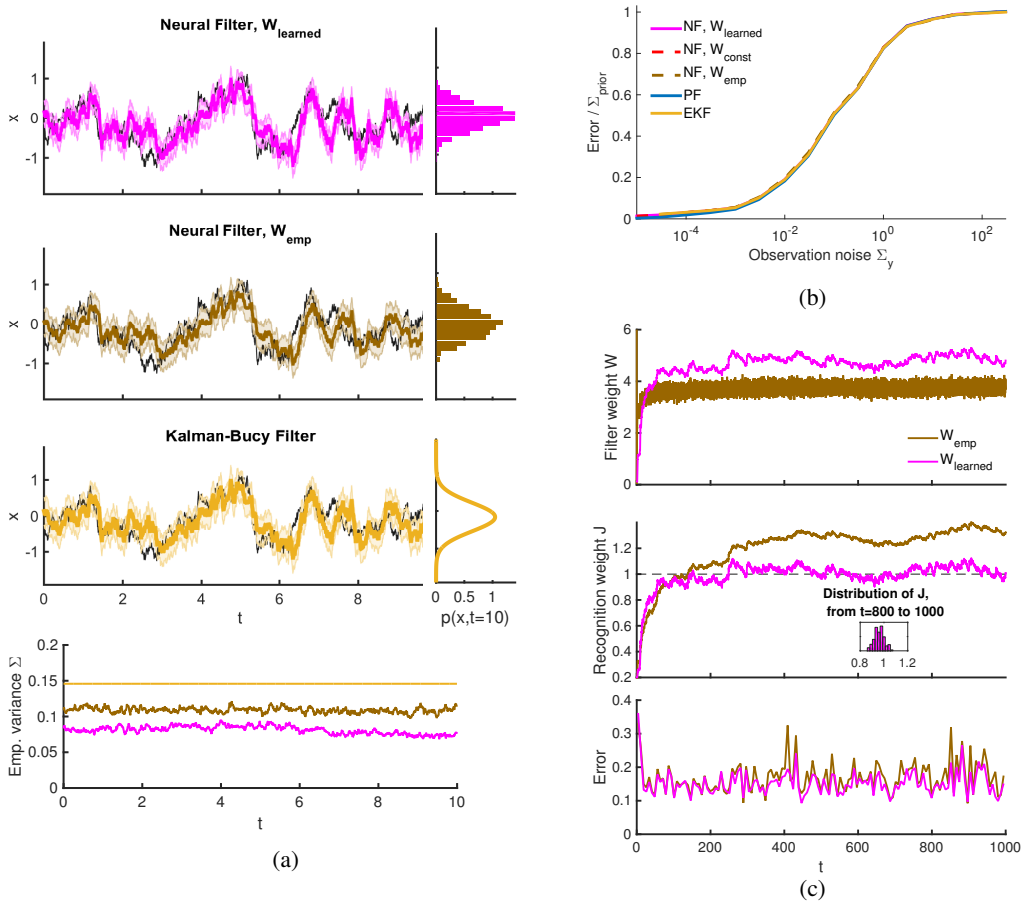

Figure S2: **(a)** Comparison of estimated posterior trajectories. The underlying hidden trajectory is represented by the black line. Magenta, blue and yellow lines correspond to the state estimation μ_t and shaded regions to the standard deviation estimated by the neural filter (NF) with $W_{learned}$, particle filter (PF) and extended Kalman filter (EKF), respectively. Histograms depict the states of $N = 1000$ sample trajectories at $t = 10$ and are thought to correspond to the posterior density $p(x_{t=10}|\mathcal{Y}_{t=10})$. **(b)** Mean squared error $E = 1/T \sum_t (x_{\text{hidden,t}} - \mu_t)^2$ as a function of observation noise Σ_y. Here, we also compared the error of NF with constant filter weights W_{const} (red) and of NF with filter weights W_{emp} (cyan). Note that since the filter performance is comparable in any of the filters, the plot lines are indistinguishable. **(c)** Parameter values of J and W as well as the MSE for both NF with W_{emp} and NF with W_{learned}. Black line denotes true generative weight. Inset: Histogram reflecting distribution of values of learned generative weights over the last 200 time steps.

	EKF	Particle Filter	Neural Filter
Representation	parametric $p(\mathbf{x}_t\|\mathcal{Y}_t) = \mathcal{N}(\mu_t, \Sigma_t)$	weighted samples $p(\mathbf{x}_t\|\mathcal{Y}_t) = \sum_i w_i \delta(\mathbf{x}_t - \mathbf{x}_t^i)$	samples $p(\mathbf{x}_t\|\mathcal{Y}_t) = \frac{1}{N}\sum_i \delta(\mathbf{x}_t - \hat{\mathbf{x}}_t^i)$
Posterior moments	only first and second moment	not restricted	first moment matched, higher order moments approximated
Parameter learning	well-suited	well-suited	well-suited
Limitations for Accuracy	linearization of $f(\mathbf{x}_t, t)$	finite number of particles	finite number of particles,
Biological plausibility	unknown	unclear how to justify importance weights and resampling	neuronal activity corresponds to variables

Table 1: Comparison of the neural filter to the (extended) Kalman filter (EKF) and a standard particle filter.