WHEN DOES PROBABILISTIC INFERENCE IMPROVE TASK LEARNING FOR STOCHASTIC RNNS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent neural networks (RNNs) are used both to solve temporal tasks and to model the dynamics of physical and biological systems. In both settings, recurrent dynamics are often subject to noise, but how this stochasticity is handled in learning algorithms differs. Task optimization typically minimizes output error along noisy rollouts of network dynamics using backpropagation through time (BPTT). System identification instead infers latent trajectories conditioned on observed outputs and learns from this posterior, for example via expectation–maximization (EM). Can such inference-based learning of recurrent weights also benefit task optimization? We compare posterior-inference-based EM with prior-rollout BPTT training and show that the two coincide in the low-noise limit, in both objective and optimization dynamics, but diverge when noise is non-negligible. In this regime, we identify theoretically and empirically three task families where posterior inference improves task learning: long-delay tasks, where it avoids the exponential, delay-dependent decay of BPTT's learning signal; tasks with distribution shifts, where conditioning latent activity on targets reduces catastrophic interference; and tasks with ambiguous targets, where EM captures temporal structure that output-error minimization ignores. Finally, we show that several alternatives to BPTT can be interpreted as different approximate E-steps within a unified EM framework, suggesting that advances in state-space-model inference may offer principled routes to new BPTT alternatives. Together, these results illuminate when inference-based training is advantageous over prior-rollout training in stochastic learning systems.

1 INTRODUCTION

Recurrent neural networks (RNNs) form an important class of nonlinear dynamical systems models. In machine learning and computational neuroscience, they are used to build or model inter-connected networks that implement computational tasks with temporal dependencies(Salehinejad et al., 2017; Sussillo & Barak, 2013; Barak, 2017). Learning the RNN weights is posed as *task-optimization*, where the network weights are adjusted to minimize the error between desired targets and model outputs. In many other scientific applications, RNNs are used to parameterize the nonlinear dynamics of physical or biological processes (Ljung et al., 2020), with applications in neuroscience (Durstewitz et al., 2023; Pandarinath et al., 2018), functional magnetic resonance imaging (Koppe et al., 2019), and nonlinear control (Raiko & Tornio, 2009). Here learning the RNN weights is framed as *system identification*, where the goal is to infer unknown system parameters from observed data by maximizing the likelihood under a generative model, or an approximation thereof (Ghahramani & Roweis, 1998).

In both of these settings, noise plays a special role. In *task-optimization*, noise is added as a regularizer during learning (Jim et al., 1996; Camuto et al., 2020) and to encourage stable and smooth dynamics (Lim et al., 2021; Rungratsameetaweemana et al., 2025; Driscoll et al., 2024; Sussillo et al., 2015). In *system identification*, an explicit noise model captures intrinsic variability of the system of interest (Tsimring, 2014; Faisal et al., 2008), or potential model mismatch. Despite these different motivations, both approaches seek to learn recurrent weights for a stochastic nonlinear dynamical system. They differ, however, in how supervision, whether from task targets or observed data, is incorporated into learning and in how stochasticity is handled by the learning algorithms.

In *task-optimization*, learning typically proceeds by differentiating the optimization objective along noisy samples of network trajectories using backpropagation through time (BPTT). Hidden state

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

trajectories are generated by rolling out the stochastic network dynamics forward in time, conditioned on the inputs. The noise distribution thus mainly influences the variability of network roll-outs under the model's *prior* trajectories. *System identification*, in contrast, typically requires *posterior* inference over hidden trajectories, with learning performed using the Expectation–Maximization (EM) algorithm (Ghahramani & Roweis, 1998), or by directly optimizing a variational objective (Pandarinath et al., 2018). When networks have nonlinear activations, exact posterior inference is computationally intractable, and learning therefore relies on approximate inference methods. Common approaches for nonlinear state-space models include extended Kalman smoothing or other Gaussian variational approximations (Särkkä & Svensson, 2023), particle smoothing (Schön et al., 2011), and Expectation Propagation (Minka, 2013). In this case, uncertainty over network trajectories arising from noise or model misspecification is maintained explicitly as a posterior distribution, and learning updates the network weights using expectations under this posterior.

While EM-based training of RNN parameterizations of dynamical systems is well established (Ma & Ji, 1998; Unkelbach et al., 2009), it has not been assessed whether a system-identification approach with explicit posterior inference can also benefit task optimization. Here we compare prior-rollout BPTT and posterior-inference-based approximate EM in stochastic RNNs to address this gap, and determine when and why the probabilistic inference can improve task learning. First, we find that the two approaches coincide in the low-noise limit (both in objective and in optimization dynamics), but diverge when noise is non-negligible, reflected in differences in task performance in the presence of noise. We then identify three task families in which probabilistic-inference-based training provides advantages over prior-rollout BPTT by producing more effective task-dependent learning signals: (i) tasks with long-term temporal dependencies between input and output, (ii) tasks with distribution shifts, and (iii) tasks with variable or ambiguous targets. Finally, we show analytically that many existing alternatives to BPTT can be interpreted as different approximations to posterior inference within a unified EM framework. A more fundamental understanding of these connections suggest principled routes for improving training strategies for stochastic RNNs using inference approaches that better capture uncertainty in the network trajectories.

2 Two training algorithms for stochastic RNNs: BPTT and EM

We consider the discrete-time RNNs with Gaussian noise innovations

$$\begin{aligned} \mathbf{h}_t &= (1-\alpha)\mathbf{h}_{t-1} + \alpha W_r \phi(\mathbf{h}_{t-1}) + \alpha(W_i \mathbf{x}_t + \mathbf{b}_i) + \sqrt{\alpha}\sigma_h \boldsymbol{\xi}_t, \\ \hat{\mathbf{y}}_t &= W_o \phi(\mathbf{h}_t) + \mathbf{b}_o + \sigma_y \boldsymbol{\epsilon}_t, \qquad t = 1, 2.., T. \end{aligned} \tag{1}$$

Each component of $\boldsymbol{\xi}_t$ and $\boldsymbol{\epsilon}_t$ is an independent standard Gaussian variable. ϕ is the component-wise nonlinear activation, and the initial state is fixed to zero, $\mathbf{h}_0 \equiv 0$. We collect the trainable parameters as $W \equiv \{W_r, W_i, \mathbf{b}_i, W_o, \mathbf{b}_o\}$. The remaining parameters, $\sigma_h, \sigma_y, \alpha \geq 0$, are treated as fixed and represent the intrinsic properties of the system. For a set of given parameter W, the network output at time t, $\hat{\mathbf{y}}_t(\mathbf{x}_{1:t}; W)$ is therefore a *random* function of the input sequence $\mathbf{x}_{1:t}$.

During training, the network receives input sequences $\mathbf{x}^k_{1:T}$, $k = 1, 2, .., P$ and is required to produce the target output $\mathbf{y}^k_t$ at specified time points $t \in \Theta \subseteq \{1, 2.., T\}$. Outputs at other time points are unconstrained. The training data are assumed to be i.i.d. samples from the data distribution, $(\mathbf{x}^k_{1:T}, \mathbf{y}^k_\Theta) \sim \mathcal{D}$. For a given input $\mathbf{x}^k_{1:T}$, since the network's output $\hat{\mathbf{y}}^k_\Theta(\mathbf{x}^k_{1:T}; W)$ is a random variable, this leads to two distinct ways of fitting the training data, corresponding to BPTT and EM respectively. We compare their objective and parameter updates below.

2.1 BPTT objective function and weight update rule

BPTT minimizes the expected discrepancy between the target output and the model prediction. Since the randomness in the model arises from state noise $\boldsymbol{\xi}_{1:T}$ and observation noise $\boldsymbol{\epsilon}_{1:T}$, this dependence can be made explicit, yielding the BPTT objective

$$L_{BPTT}(W) \propto \sum_{k=1}^{P} \mathbb{E}_{\boldsymbol{\xi}^k, \boldsymbol{\epsilon}^k} \left[(\mathbf{y}^k_\Theta - \hat{\mathbf{y}}^k_\Theta(\mathbf{x}^k, \boldsymbol{\xi}^k, \boldsymbol{\epsilon}^k; W))^2 \right]. \tag{2}$$

For notational simplicity, we omit the subscript $1:T$ when no ambiguity arises.

Gradient descent on this objective yields a parameter update at training iteration n

$$W^{n+1} - W^n \propto -\frac{\partial L_{BPTT}(W^n)}{\partial W^n} \propto \sum_{k,t} \mathbb{E}_{\boldsymbol{\xi}^k, \boldsymbol{\epsilon}^k} \left[\boldsymbol{\delta}_t^k \phi(\mathbf{h}_{t-1}^k)^T \right]. \tag{3}$$

Where $\boldsymbol{\delta}_t^k$ denotes the backpropagation error term. Because the expectation in $L_{BPTT}(W)$ is generally intractable, the gradient is typically estimated by sampling the noise and applying BPTT on the resulting stochastic trajectories. In the zero-noise limit, this update rule reduces to the standard BPTT algorithm for deterministic RNNs.

When viewed as a probabilistic model, the output $\hat{\mathbf{y}}_t^k(\mathbf{x}^k; W)$ is Gaussian conditioned on the latent state $\mathbf{h}_t^k$, (Eq.(1), with conditional density $p_W(\mathbf{y}_t|\mathbf{h}_t, \mathbf{x}) \propto \exp\left[-(\mathbf{y}_t - W_o\phi(\mathbf{h}_t) - \mathbf{b}_o)^2/(2\sigma_y^2)\right]$. So up to a constant, Eq.(2) can be equivalently rewritten as

$$L_{BPTT}(W) = -\sum_{k=1}^{P} \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{x}^k)} \left[\ln p_W(\mathbf{y}_\Theta^k|\mathbf{h}^k, \mathbf{x}^k) \right]. \tag{4}$$

Here $\mathbf{h}^k \equiv \mathbf{h}_{1:T}^k, \mathbf{x}^k \equiv \mathbf{x}_{1:T}^k$. The distribution $p_W(\mathbf{h}^k|\mathbf{x}^k)$ denotes the conditional distribution of $\mathbf{h}^k$ given the input $\mathbf{x}^k$, which we refer to as the prior distribution of $\mathbf{h}^k$.

2.2 Maximum likelihood objective and EM

An alternative approach to train the network is to match the full conditional distribution of the network output given the input, $p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k)$, to the training data distribution $p_\mathcal{D}(\mathbf{y}_\Theta^k|\mathbf{x}^k)$. This can be achieved by minimizing the Kullback–Leibler (KL) divergence $\mathrm{KL}\left[p_\mathcal{D}(\mathbf{y}_\Theta^k|\mathbf{x}^k) \| p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k)\right]$, averaged over the training inputs $\mathbf{x}^k$. Since the data distribution does not depend on the parameters W, this is equivalent to minimizing the negative log-likelihood (or log-evidence)

$$L_{EM}(W) \equiv -\sum_{k=1}^{P} \ln p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k). \tag{5}$$

Following standard results in variational inference, minimizing L_{EM} is equivalent to maximizing the evidence lower bound (ELBO) (Wainwright & Jordan, 2008), $\mathrm{ELBO}[q, W] \equiv \mathbb{E}_{q(\mathbf{h}^k)}\left[\ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)\right] + H\left[q(\mathbf{h}^k)\right]$. Here $p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)$ denotes the joint distribution of $(\mathbf{y}_\Theta^k, \mathbf{h}^k)$ given $\mathbf{x}^k$ and $H\left[q(\mathbf{h}^k)\right]$ is the entropy of q.

The EM algorithm (Dempster et al., 1977; McLachlan & Krishnan, 2008) performs coordinate ascent on $\mathrm{ELBO}[q, W]$ by alternatively maximizing q (E-step) and W (M-step). A separate variational distribution q^k is introduced for each training sample $(\mathbf{x}^k, \mathbf{y}_\Theta^k)$.

At training iteration n, the optimal q_*^k for each data point in the E-step is given by the posterior,

$$q_*^k(\mathbf{h}^k) \leftarrow \underset{q}{\mathrm{argmin}}\, \mathrm{KL}[q^k(\mathbf{h}^k) \| p_{W^n}(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)]. \tag{6}$$

For nonlinear RNN models in Eq.(1), this distribution is computationally intractable. This E-step thus requires approaches for approximate inference (Särkkä & Svensson, 2023; Murphy, 2023). For the M-step, parameters are updated according to

$$W^{n+1} \leftarrow \underset{W}{\mathrm{argmax}} \sum_{k=1}^{P} \mathbb{E}_{q_*^k(\mathbf{h}^k)} \left[\ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k) \right] \tag{7}$$

For the stochastic RNN defined in Eq.(1), this M-step admits closed-form solutions (Appendix A).

Alternatively, this M-step (Eq. (7) can also be implemented using a single gradient step. This yields the generalized EM (GEM) algorithm (McLachlan & Krishnan, 2008), with the weight update (using the recurrent weights as an example)

$$W_r^{n+1} - W_r^n \propto \mathbb{E}_{q_*^k}\left[\frac{\partial}{\partial W_r^n} \ln p_{W^n}(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)\right] \overset{\text{Eq.(1)}}{=\!=\!=}_{\alpha=1} \sum_{t=0}^{T-1} \mathbb{E}_{q_*^k}\left[\frac{\left(\mathbf{h}_{t+1}^k - W_r^n\phi(\mathbf{h}_t^k) - W_i\mathbf{x}_{t+1}^k - \mathbf{b}_i\right)\phi(\mathbf{h}_t^k)^T}{\sigma_h^2}\right]. \tag{8}$$

This update closely resembles the BPTT rule in Eq. (3), with the main difference that the backpropagation error term δ_t is replaced by the expected error between $\mathbf{h}_{t+1}$ and the one-time-step network prediction $W_r\phi(\mathbf{h}_t) + W_i\mathbf{x}_t + \mathbf{b}_i$ under the (approximate) posterior q_*^k.

2.3 CONVERGENCE OF BPTT AND EM IN THE LOW-NOISE REGIME

To see the difference between two algorithms, a key observation is that the BPTT (Eq.(4)) and EM objectives (Eq.(5)) can both be derived from the ELBO by choosing different distributions q: BPTT corresponds to replacing q with the prior distribution over $\mathbf{h}^k$ while EM corresponds to using the posterior distribution (Appendix C)

$$L_{BPTT}(W) = -\sum_{k=1}^{P} \mathrm{ELBO}[p_W(\mathbf{h}^k|\mathbf{x}^k), W], \quad L_{EM}(W) = -\sum_{k=1}^{P} \mathrm{ELBO}[p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k), W].$$

This observation may suggest interpreting BPTT as using the prior distribution as an approximate E-step (Eq. (6)). However, directly substituting the prior distribution into the M-step yields zero updates on the recurrent and input weights (Appendix B.5), which differs from the BPTT update (Eq. (3). This discrepancy arises because EM is a coordinate ascent algorithm over (q, W), so during the M-step $p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)$ is treated as constant, whereas BPTT performs direct gradient descent on W, which treats $p_W(\mathbf{h}^k|\mathbf{x}^k)$ as a function of W. Despite this difference, we find that EM converges to BPTT in the small-noise limit (Appendix C). More precisely, as $\sigma_h \downarrow 0$,

1. the difference between the objectives $L_{EM}(W) - L_{BPTT}(W)$ converges to a constant;
2. the parameter update of the GEM algorithm (Eq.(8)) converges to the BPTT (Eq.(3)); and
3. the M-step update (Eq.(7)) converges to a preconditioned version of BPTT update (Eq.(3)).

Given that the two algorithms coincide in the small-noise limit, we therefore focus our analyses on comparing the performance of EM and BPTT when the noise level is non-negligible.

3 COMPARISONS BETWEEN BPTT AND EM IN FINITE-NOISE REGIME

In the following, we identify task settings where EM outperforms BPTT or vice versa. Since the nonlinear RNN does not permit for exact inference, we use approximate inference based on assumed density smoothing (Särkkä & Svensson, 2023) for EM training throughout this section. Full details on numerical simulations are provided in Appendix A and G.

3.1 EVALUATING TASK PERFORMANCE

The main metric we use for evaluating task performance is the mean squared error (MSE) on test data $(\mathbf{x}, \mathbf{y}_\Theta) \sim \mathcal{D}_{test}$,

$$MSE \equiv \mathbb{E}_{\mathcal{D}_{test}} \mathbb{E}_{p_W(\hat{\mathbf{y}}_\Theta|\mathbf{x})} \left[(\mathbf{y}_\Theta - \hat{\mathbf{y}}_\Theta)^2 \right], \tag{9}$$

where $\hat{\mathbf{y}}_\Theta$ denotes network outputs at observation time points Θ. This is precisely the objective optimized by prior-rollout BPTT (Eq.(4), under $\mathcal{D}_{test} = \mathcal{D}$ and a sufficient amount of training data $P \to +\infty$.

For comparison, we also consider a second metric: the prediction error of the (prior) mean predictor, which we refer to as the Bayesian loss (BL),

$$BL \equiv \mathbb{E}_{\mathcal{D}_{test}} \left[(\mathbf{y}_\Theta - \mathbb{E}_{p_W(\hat{\mathbf{y}}_\Theta|\mathbf{x})}[\hat{\mathbf{y}}_\Theta])^2 \right]. \tag{10}$$

With a sufficent amount of data, this is the objective optimized by EM (Eq.(5), Appendix C).

3.2 TEACHER-STUDENT SETTING

We start with the teacher–student setting (Fig. 1A) and evaluate the performance of student networks trained using either BPTT (Eq.(3)) and EM (Eq.(6)-(7)) algorithms, across different state-noise levels σ_h and training set sizes P. Under the MSE metric, BPTT consistently outperforms EM at moderate noise levels (Figs. 1A and 1B, middle). While networks trained with EM yields MSE comparable to that of the teacher, network trained with BPTT can achieve even lower MSE than the teacher network itself (Fig. 1A). This behavior is characteristic of settings where the model has internal noise. In this setting, the best parameter W minimizing the task performance (MSE) need not coincide with the ground-truth parameters of the teacher model, reflecting a bias-variance trade-off (Shmueli, 2010).

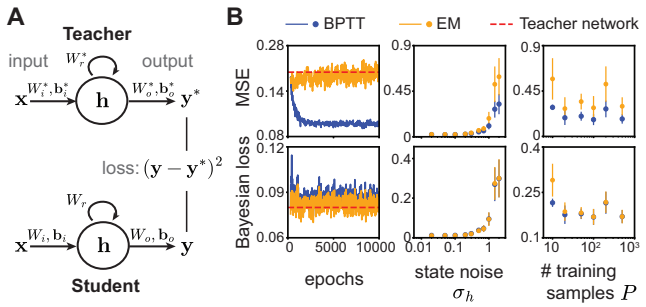


Figure 1: **Teacher-student scenario.** A. Task schematics. B. The performance of BPTT and EM across training epochs, state noise levels σ_h and number of training examples P quantified using the MSE measure and Bayesian loss. Unless specified, $\sigma_h = 1.5$, $P = 200$.

As BPTT is designed to optimize task performance (MSE), the superior MSE performance of BPTT is expected in the large-data regime. Empirically, this advantage persists even with limited training data (Fig. 1B, right).

In contrast, the two algorithms exhibit comparable Bayesian loss across noise levels and training set sizes (Fig. 1C). Although BPTT does not explicitly minimize the Bayesian loss, its comparable performance to EM here may reflect the special teacher–student setup.

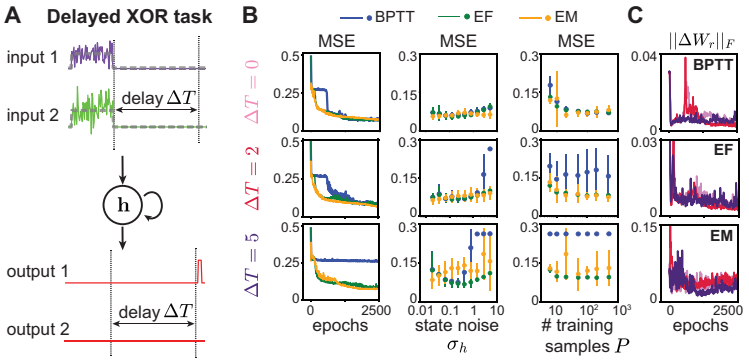


Figure 2: **Tasks with delay periods.** A. Task schematics. B. Generalization performance (MSE) for BPTT and EM for three delay periods. Error forcing (EF) (Sağtekin et al., 2025) is used as an E-step approximation for comparison. As delay and state noise level increases, BPTT generally has worse performance than EM and its approximation EF. C. Norm of the weight change vs. training epoch for the three algorithms, colors denote the different delay durations as indicated in panel B. Curves are smoothed with a Gaussian filter to aid visibility. Shown results are for $\sigma_h = 1.5$ and $P = 10$.

3.3 Tasks with long delay periods

By design, if BPTT reliably finds a global optimum of its objective (Eq.(2)), then (because its training objective is MSE in the large-data regime) it should outperform EM in terms of task performance (MSE). However, because BPTT can suffer from optimization difficulties, EM may outperform BPTT even under MSE metric. This would indicate that EM's advantage stems from its optimization properties of more precisely accounting for randomness. Unless noted otherwise, we report the MSE metric below and leave the Bayesian loss comparison in supplement (Fig. B.2).

A canonical failure mode of BPTT arises in tasks with long-term temporal dependencies, where gradients can vanish or explode over long horizons. As a simple example, we consider a delayed-input–output task with an explicit delay period (Fig. 2A). The task is XOR-like: the 2D input has component-wise means $\pm\mu$, and the output is two-dimensional. The first output unit is active when the two input means are equal, and the second is active when they differ (Fig. 2A). We vary the

delay length ΔT, the state noise level σ_h, and the number of training examples P to compare the performance of BPTT and EM.

When the delay is short ($\Delta T = 0$), BPTT attains a lower test MSE (Fig. 2B, top), consistent with the teacher–student setting. For longer delays, BPTT exhibits an increasingly long training plateau (Fig. 2B, 1st column). This plateau is consistent with vanishing gradients (Bengio et al., 1993; 1994; Pascanu et al., 2013). Also see the $\|\Delta W_r\|_F$ plots in Fig. 2C. EM training does not exhibit this plateau, and its training dynamics are largely insensitive to the delay length. As the noise level increases, BPTT's task performance degrades substantially, whereas EM appears to be more noise robust (Fig. 2B, 2nd column). This robustness might stem from EM's ability to better account for and propagate uncertainty over long horizons, compared to sampling-based BPTT. These effects persist under varying training set size (Fig. 2B, 3rd column) and network size (Fig.B.3, right).

To better understand why EM is robust to long delays and large noise, we investigate the weight updates of BPTT and EM in a linear networks for a similar task (Appendix D), where the input is nonzero only at the initial time step, and the network must produce a target output after a delay ΔT. In the long-delay regime $\Delta T \gg 1$ and for stable network dynamics $|\lambda_{max}(W_\alpha)| < 1$ with $W_\alpha \equiv (1-\alpha)+\alpha W_r$, we show that the task-dependent component of the recurrent-weight updates for the two algorithms exhibit qualitatively different decays, $\|\Delta W_{r,BPTT}\| \leq C\Delta T|\lambda_{max}(W_\alpha)|^{\Delta T-1}$ and $\|\Delta W_{r,EM}\| \leq C' \frac{\sigma_h^2}{\Delta T}$. These results suggests that long delays have a stronger impact on BPTT than on EM. Moreover, the exponential decay in BPTT is weight-dependent, while the linear factor in EM is weight-independent and can be largely compensated for by appropriately choosing the learning rate. The same calculations show that increasing noise also impairs BPTT more severely than EM. Detailed derivations are provided in Appendix D.

In summary, these results demonstrate that EM outperforms BPTT when internal noise is non-negligible and the task requires maintaining information over long delays. In this regime, the E-step can be viewed as a denoising procedure: it smooths the latent-state estimates, yielding more informative weight updates in the subsequent M-step.

3.4 Tasks with distribution shifts

Another difference between the parameter updates in BPTT (Eq. (3)) and EM (Eqs. (6)–(7)) is how the target outputs impact learning. In EM, the outputs are used in the E-step to form the posterior over latent states $p_W(\mathbf{h}^k|\mathbf{y}^k_\Theta, \mathbf{x}^k)$. The M-step update depends on posterior expectations of neural activations. In contrast, BPTT uses the outputs to construct a backpropagating error signal, and updates weights using the product of this error and the presynaptic activity, $\propto \boldsymbol{\delta}^k_t \phi(\mathbf{h}^k_{t-1})$. Importantly, the latter is computed along the (prior) trajectory. Thus, the presynaptic activity is used differently in EM and BPTT (posterior vs. prior, respectively), suggesting that the two methods may differ when learning requires inference from the target.

To investigate this regime further, we consider a context-dependent evidence accumulation task (Fig. 3A) involving two phases: In phase 1, the network is trained from scratch to report the mean of one of two noisy input streams, based on a context cue that indicates which dimension should be integrated (context 1: first dimension, context 2: second dimension). The two contexts are sampled uniformly. In phase 2, we introduce a distribution shift (Fig. 3A, right) after which context 2 is no longer presented, and the input mean in context 1 is halved, thereby reducing the signal-to-noise ratio.

We evaluated the network performance on the shifted context 1 and the original context 2 (Fig. 3B). Both training methods maintain good performance on the shifted context 1. However, on context 2, only EM-trained networks retain good performance after the distribution shift (Fig. 3B, left two columns). Increasing the noise level degrades BPTT's performance substantially more than EM in both contexts (Fig. 3B, top right), illustrating BPTT's sensitivity to catastrophic interference (French, 1999; Goodfellow et al., 2013). This effect remains for different numbers of training examples (Fig. 3B, bottom right) and contexts (Fig.C.4).

Networks trained on context-dependent tasks often develop modular subnetworks specialized for different contexts (Yang et al., 2019; Johnston & Fusi, 2024; Lepori et al., 2023). To explain the post-shift difference between EM and BPTT, we mathematically analyzed a network with an arbitrary number of modules, each active in one context and projecting to the corresponding output units (Fig. 3C, Appendix E), mimicking the network at the end of phase 1. In phase 2, the lower

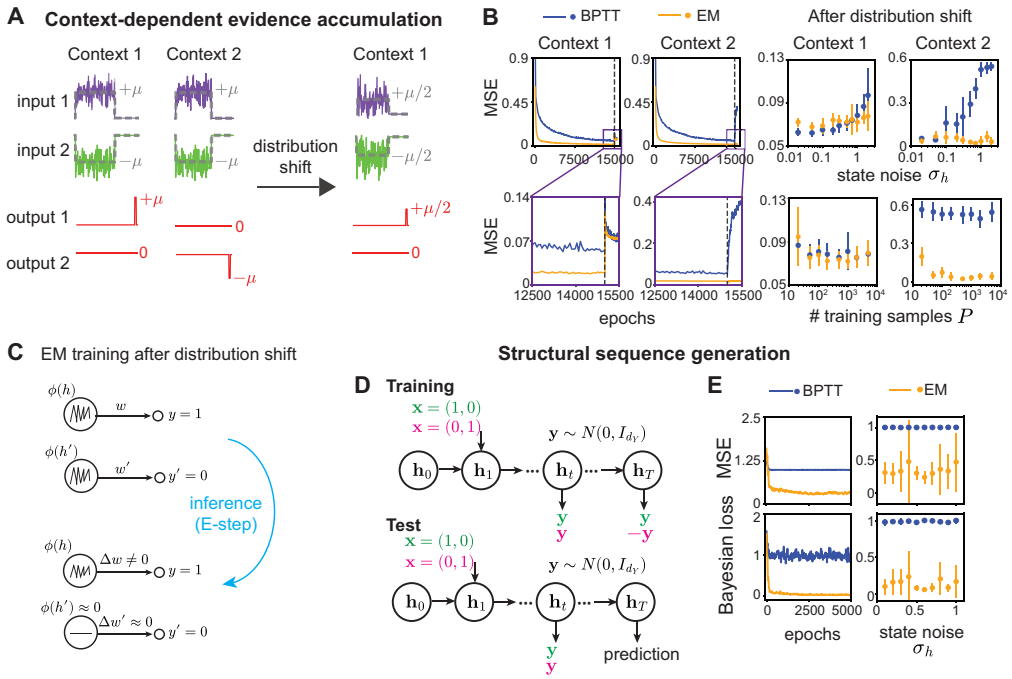


Figure 3: **EM outperforms BPTT for task learning that requires inference from the output.** A. Schematic of the two task phases, with a distribution shift introduced between them. B. After the shift, BPTT performance on context 2 deteriorates much more rapidly than EM. Here $\sigma_h = 1, P = 200$ are used in this panel. C.Illustration of the E-step after the shift. Each module projects selectively to one output unit. By using the target output to infer latent activity, EM suppresses spurious activations and preserves the modular weights. D. Schematics of the structural sequence generation task where the output may be ambiguous given only the input information. E. EM can successfully learn the task, while BPTT cannot, when measured by both MSE and Bayesian loss. Here $\sigma_h = 0.5, P = 5000$.

signal-to-noise ratio in context 1 inputs causes occasional noise-driven activation of neurons in other modules. Under EM, the E-step uses the target output to suppress this spurious activity: $p_W(\mathbf{h}'|\mathbf{y}, \mathbf{y}', \mathbf{x}) \propto p_W(\mathbf{h}'|\mathbf{x}) \exp[-(0 - W'\phi(\mathbf{h}'))^2/\sigma_y^2]$. When $\sigma_h \gg \sigma_y$, the likelihood strongly penalizes nonzero activity in the irrelevant module, suppressing $\phi(\mathbf{h}')$ and yielding negligible updates to W' in the M-step (Eq.(8) or Eq.(7)). EM therefore preserves the other context-specific modules. By contrast, BPTT updates weights using the noisy prior activity $\phi(\mathbf{h}')$, which can induce systematic changes in W' and erodes the modular specialization.

3.5 Structural sequence generation task

Lastly, we notice that the inference E-step in EM follows naturally from its objective: maximizing the marginal likelihood of the observed output (Eq.(5)), rather than fitting the best point estimates of the output as in BPTT (Eq.(2)). This suggests that EM should be advantageous in tasks where the input specifies only the high-level structure of the output distribution, rather than the exact target value.

To test this idea, we consider a structural sequence generation task (Fig. 3DE). Conditioned on the input, the network must generate a pair of outputs that are either identical or opposite. Importantly, during training, the same input may correspond to multiple valid outputs. Specifically, the outputs are sampled from a Gaussian distribution subject to the constraint specified by the input (Fig. 3D): on each trial, the two outputs are constrained to be either equal to or opposite of each other, as determined by the input. During testing, we provide the input together with one element of the output pair and ask the network to predict the other. We find that BPTT fails to learn this task, whereas EM succeeds when we evaluate the metric that uses the posterior distribution at the test data (Fig. 3E, Appendix G). This advantage persists across noise levels for both MSE and Bayesian loss. To understand why this happens, we show in Appendix F that, because the targets at both time points have zero mean, the

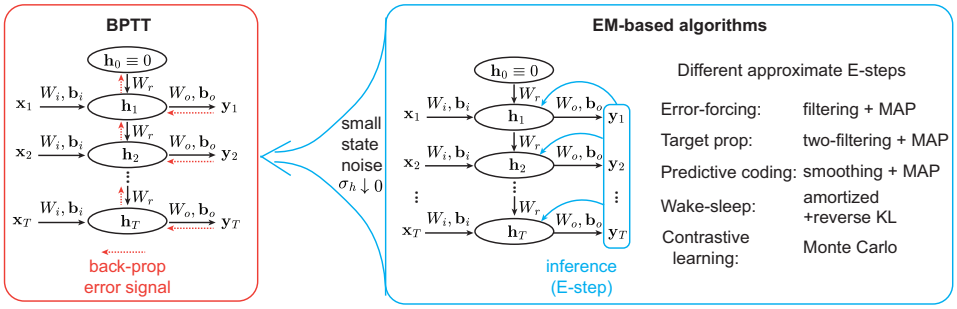


Figure 4: **EM offers a unified framework to formulate alternatives to BPTT.** Various BPTT alternatives can be viewed as the EM-like algorithm with different approximate E-steps. In the limit of small state noise, they all recover the standard BPTT algorithm.

BPTT weight update is in fact independent of the targets in the task. Consequently, BPTT cannot exploit the structure underlying these ambiguous targets. Therefore, EM-trained stochastic RNNs can use output-conditioned inference to exploit structures in ambiguous and variable training data during task learning and prediction, an ability that standard roll-out BPTT lacks.

4 EM-BASED ALGORITHM AS A PRINCIPLED ROUTE TO BPTT ALTERNATIVES

While BPTT remains one of the most widely used approaches for training neural networks, several alternatives have been proposed (Bengio & Frasconi, 1993; Engelken, 2023; Hess et al., 2023; Sağtekin et al., 2025). In this section, we suggest that many of these BPTT alternatives can be interpreted as instances of an EM-like algorithm (Fig. 4).

Different algorithms arise from different approximations q_*^k to the posterior $p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)$ in the E-step. Combined with gradient-based updates in the M-step (Eq.(8)), these choices recover a range of existing alternatives to BPTT for training noisy RNNs, as shown below. In particular, the results from the previous section extend naturally to these methods, implying that they all reduce to standard BPTT in the small-noise limit (Fig. 4) and may outperform BPTT in the three task regimes identified above at finite noise levels.

Error forcing: The distribution $q_*^k(\mathbf{h}_{1:T}^k)$ can be approximated by applying Bayesian filtering to the latent states conditioned on the observations, and then taking a maximum-a-posteriori (MAP) estimate at each filtering step (Appendix B.1). Specifically, at each filtering time step t, the algorithm computes the most probable latent state $\mathbf{h}_t$ given observations up to time t, and uses this MAP estimate to update the weights. After applying the reparameterization trick, this recovers the error forcing algorithm (Sağtekin et al., 2025), which is also related to the generalized teacher forcing (Doya, 2000; Hess et al., 2023) algorithm.

Target propagation: $q_*^k(\mathbf{h}_{1:T}^k)$ can be obtained using the two-filter smoothing procedure (Cappé et al., 2005; Chopin et al., 2020), followed by MAP estimate for each filter (Appendix B.2). The forward filter computes the most probable latent state from the RNN dynamics (Eq. (1)), while the backward filter computes the latent state that is most consistent with the network outputs (i.e., the targets). The weight update in Eq. (8) is then based on the local discrepancy between the two MAP estimates at each time step. This procedure recovers the target-propagation algorithm (Bengio, 2014). Notably, to our knowledge, this interpretation of target propagation as a two-filter smoothing procedure has not been suggested in previous work.

Predictive coding: The MAP estimate $q_*^k(\mathbf{h}^k) = \delta(\mathbf{h}^k - \mathbf{h}_*^k)$ can be obtained by directly minimizing the E-step objective (Eq.(6)) via gradient descent (Appendix B.3). The weight update (Eq.(8)) then reduces to a delta rule evaluated at this MAP estimate. This recovers predictive coding algorithms for RNNs (Spratling, 2017; Salvatori et al., 2023).

Wake-sleep algorithm: When the variational distribution $q^k(\mathbf{h}^k)$ is parameterized by a separate neural network whose parameters are optimized to minimize the reverse KL divergence $\mathrm{KL}[p_{W^n}(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)\|q^k(\mathbf{h}^k)]$, the resulting EM algorithm recovers the wake–sleep algorithm (Hin-

ton et al., 1995b;a; Dayan, 2000) (Appendix B.4). Although the wake–sleep algorithm is typically studied in unsupervised settings, here we derive it from the EM algorithm in a supervised setting for stochastic RNNs (Eq.(1)).

Contrastive-based learning: Contrastive-based algorithms (Xie & Seung, 2003; Movellan, 1991; Hinton, 2002) alternate between a clamped phase, where inputs/outputs are fixed to data, and a free phase, where only inputs are clamped. Updates are computed by the gradient difference, with latent states typically estimated by Monte Carlo sampling. When the complete-data log-likelihood $\ln p_W(\mathbf{y}^k_\Theta, \mathbf{h}^k|\mathbf{x}^k)$ is used as the contrastive energy (Appendix B.5), the resulting update recovers the GEM algorithm (Eq.(8)).

Taken together, these results show that many alternatives to BPTT can be interpreted as different approximations to minimizing the EM objective $L_{EM}(W)$. These relationships between EM and the algorithms discussed above are summarized in Fig. 4. This unified perspective also suggests a principled way to improve existing BPTT alternatives by using more accurate posterior inference in the E-step. We demonstrate this by adapting a recently developed natural-gradient-based variational inference method, SING (Hu et al., 2026), to the delayed-XOR task studied in the previous section. This improved inference method outperforms all of the methods considered above, including prior-rollout BPTT, error forcing, and Gaussian smoothing (Fig. A.1). Thus, this EM perspective opens a direct route for translating advances in state-space-model literature into improved alternatives to BPTT for task-optimization of stochastic RNNs.

5 Discussion and future work

We studied the relationship between EM-based training and BPTT for stochastic RNNs. The two methods coincide in the small-noise limit, but diverge when noise is non-negligible. In this regime, EM is found to preserve input information over long temporal horizons and leverages inference from the output (via the E-step) to support learning and prediction from noisy and ambiguous data. We therefore identified task families where such an explicit probabilistic treatment of the learning problem can be beneficial and training with EM outperforms BPTT.

At the same time, EM is not uniformly superior over BPTT. While maximum-likelihood methods (including the EM algorithm with exact E-step) asymptotically achieve the minimum MSE among all unbiased predictors (Lehmann & Casella, 1998; Keener, 2010), they can nevertheless exhibit a larger MSE than BPTT, as BPTT learns a biased predictor (Fig. 1). This contrast highlights the different roles of the two algorithms: EM-based methods are better suited for recovering the ground-truth parameters and mechanisms, whereas BPTT focuses on minimizing prediction error without explicitly modeling the underlying mechanisms. Although this behavior is expected when BPTT successfully finds a global minimizer of the loss, its optimization dynamics can be less effective or robust than EM in certain scenarios (Fig. 2). These observations point to a promising direction for future work: developing hybrid training schemes that leverage EM to obtain a good initialization and then switch to BPTT for refinement, potentially achieving lower MSE than either method alone.

EM separates inference (E-step) from learning (M-step), paralleling many brain-inspired learning algorithms (Friston, 2003; Spratling, 2017; Wang et al., 2024). These approaches often use MAP-style inference, yielding local weight updates, whereas our method uses full distributional information. Although we focus here on performance rather than biological realism, future work could explore how such distributional information might be realized into local learning rules (Lillicrap et al., 2020).

We interpret several existing alternatives to BPTT as EM algorithms with different approximate E-steps, showing how different inference schemes induce different learning rules. Although our experiments focus on Gaussian approximate E-step, future work could systematically study how other approximations affect performance and robustness, including amortized inference trained by optimizing the ELBO. Differences across algorithms may also depend on factors such as batch size, initialization, and the efficiency of exploring RNN parameter space through sampling-based rollouts.

Finally, our comparison focuses on performance across task regimes rather than practical scaling. Full posterior inference in EM is typically more expensive than BPTT (Appendix A), but EM also allows structured variational approximations in the E-step, such as low-rank, sparse, or modular forms (Pals et al., 2024; Murphy, 2023), which may reduce cost and improve parallelization. Developing scalable EM-based methods for large stochastic RNNs is therefore an important direction for future work.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

The majority of this work was written before the release of GPT 5.5. We used generative AI tools for providing feedback on the manuscript and providing the critical ingredients for mathematical claims in Appendix D.2.4,E and F. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, assisting in the writing of proofs, proposing or refine hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, interpreting results, assisting with translation and cleaning or reformating dataset are not applicable to this work. Additionally, we used generative AI tools for editing the paper to improve readability. We have reviewed all AI-assisted work. We checked all the AI-written texts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Nick Alonso, Beren Millidge, Jeffrey Krichmar, and Emre O Neftci. A theoretical framework for inference learning. *Advances in Neural Information Processing Systems*, 35:37335–37348, 2022.

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Yoshua Bengio and Paolo Frasconi. Credit assignment through time: Alternatives to backpropagation. *Advances in neural information processing systems*, 6, 1993.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pp. 1183–1188. IEEE, 1993.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Colin Bredenberg and Cristina Savin. Desiderata for normative models of synaptic plasticity, 2023. URL `https://arxiv.org/abs/2308.04988`.

Alexander Camuto, Matthew Willetts, Umut Simsekli, Stephen J Roberts, and Chris C Holmes. Explicit regularisation in gaussian noise injections. *Advances in Neural Information Processing Systems*, 33:16603–16614, 2020.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in hidden Markov models*. Springer, 2005.

Nicolas Chopin, Omiros Papaspiliopoulos, et al. *An introduction to sequential Monte Carlo*, volume 4. Springer, 2020.

Peter Dayan. Helmholtz machines and wake-sleep learning. *Handbook of Brain Theory and Neural Network. MIT Press, Cambridge, MA*, 44(0):1–12, 2000.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1): 1–22, 1977.

Kenji Doya. Bifurcations in the learning of recurrent neural networks. *Proc. IEEE Int. Symp. Circuits and Systems*, 01 2000. doi: 10.1109/ISCAS.1992.230622.

Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

Daniel Durstewitz, Georgia Koppe, and Max Ingo Thurm. Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nature Reviews Neuroscience*, 24(11): 693–710, 2023.

Rainer Engelken. Gradient flossing: Improving gradient descent through dynamic control of jacobians. *Advances in Neural Information Processing Systems*, 36:10412–10439, 2023.

A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Karl Friston. Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352, 2003.

Zoubin Ghahramani and Sam Roweis. Learning nonlinear dynamical systems using an em algorithm. *Advances in neural information processing systems*, 11, 1998.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Florian Hess, Zahra Monfared, Manuel Brenner, and Daniel Durstewitz. Generalized teacher forcing for learning chaotic dynamics. *arXiv preprint arXiv:2306.04406*, 2023.

G Hinton, Peter Dayan, Ava To, and Radford Neal. The helmholtz machine through time. In *ICANN-95*, pp. 483–490, 1995a.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995b.

Amber Hu, Henry Smith, and Scott Linderman. Sing: Sde inference via natural gradients. *Advances in Neural Information Processing Systems*, 38:153894–153947, 2026.

Kam-Chuen Jim, C Lee Giles, and Bill G Horne. An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on neural networks*, 7(6):1424–1438, 1996.

W Jeffrey Johnston and Stefano Fusi. Modular representations emerge in neural networks trained to perform context-dependent tasks. *bioRxiv*, 2024.

Robert W Keener. *Theoretical statistics: Topics for a core course*. Springer Science & Business Media, 2010.

Georgia Koppe, Hazem Toutounji, Peter Kirsch, Stefanie Lis, and Daniel Durstewitz. Identifying nonlinear dynamical systems via generative recurrent neural networks with applications to fmri. *PLoS computational biology*, 15(8):e1007263, 2019.

Erich Leo Lehmann and George Casella. *Theory of point estimation*. Springer, 1998.

Michael Lepori, Thomas Serre, and Ellie Pavlick. Break it down: Evidence for structural compositionality in neural networks. *Advances in Neural Information Processing Systems*, 36:42623–42660, 2023.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Soon Hoe Lim, N Benjamin Erichson, Liam Hodgkinson, and Michael W Mahoney. Noisy recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:5124–5137, 2021.

Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020.

Sheng Ma and Chuanyi Ji. Fast training of recurrent networks based on the em algorithm. *IEEE Transactions on Neural Networks*, 9(1):11–26, 1998.

Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2008.

Thomas P Minka. Expectation propagation for approximate bayesian inference. *arXiv preprint arXiv:1301.2294*, 2013.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pp. 10–17. Elsevier, 1991.

Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

Matthijs Pals, A Erdem Sağtekin, Felix Pei, Manuel Gloeckler, and Jakob H Macke. Inferring stochastic low-rank recurrent neural networks from neural data. *Advances in Neural Information Processing Systems*, 37:18225–18264, 2024.

Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Tapani Raiko and Matti Tornio. Variational bayesian learning of nonlinear hidden state-space models for model predictive control. *Neurocomputing*, 72(16-18):3704–3712, 2009.

Nuttida Rungratsameetaweemana, Robert Kim, Thiparat Chotibut, and Terrence J Sejnowski. Random noise promotes slow heterogeneous synaptic dynamics important for robust working memory computation. *Proceedings of the National Academy of Sciences*, 122(3):e2316745122, 2025.

A Erdem Sağtekin, Colin Bredenberg, and Cristina Savin. Error forcing in recurrent neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=xVI8g50Qfk.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

Tommaso Salvatori, Ankur Mali, Christopher L Buckley, Thomas Lukasiewicz, Rajesh PN Rao, Karl Friston, and Alexander Ororbia. Brain-inspired computational intelligence via predictive coding. *arXiv preprint arXiv:2308.07870*, 13, 2023.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

Galit Shmueli. To explain or to predict? *Statistical science*, pp. 289–310, 2010.

Michael W Spratling. A review of predictive coding algorithms. *Brain and cognition*, 112:92–97, 2017.

David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

David Sussillo, Mark M Churchland, Matthew T Kaufman, and Krishna V Shenoy. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience*, 18(7): 1025–1033, 2015.

Lev S Tsimring. Noise in biology. *Reports on Progress in Physics*, 77(2):026601, 2014.

Jan Unkelbach, Sun Yi, and Jürgen Schmidhuber. An em based training algorithm for recurrent neural networks. In *International Conference on Artificial Neural Networks*, pp. 964–974. Springer, 2009.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Bin Wang, Nicholas J Audette, David M Schneider, and Johnatan Aljadeff. Desegregation of neuronal predictive processing. *bioRxiv*, 2024.

Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

A EM UPDATE FOR NOISY RNNS

Unless specified, Gaussian smoothing (Särkkä & Svensson, 2023) is used in the E-step [Eq.(6)] in all the simulations (Figure 1-3). Specifically, this is equivalent to approximating the RNN nonlinear dynamics [Eq.(1)] using Gaussian moment matching or by minimizing the reverse KL divergence. Note that this E-step takes into account the effect of noise compared to the MAP-based BPTT estimate and, at the same time, has no sampling issues compared to the Monte Carlo-based BPTT alternatives.

The M-step update is derived as follows. The joint distribution in the M-step objective for noisy RNN [Eq.(1)] can be written as

$$\begin{aligned}\ln p_W(\mathbf{y}^k_\Theta, \mathbf{h}^k|\mathbf{x}^k) = &-\frac{1}{2\sigma_y^2}\sum_{t\in\Theta}(\mathbf{y}^k_t - W_o\phi(\mathbf{h}^k_t) - \mathbf{b}_o)^2 \\ &-\frac{1}{2\alpha\sigma_h^2}\sum_{t=1}^{T}\left[\mathbf{h}^k_t - (1-\alpha)\mathbf{h}^k_{t-1} - \alpha W_r\phi(\mathbf{h}^k_{t-1}) - \alpha(W_i\mathbf{x}^k_t + \mathbf{b}_i)\right]^2 + \text{const.}\end{aligned} \tag{11}$$

For notational simplicity, we will assume that the input bias is absorbed into the weight matrix by appending a constant 1 to the input vector $\mathbf{x}_t$. For numerical stability, we also add weak l_2-norm weight regularization to the objective above. The resulting function is a quadratic function of the parameters and can be optimized analytically. We denote the corresponding sufficient statistics as

$$\begin{aligned}
S_{y\phi} &= \sum_{k=1}^{P}\sum_{t\in\Theta}\mathbf{y}^k_t\mathbb{E}_{q^k_*}\left[\phi(\mathbf{h}^k_t)^T\right], & S_y &= \sum_{k=1}^{P}\sum_{t\in\Theta}\mathbf{y}^k_t, \\
S_{\phi\phi,\Theta} &= \sum_{k=1}^{P}\sum_{t\in\Theta}\mathbb{E}_{q^k_*}[\phi(\mathbf{h}^k_t)\phi(\mathbf{h}^k_t)^T], & S_{\phi,\Theta} &= \sum_{k=1}^{P}\sum_{t\in\Theta}\mathbb{E}_{q^k_*}[\phi(\mathbf{h}^k_t)], \\
S_{h\phi,next} &= \sum_{k=1}^{P}\sum_{t=1}^{T-1}\mathbb{E}_{q^k_*}[\mathbf{h}_{t+1}\phi(\mathbf{h}_t)^T], & S_{h\phi} &= \sum_{k=1}^{P}\sum_{t=1}^{T}\mathbb{E}_{q^k_*}[\mathbf{h}_t\phi(\mathbf{h}_t)^T], \\
S_{\phi x,next} &= \sum_{k=1}^{P}\sum_{t=1}^{T-1}\mathbb{E}_{q^k_*}[\phi(\mathbf{h}_t)]\mathbf{x}^T_{t+1}, & S_{hx,next} &= \sum_{k=1}^{P}\sum_{t=1}^{T-1}\mathbb{E}_{q^k_*}[\mathbf{h}_t]\mathbf{x}^T_{t+1}, \\
S_{hx} &= \sum_{k=1}^{P}\sum_{t=1}^{T}\mathbb{E}_{q^k_*}[\mathbf{h}_t]\mathbf{x}^T_t, & S_x &= \sum_{k=1}^{P}\sum_{t=1}^{T}\mathbf{x}_t\mathbf{x}^T_t, \\
S_{\phi\phi} &= \sum_{k=1}^{P}\sum_{t=1}^{T-1}\mathbb{E}_{q^k_*}[\phi(\mathbf{h}^k_t)\phi(\mathbf{h}^k_t)^T].
\end{aligned} \tag{12}$$

Then the exact M-step update [Eq.(7)] can be solved from the linear regression problem defined by Eq.(11)

$$\begin{aligned}
(W_o^{n+1}, \mathbf{b}_o^{n+1}) &= (S_{y\phi}, S_y)\begin{pmatrix}\lambda\sigma_y^2 I + S_{\phi\phi,\Theta} & S_{\phi,\Theta} \\ S^T_{\phi,\Theta} & \lambda\sigma_y^2 I + P|\Theta|\end{pmatrix}^{-1}, \\
(W_r^{n+1}, W_i^{n+1}) &= (S_{h\phi,next} - (1-\alpha)S_{h\phi}, S_{hx} - (1-\alpha)S_{hx,next})\begin{pmatrix}\lambda\sigma_h^2 I + \alpha S_{\phi\phi} & \alpha S_\phi \\ \alpha S_\phi^T & \lambda\sigma_h^2 I + \alpha S_x\end{pmatrix}^{-1},
\end{aligned} \tag{13}$$

where $\lambda > 0$ is the regularization strength and $|\Theta|$ is the number of observation points. In simulations, we maintain a running average of the sufficient statistics [Eq.(12)] from the mini-batches (See Appendix G).

Finally, we comment on the computational complexity of this EM algorithm and BPTT. In the overparameterized regime (network size $N \gg$ the input/output dimensions), the memory and time complexity for EM are $O(TN^2)$ and $O(TN^3)$, respectively. EM has worse than the memory and time complexity of BPTT, which are $O(TN)$ and $O(TN^2)$, respectively. The cubic cost in EM arises from matrix inversions in the E- and M-steps and could be reduced or approximated in other EM variants.

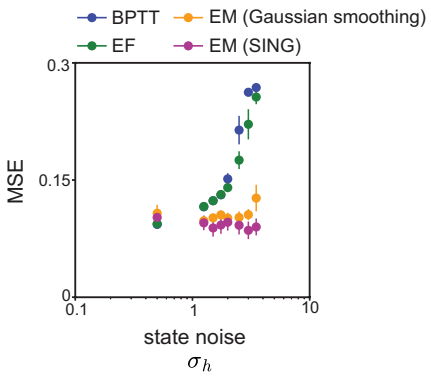


Figure A.1: Comparison of different E-step approximations for delayed XOR task in Section 3.3. The natural-gradient-based EM methods outperform other methods for large noise levels. $\alpha = 0.2$, $\Delta T = 5$ and $P = 100$.

B Deriving other algorithms from EM

We note that the weight update in GEM [Eq.(8)] for a mini-batch is the sum of the contributions from each data point. Therefore, it suffices to consider each data point's contribution and simply sum them. In the following derivations, we will omit the data index k.

B.1 Error forcing

Since the RNN dynamics are Markov [Eq.(1)], the posterior distribution of the neural activations can be factorized as

$$p_W(\mathbf{h}_{1:T}|\mathbf{y}_\Theta, \mathbf{x}_{1:T}) \propto \prod_{t=1}^{T} p_W(\mathbf{h}_t|\mathbf{h}_{1:t-1}, \mathbf{y}_\Theta, \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{y}_{\Theta_{\geq t}}, \mathbf{x}_{t:T}). \tag{14}$$

Here $\Theta_{\geq t} = \Theta \cap \{t, t+1.., T\}$ contains all the time steps in Θ that are greater than or equal to t. We can ignore the normalization factor since we will do MAP estimation on the neural activities.

We first approximate each term in the product, $p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{y}_{\Theta_{\geq t}}, \mathbf{x}_{t:T})$, by the filtering distribution, $p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{y}_t, \mathbf{x}_{1:t})$ for $t \in \Theta$, or $p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_{1:t})$ for $t \notin \Theta$, yielding

$$\approx \left[\prod_{t\in\Theta} p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{y}_t, \mathbf{x}_{1:t})\right] \left[\prod_{t\notin\Theta} p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_{1:t})\right]. \tag{15}$$

Roughly speaking, the above filtering approximation ignores the information of the latent variable $\mathbf{h}_t$ at time t from future times $\mathbf{y}_s (s > t)$.

Note that if $t \in \Theta$, the corresponding term in the above product has the form

$$\begin{aligned} p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{y}_t, \mathbf{x}_{1:t}) &\propto p_W(\mathbf{y}_t|\mathbf{h}_t) \cdot p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_{1:t}) \\ &= e^{-\frac{1}{2\sigma_y^2}(\mathbf{y}_t - W_o\phi(\mathbf{h}_t) - \mathbf{b}_o)^2 - \frac{1}{2\alpha\sigma_h^2}[\mathbf{h}_t - (1-\alpha)\mathbf{h}_{t-1} - \alpha W_r\phi(\mathbf{h}_{t-1}) - \alpha(W_i\mathbf{x}_t + \mathbf{b}_i)]^2}, \end{aligned}$$

or if $t \notin \Theta$,

$$p_W(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{x}_{1:t}) \propto e^{-\frac{1}{2\alpha\sigma_h^2}[\mathbf{h}_t - (1-\alpha)\mathbf{h}_{t-1} - \alpha W_r\phi(\mathbf{h}_{t-1}) - \alpha(W_i\mathbf{x}_t + \mathbf{b}_i)]^2}.$$

Then we can approximate each distribution by its mode iteratively. Start at $\mathbf{h}_0^* \equiv \mathbf{0}$. For $t \in \Theta$, the mode is given by

$$\mathbf{h}_t^* = \underset{\mathbf{h}}{\text{argmin}} \left\{ \frac{1}{2\sigma_y^2}(\mathbf{y}_t - W_o\phi(\mathbf{h}) - \mathbf{b}_o)^2 + \frac{1}{2\alpha\sigma_h^2}\left[\mathbf{h} - (1-\alpha)\mathbf{h}_{t-1}^* - \alpha W_r\phi(\mathbf{h}_{t-1}^*) - \alpha(W_i\mathbf{x}_t + \mathbf{b}_i)\right]^2 \right\}, \tag{16}$$

and for $t \notin \Theta$, the mode is

$$\begin{aligned}\mathbf{h}_t^* &= \underset{\mathbf{h}}{\text{argmin}} \left[\mathbf{h} - (1-\alpha)\mathbf{h}_{t-1}^* - \alpha W_r \phi(\mathbf{h}_{t-1}^*) - \alpha(W_i \mathbf{x}_t + \mathbf{b}_i)\right]^2 \\ &= (1-\alpha)\mathbf{h}_{t-1}^* + \alpha W_r \phi(\mathbf{h}_{t-1}^*) + \alpha(W_i \mathbf{x}_t + \mathbf{b}_i). \end{aligned} \tag{17}$$

So, the optimal q_* [Eq.(6)] is $q_*(\mathbf{h}_{1:T}) = \prod_{t=1}^T \delta(\mathbf{h}_t - \mathbf{h}_t^*)$.

Intuitively, the iterative inference procedure through Eq.(16)-(17) has a natural interpretation: if there is no observation at time t, the neural activation is updated according to the average of the original RNN dynamics [Eq.(1)]. If there is observation at time t, the neural activity should take into account of the observation and should be "forced" to those states that are consistent with the observation $\mathbf{y}_t$ [Eq.(16)].

Unlike Eq.(17), Eq.(16) does not have an closed-form solution for the RNN model we considered here [Eq.(1)]. But if we assume that the observation is a linear function of the neural activity $\mathbf{y}_t = W_o \mathbf{h}_t + \mathbf{b}_o$, then Eq.(16) reduces a linear regression problem and can be solved exactly. The resulting update equation for $\mathbf{h}_t^*$ is known as error forcing (Sağtekin et al., 2025), which, for its deterministic version, is equivalent to generalized teacher forcing (Hess et al., 2023) when the output dimension is greater than the network size.

Finally, a variant of the reparametrization trick is applied during the M-step to counteract the fact that future information is neglected during the E-step, since reparametrization enables the use of future information through BPTT. This is achieved by expressing the M-step objective in terms of the internal noise $\boldsymbol{\xi}_{1:T}$, rather than the neural activations $\mathbf{h}_{1:T}$. Consequently, during the M-step the posterior remains a function of the recurrent weights, rather than a constant, allowing gradients to flow. This yields the same weight update as in the original teacher/error-forcing literature(Sağtekin et al., 2025; Hess et al., 2023).

B.2 Target propagation

The posterior distribution $p_W(\mathbf{h}_{1:T}|\mathbf{y}_\Theta, \mathbf{x}_{1:T})$ can be obtained through the two-filter smoothing algorithm (Chopin et al., 2020). The forward filtering consists of the prediction and update steps (Särkkä & Svensson, 2023),

$$\begin{aligned} \text{Prediction: } p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t-1}}, \mathbf{x}_{1:t}) &= \int d\mathbf{h}_{t-1} p_W(\mathbf{h}_t|\mathbf{h}_{t-1}) p_W(\mathbf{h}_{t-1}|\mathbf{y}_{\Theta_{\le t-1}}, \mathbf{x}_{1:t-1}), \\ \text{Update: } p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t}}, \mathbf{x}_{1:t}) &\propto p_W(\mathbf{y}_t|\mathbf{h}_t) p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t-1}}, \mathbf{x}_{1:t}), \quad \text{if } t \in \Theta. \end{aligned} \tag{18}$$

Starting from $p_W(\mathbf{h}_0|\mathbf{y}_0, \mathbf{x}_0) \equiv p_W(\mathbf{h}_0) = \delta(\mathbf{h}_0)$, the above iteration allows to compute the filtering distribution $p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t}}, \mathbf{x}_{1:t})$ iteratively.

The backward filter follows from the following iteration,

$$\begin{aligned} \text{Update: } p_W(\mathbf{y}_{\Theta_{\ge t+1}}|\mathbf{h}_t) &= \int p_W(\mathbf{y}_{\Theta_{\ge t+1}}|\mathbf{h}_{t+1}) p_W(\mathbf{h}_{t+1}|\mathbf{h}_t) d\mathbf{h}_{t+1}, \\ \text{Filtering: } p_W(\mathbf{y}_{\Theta_{\ge t}}|\mathbf{h}_t) &= p_W(\mathbf{y}_t|\mathbf{h}_t) p_W(\mathbf{y}_{\Theta_{\ge t+1}}|\mathbf{h}_t), \quad \text{if } t \in \Theta. \end{aligned} \tag{19}$$

Start from $p_W(\mathbf{y}_{\Theta_{\ge T}}|\mathbf{h}_T) = p_W(\mathbf{y}_T|\mathbf{h}_T)$ for $T \in \Theta$ or $= 1$ for $T \notin \Theta$, this iteratively computes $p_W(\mathbf{y}_{\Theta_{\ge t}}|\mathbf{h}_t)$.

Finally, the posterior distribution is given by

$$p_W(\mathbf{h}_{t+1}, \mathbf{h}_t|\mathbf{y}_\Theta) \propto p_W(\mathbf{y}_{\Theta_{\ge t+1}}|\mathbf{h}_{t+1}) p_W(\mathbf{h}_{t+1}|\mathbf{h}_t) p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t}}). \tag{20}$$

Note that here we focus on the joint (posterior) distribution of $\mathbf{h}_{t+1}, \mathbf{h}_t$ since this is the one involved in computing the weight update in GEM [Eq.(8)].

We will derive the target propagation algorithm for deep neural networks where there is only output at the last timestep and input at the first time step (Bengio, 2014), $\Theta = \{T\}$. We approximate the forward filtering and backward filtering distributions by a Dirac-delta measure at their maximum, $p_W(\mathbf{h}_t|\mathbf{y}_{\Theta_{\le t}}, \mathbf{x}_{1:t}) = p_W(\mathbf{h}_t|\mathbf{x}_{1:t}) \approx \delta(\mathbf{h}_t - \mathbf{h}_t^f)$ and $p_W(\mathbf{y}_{\Theta_{\ge t}}|\mathbf{h}_t) \approx \delta(\mathbf{h}_t - \mathbf{h}_t^b)$ for $t = 1, 2, 3, .., T$. Then the posterior [Eq.(20)] becomes $p_W(\mathbf{h}_{t+1}, \mathbf{h}_t|\mathbf{y}_\Theta) \propto \delta(\mathbf{h}_{t+1} - \mathbf{h}_{t+1}^b)\delta(\mathbf{h}_t - \mathbf{h}_t^f)$.

Plugging this into Eq.(18), we get

$$\mathbf{h}_t^f = \underset{\mathbf{h}}{\text{argmax}}\, p_W(\mathbf{h}|\mathbf{h}_{t-1}^f) = (1-\alpha)\mathbf{h}_{t-1}^f + \alpha W_r \phi(\mathbf{h}_{t-1}^f) + \alpha(W_i \mathbf{x}_t + \mathbf{b}_i)\delta_{t1}, \tag{21}$$

for $t = 1, 2.., T$.

Similarly, Eq.(19) becomes

$$\mathbf{h}_t^b = \underset{\mathbf{h}}{\text{argmax}}\, p_W(\mathbf{h}_{t+1}^b|\mathbf{h}) = \underset{\mathbf{h}}{\text{argmin}}\, \left[\mathbf{h}_{t+1}^b - (1-\alpha)\mathbf{h} - \alpha W_r \phi(\mathbf{h})\right]^2. \tag{22}$$

for $t = 1, 2, 3, .., T-1$ and

$$\mathbf{h}_T^b = \underset{\mathbf{h}}{\text{argmin}}\, \left[\mathbf{y}_T - W_o \phi(\mathbf{h})\right]^2. \tag{23}$$

Substituting these results in the GEM update [Eq.(8)], we get the standard target propagation update

$$\begin{aligned}
\Delta W_o &\propto (\mathbf{y}_T - W_o \phi(\mathbf{h}_T^f))\phi(\mathbf{h}_T^f)^T, \\
\Delta W_r &\propto \sum_{t=1}^{T-1} \left[\mathbf{h}_{t+1}^b - (1-\alpha)\mathbf{h}_t^f + \alpha W_r \phi(\mathbf{h}_t^f)\right] \phi(\mathbf{h}_t^f)^T, \\
\Delta W_i &\propto \left(\mathbf{h}_1^b - W_i \mathbf{x}_1\right) \mathbf{x}_1^T.
\end{aligned}$$

B.3 Predictive coding

If we directly approximate q_* by a Dirac-delta distribution $q_*(\mathbf{h}_{1:T}) \approx \delta(\mathbf{h}_{1:T} - \mathbf{h}_{1:T}^*)$, the ELBO objective becomes a function of $\mathbf{h}_{1:T}^*$ (ignoring the entropy term),

$$\begin{aligned}
E_W(\mathbf{h}_{1:T}^*) \equiv -\text{ELBO}[q_*, W] &= -\ln p_W(\mathbf{y}_\Theta, \mathbf{h}_{1:T}^*|\mathbf{x}_{1:T}) \\
&= \frac{1}{2\sigma_y^2} \sum_{t\in\Theta} (\mathbf{y}_t - W_o \phi(\mathbf{h}_t^*) - \mathbf{b}_o)^2 + \frac{1}{2\alpha\sigma_h^2} \sum_{t=1}^{T} \left[\mathbf{h}_t^* - (1-\alpha)\mathbf{h}_{t-1}^* - \alpha W_r \phi(\mathbf{h}_{t-1}^*) - \alpha(W_i \mathbf{x}_t + \mathbf{b}_i)\right]^2 \\
&+ \text{const.},
\end{aligned} \tag{24}$$

As $\mathbf{h}^*$ should optimize ELBO, this can be obtained by running gradient descent over $E(\mathbf{h}_{1:T}^*)$,

$$\frac{d\mathbf{h}_t^*}{ds} \propto -\frac{\partial E(\mathbf{h}_{1:T}^*)}{\partial \mathbf{h}_t^*}, \tag{25}$$

Then the weight update in the M-step also optimizes ELBO, and becomes

$$\frac{dW}{ds} \propto -\frac{\partial E_W(\mathbf{h}_{1:T}^*)}{\partial W}, \tag{26}$$

for the input, recurrent, and output weights. This is known as the predictive coding algorithm (Alonso et al., 2022; Spratling, 2017).

B.4 Wake-sleep algorithm

As mentioned in the main text, the standard EM algorithm alternates between an E-step and an M-step. To interpret the wake–sleep algorithm through the lens of EM, it is helpful to start with the M-step, which corresponds to the wake phase.

B.4.1 M-step (Wake phase)

In the standard M-step, we maxmize the ELBO$[q_*, W]$, with respect to the generative parameters W, treating the approximate posterior q_* (computed in the previous iteration) as fixed. It is easier to see from the ELBO form in Eq. (31), the second term does not depend on W. Therefore, maximizing the ELBO with respect to W is equivalent to maximizing the expected joint log-likelihood $\sum_{k=1}^{P} \mathbb{E}_{q_*^k(\mathbf{h}^k)} \left[\ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k \mid \mathbf{x}^k)\right]$. In standard EM, one typically maintains a separate approximate posterior $q_*^k(\mathbf{h}^k)$ for each data point k.

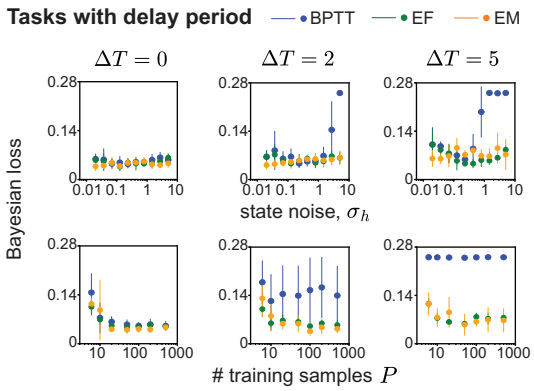


Figure B.2: Bayesian loss for EM and BPTT trained network for delayed XOR task in Section 3.3. Similar trends hold for Bayesian loss as MSE in Fig. 2. Unless specified, we show results for $\sigma_h = 1.5$ and $P = 200$.

In wake–sleep, instead of fitting P separate posteriors, we use a single recognition network with shared parameters (amortization). Concretely, we define an amortized posterior $q_*^\Gamma(\mathbf{h}^k \mid \mathbf{x}^k, \mathbf{y}_\Theta^k)$, where Γ denotes recognition-network parameters. The wake-phase update of the generative model then optimizes $\sum_{k=1}^P \mathbb{E}_{q_*^\Gamma(\mathbf{h}^k|\mathbf{x}^k,\mathbf{y}_\Theta^k)} \left[\ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k \mid \mathbf{x}^k)\right]$. In other words, during the wake phase, the recognition network is driven by real data (in our supervised setting, both $\mathbf{x}^k$ and $\mathbf{y}_\Theta^k$) to infer latent variables $\mathbf{h}^k$, which are then used to update the generative model parameters W.

B.4.2 E-STEP (SLEEP PHASE)

In the standard E-step, we maximize ELBO$[q, W_*]$ with respect to q, treating the generative parameters W_* as fixed. From the ELBO form in Eq. (33), the first term does not depend on q. Hence, optimizing the E-step is equivalent to minimizing the forward KL divergence between the approximate posterior and the true model posterior: $\mathrm{KL}\left[q^\Gamma(\mathbf{h}^k \mid \mathbf{x}^k, \mathbf{y}_\Theta^k) \,\|\, p_W(\mathbf{h}^k \mid \mathbf{y}^k, \mathbf{x}^k)\right]$.

Wake–sleep introduces an additional approximation (first one was amortization): rather than minimizing this forward KL, it minimizes the reverse KL divergence: $\mathrm{KL}\left[p_W(\mathbf{h}^k \mid \mathbf{y}^k, \mathbf{x}^k) \,\|\, q^\Gamma(\mathbf{h}^k \mid \mathbf{x}^k, \mathbf{y}_\Theta^k)\right]$. Minimizing reverse KL turns out to more tractable than directly minimizing the forward KL (Bredenberg & Savin, 2023).

B.5 CONTRASTIVE-BASED LEARNING

We use the complete data likelihood as the contrastive energy. The weight update in contrastive learning (Movellan, 1991; Xie & Seung, 2003) is given by

$$\Delta W_{constrast} \propto \underbrace{\mathbb{E}_{p_W(\mathbf{h}_{1:T}|\mathbf{y}_\Theta,\mathbf{x}_{1:T})}\left[\frac{\partial}{\partial W}\ln p_W(\mathbf{y}_\Theta, \mathbf{h}_{1:T}|\mathbf{x}_{1:T})\right]}_{\text{clamped phase update}} - \underbrace{\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{\partial}{\partial W}\ln p_W(\mathbf{y}_\Theta, \mathbf{h}_{1:T}|\mathbf{x}_{1:T})\right]}_{\text{free phase update}}. \tag{27}$$

We note that the difference between the two terms is that the update in the clamped phase is averaged over the posterior distribution given the training data $(\mathbf{x}, \mathbf{y}_\Theta)$ while the update in the free phase is over the prior distribution for the corresponding variables.

Note that the above update has an additional term for the free phase compared to the GEM update. However, for the stochastic RNN model in Eq.(1), the free phase update is indeed zero. We illustrate this as follows (without loss of generality, we take $\alpha = 1$).

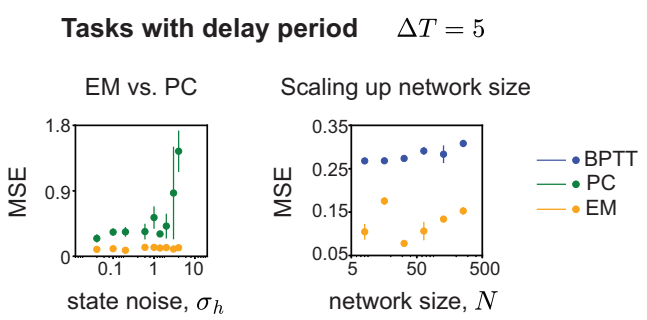


Figure B.3: (Left) MSE for EM and predictive coding (PC) trained network for delayed XOR task in Section 3.3. The results are for $\sigma_h = 1$ and $P = 200$. The performance of PC algorithm becomes worse as noise level increases (Right) MSE for EM and BPTT with different network sizes. Network trained with EM generally has smaller MSE error than BPTT across different sizes. Note that here for EM algorithms, we use the same hyperparameter values for network with different sizes for EM and BPTT, which are chosen at small network size $N = 8$.

For the recurrent weight, the free phase update is

$$\begin{aligned}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{\partial}{\partial W_r}\ln p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})\right] &= \sum_{t=0}^{T-1}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{(\mathbf{h}_{t+1}-W_r\phi(\mathbf{h}_t)-W_i\mathbf{x}_{t+1}-\mathbf{b}_i)\,\phi(\mathbf{h}_t)^T}{\sigma_h^2}\right]\\ &= \sum_{t=0}^{T-1}\mathbb{E}_{p_W(\mathbf{h}_t|\mathbf{x}_{1:T})}\left[\frac{\mathbb{E}_{p_W(\mathbf{h}_{t+1}|\mathbf{h}_t)}\left[\mathbf{h}_{t+1}-W_r\phi(\mathbf{h}_t)-W_i\mathbf{x}_{t+1}-\mathbf{b}_i\right]\phi(\mathbf{h}_t)^T}{\sigma_h^2}\right]\\ &= 0. \end{aligned} \tag{28}$$

The last line is because at each time step for the RNN dynamics [Eq.(1)], a zero-mean Gaussian noise is added.

For output weight, the free phase update is

$$\begin{aligned}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{\partial}{\partial W_{out}}\ln p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})\right] &= \sum_{t\in\Theta}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{(\mathbf{y}_t-W_o\phi(\mathbf{h}_t))\,\phi(\mathbf{h}_t)^T}{\sigma_y^2}\right]\\ &= \sum_{t\in\Theta}\mathbb{E}_{p_W(\mathbf{h}_t|\mathbf{x}_{1:T})}\left[\frac{\mathbb{E}_{p_W(\mathbf{y}_t|\mathbf{h}_t)}\left[\mathbf{y}_t-W_o\phi(\mathbf{h}_t)\right]\phi(\mathbf{h}_t)^T}{\sigma_h^2}\right]\\ &= 0. \end{aligned} \tag{29}$$

For input weight, the free phase update is then

$$\begin{aligned}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{\partial}{\partial W_i}\ln p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})\right] &= \sum_{t=0}^{T-1}\mathbb{E}_{p_W(\mathbf{y}_\Theta,\mathbf{h}_{1:T}|\mathbf{x}_{1:T})}\left[\frac{(\mathbf{h}_{t+1}-W_r\phi(\mathbf{h}_t)-W_i\mathbf{x}_t-\mathbf{b}_i)\,\mathbf{x}_t^T}{\sigma_h^2}\right]\\ &= \sum_{t=0}^{T-1}\mathbb{E}_{p_W(\mathbf{h}_t|\mathbf{x}_{1:T})}\left[\frac{\mathbb{E}_{p_W(\mathbf{h}_{t+1}|\mathbf{h}_t)}\left[\mathbf{h}_{t+1}-W_r\phi(\mathbf{h}_t)-W_i\mathbf{x}_t-\mathbf{b}_i\right]\phi(\mathbf{h}_t)^T}{\sigma_h^2}\right]\\ &= 0. \end{aligned} \tag{30}$$

C Relationship between EM and BPTT

We first show that the objectives for EM and BPTT can be obtained by substituting the posterior and prior distributions into ELBO. There are three ways to write ELBO

$$\begin{aligned}\text{ELBO}[q,W] &= \mathbb{E}_{q(\mathbf{h}^k)}\left[\ln p_W(\mathbf{y}^k_\Theta,\mathbf{h}^k|\mathbf{x}^k)\right] + H\left[q(\mathbf{h}^k)\right] &(31)\\ &= \mathbb{E}_{q(\mathbf{h}^k)}\left[\ln p_W(\mathbf{y}^k_\Theta|\mathbf{h}^k,\mathbf{x}^k)\right] - \text{KL}\left[q(\mathbf{h}^k)\|p_W(\mathbf{h}^k|\mathbf{x}^k)\right] &(32)\\ &= \ln p_W(\mathbf{y}^k_\Theta|\mathbf{x}^k) - \text{KL}\left[q(\mathbf{h}^k)\|p_W(\mathbf{h}^k|\mathbf{y}^k,\mathbf{x}^k)\right]. &(33)\end{aligned}$$

So if we plug the prior distribution $q = p_W(\mathbf{h}^k|\mathbf{x}^k)$ into the 2nd line [Eq.(32)], this gives the BPTT objective L_{BPTT}. If we plug the posterior distribution $q = p_W(\mathbf{h}^k|\mathbf{y}^k, \mathbf{x}^k)$ into the 3rd line [Eq.(33)], this gives the EM objective L_{EM}.

Now we show that the backpropagation and EM converge to each other when the state noise is small $\sigma_h \downarrow 0$.

The marginal evidence can be written as

$$p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k) = \int p_W(\mathbf{y}_\Theta|\mathbf{h}^k) p_W(\mathbf{h}^k|\mathbf{x}^k) d\mathbf{h}^k = \frac{1}{(2\pi\sigma_y^2)^{d_Y}} \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{x}^k)} \left[e^{-\frac{1}{2\sigma_y^2} \sum_{t\in\Theta} (\mathbf{y}_t^k - W_o \phi(\mathbf{h}_t^k) - \mathbf{b}_o)^2} \right]. \tag{34}$$

In the limit $\sigma_h \downarrow 0$, the prior distribution $p_W(\mathbf{h}^k|\mathbf{x}^k)$ converges weakly to the Dirac-delta measures $\delta(\mathbf{h}^k - \mathbf{h}^k(\mathbf{x}^k; W))$, where $\mathbf{h}^k(\mathbf{x}^k; W))$ is the function defined by the standard deterministic RNNs [Eq.(1)]. Therefore,

$$\begin{aligned} L_{EM}(W) = -\ln p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k) &= -\ln \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{x}^k)} \left[e^{-\frac{1}{2\sigma_y^2} \sum_{t\in\Theta} (\mathbf{y}_t^k - W_o \phi(\mathbf{h}_t^k) - \mathbf{b}_o)^2} \right] + \text{const.} \\ &\overset{\sigma_h \downarrow 0}{\to} \frac{1}{2\sigma_y^2} \sum_{t\in\Theta} (\mathbf{y}_t^k - W_o \phi(\mathbf{h}_t^k(\mathbf{x}^k; W)) - \mathbf{b}_o)^2 + \text{const.} \end{aligned} \tag{35}$$

Up to a constant, the last line is exactly the BPTT loss $L_{BPTT}(W)$ [Eq.(4)] when the state noise level goes to zero.

For the GEM algorithm [Eq.(8)], the parameter update is

$$\Delta W_{GEM} \propto \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)} \left[\frac{\partial \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)}{\partial W} \right]. \tag{36}$$

By Fisher's identity (Cappé et al., 2005),

$$\mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)} \left[\frac{\partial \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)}{\partial W} \right] = \frac{\partial \ln p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k)}{\partial W}. \tag{37}$$

So the right-hand side of Eq.(36) is the same as the weight derivative of the log-evidence, i.e. $-\partial_W L_{EM}$. When the state noise $\sigma_h \downarrow 0$, $L_{EM} \to L_{BPTT} + \text{const.}$, and thus it becomes the same as gradient descent on L_{BPTT}.

For the exact M-step update [Eq.(7)], since the complete data likelihood is a (convex) quadratic function of the parameters W [Eq.(11)], the parameter update in Eq.(13) is therefore the same as using the Newton method on the complete data likelihood

$$\begin{aligned} \Delta W_{EM} &\propto -\mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)} \left[\frac{\partial^2 \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)}{\partial^2 W} \right]^{-1} \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)} \left[\frac{\partial \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)}{\partial W} \right] \\ &\overset{\text{Eq.(37)}}{=} \mathbb{E}_{p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k)} \left[-\frac{\partial^2 \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k|\mathbf{x}^k)}{\partial^2 W} \right]^{-1} \frac{\partial \ln p_W(\mathbf{y}_\Theta^k|\mathbf{x}^k)}{\partial W} \end{aligned} \tag{38}$$

Here, we will think of the parameter W as a vector. The 2nd factor above is the same as the GEM update [Eq.(36)] and therefore converges to $-\partial_W L_{BPTT}$ in the small state noise limit.

On the other hand, the 1st factor above can be viewed as a preconditioning matrix, and it is positive semidefinite due to the quadratic form of the complete data likelihood [Eq.(11)].

To get the limit of it, we first use the Bayesian rule,

$$p_W(\mathbf{h}^k|\mathbf{y}_\Theta^k, \mathbf{x}^k) = \frac{p_W(\mathbf{y}_\Theta^k|\mathbf{h}^k) p_W(\mathbf{h}^k|\mathbf{x}^k)}{\int p_W(\mathbf{y}_\Theta^k|\mathbf{h}^k) p_W(\mathbf{h}^k|\mathbf{x}^k) d\mathbf{h}^k}. \tag{39}$$

So the posterior distribution will weakly converge to the prior distribution $p_W(\mathbf{h}^k|\mathbf{x}^k) = \delta(\mathbf{h}^k - \mathbf{h}^k(\mathbf{x}^k; W))$ in the small state noise limit. The preconditioning matrix becomes $\left[-\frac{\partial^2 \ln p_W(\mathbf{y}_\Theta^k, \mathbf{h}^k(\mathbf{x}^k;W)|\mathbf{x}^k)}{\partial^2 W} \right]^{-1}$, which is still a (nonzero) positive semidefinite matrix.

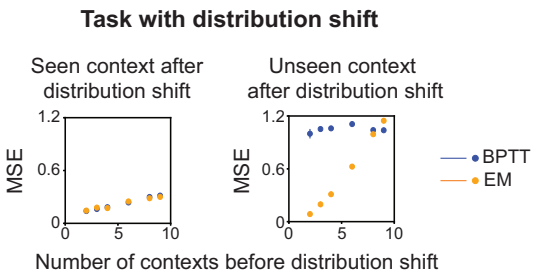


Figure C.4: MSE for EM and BPTT trained network for the task with distribution shift in Section 3.4. We increase the number of contexts during the initial training phase systematically. After the distribution shift, the network only sees a distribution-shifted version of one of the context but does not see other contexts. We test the network performance both on the seen and unseen contexts after the distribution shift. Although both algorithms perform similarly on the seen context, EM performs much better than BPTT on the unseen contexts because of its probabilistic inference mechanism. The results are for $\sigma_h = 1$ and $P = 200$.

Finally, we show that EM minimizes the Bayesian loss [Eq.(10)] with sufficient data and $\mathcal{D}_{test} = \mathcal{D}$. To see this, we consider the Bayesian loss of an arbitrary random function $f(\mathbf{x})$,

$$\mathbb{E}_{\mathcal{D}_{test}}\mathbb{E}_f\left[(\mathbf{y}_\Theta - f(\mathbf{x}))^2\right] \geq \mathbb{E}_{\mathcal{D}_{test}}\left[(\mathbf{y}_\Theta - \mathbb{E}_f f(\mathbf{x}))^2\right], \tag{40}$$

where we have used Jensen's inequality. The last line is optimized by the Bayesian predictor (Keener, 2010)

$$\begin{aligned} f_*(\mathbf{x}) &= \mathbb{E}_{p_{\mathcal{D}_{test}}(\mathbf{y}_\Theta|\mathbf{x})}\left[\mathbf{y}_\Theta\right] \\ &\approx \mathbb{E}_{p_W(\mathbf{y}_\Theta|\mathbf{x})}\left[\mathbf{y}_\Theta\right], \quad \text{if } \mathrm{KL}[p_{\mathcal{D}_{test}}(\mathbf{y}_\Theta|\mathbf{x})|p_W(\mathbf{y}_\Theta|\mathbf{x})] \approx 0. \end{aligned} \tag{41}$$

Since the objective of EM is exactly the KL divergence above, the Bayesian loss will be minimized when the objective of EM becomes very small.

D Weight updates in linear RNNs

We first derive the updates for a single neuron, and then extend each algorithm to an arbitrary fixed network size N. For one neuron with linear activation $\phi(h) = h$, the stochastic RNN model reduces to:

$$\begin{aligned} h_t &= (1 - \alpha + \alpha w_r)h_{t-1} + \delta_{t1}\alpha w_i x + \sqrt{\alpha}\sigma_h \xi_t, \quad h_0 = 0, \\ \hat{y}_T &= w_o h_T + \sigma_y \epsilon_T, \end{aligned} \tag{42}$$

where we also assume that the input x is only nonzero at $t = 1$, the output is only read out at the last timestep T and input/output bias are zero, $b_i = b_o = 0$. For convenience, we denote the effective recurrent weight as $w_\alpha \equiv 1 - \alpha + \alpha w_r$.

D.1 BPTT

For illustration, we will first derive the result for networks with a single neuron and extend the calculations to networks with arbitrary number of neurons.

D.1.1 Single-neuron network

As Eq.(42) is a linear recursion, the final output can be solved in terms of the input and the noise variables as

$$\hat{y}_T = \alpha w_o w_\alpha^{T-1} w_i x + \sqrt{\alpha}\sigma_h w_o \sum_{t=1}^{T} w_\alpha^{T-t}\xi_t + \sigma_y \epsilon_T. \tag{43}$$

So the BPTT objective [Eq.(4)] becomes

$$L_{BPTT}(w) = \mathbb{E}_{\xi_t,\epsilon_T}\left[(y_T - \hat{y}_T)^2\right] = (y_T - \alpha w_\alpha^{T-1} w_o w_i x)^2 + \alpha\sigma_h^2 w_o^2 \sum_{t=0}^{T-1} w_\alpha^{2t} + \sigma_y^2$$
$$= (y_T - \alpha w_\alpha^{T-1} w_i w_o x)^2 + \alpha w_o^2 \sigma_h^2 \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} + \sigma_y^2, \tag{44}$$

where in the last equality, we assume that the network dynamics is stable $|w_\alpha| < 1$.

Gradient descent on this loss, with a fixed learning rate $\eta > 0$, yields the recurrent weight update

$$\Delta w_r = 2\eta\alpha^2(T-1) w_\alpha^{T-2} w_i w_o x (y_T - \alpha w_o w_\alpha^{T-1} w_i x)$$
$$- 2\eta\alpha^2\sigma_h^2 w_o^2 \frac{w_\alpha - T w_\alpha^{2T-1} + (T-1) w_\alpha^{2T+1}}{(1 - w_\alpha^2)^2}$$
$$= 2\eta\alpha^2(T-1) w_\alpha^{T-2} w_i w_o x\, y_T - 2\eta\alpha^2\sigma_h^2 w_o^2 \frac{w_\alpha}{(1 - w_\alpha^2)^2} + o(1), \qquad T \to \infty. \tag{45}$$

The first term comes from the mean prediction error and carries information about task-relevant input–target mapping. Its $(T-1)w_\alpha^{T-2}$ factor decays exponentially, up to a polynomial prefactor, for fixed $|w_\alpha| < 1$. The second term is induced by the state-noise contribution to the loss (Lim et al., 2021). It need not vanish with the delay but does not contain information about the task. Therefore, this shows the task-relevant learning signal in BPTT decays exponentially as the delay T increases.

D.1.2 Extension to network with N neurons

We now extend the calculation to an arbitrary fixed network size N. Consider the case where the input is only nonzero at the initial time and the target at the final time,

$$\mathbf{h}_t = W_\alpha \mathbf{h}_{t-1} + \delta_{t1}\alpha W_i \mathbf{x} + \boldsymbol{\xi}_t, \qquad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, Q),$$
$$\hat{\mathbf{y}}_T = W_o \mathbf{h}_T + \boldsymbol{\epsilon}_T, \qquad \boldsymbol{\epsilon}_T \sim \mathcal{N}(\mathbf{0}, R). \tag{46}$$

Here $\mathbf{h}_0 = \mathbf{0}$, $W_\alpha = (1-\alpha)I + \alpha W_r$. The noise convention in Eq. (1) gives $Q = \alpha\sigma_h^2 I$ and $R = \sigma_y^2 I$. We assume $\alpha > 0$, $\sigma_h > 0$, $\sigma_y > 0$, and fixed stable dynamics (the spectral radius of W_α, $\rho(W_\alpha) < 1$). For clarity, we use $\top$ for transpose, to distinguish it from the time horizon T.

The prior mean, covariance, and mean output error are

$$\mathbf{m}_t = \alpha W_\alpha^{t-1} W_i \mathbf{x}, \qquad P_t = \sum_{k=0}^{t-1} W_\alpha^k Q (W_\alpha^k)^\top, \qquad \mathbf{e}_T = \mathbf{y}_T - W_o \mathbf{m}_T. \tag{47}$$

As the single-neuron network case, the BPTT objective can be written as

$$L_{BPTT} = \mathbb{E}_{\xi_t,\epsilon_t}[(\mathbf{y}_T - \mathbf{y})^2] = \mathbf{e}_T^2 + \mathrm{tr}(W_o P_T W_o^\top) + \mathrm{tr}(R). \tag{48}$$

After some calculations, the expected recurrent update in BPTT can be written exactly into two terms as in the single-neuron case [Eq. (44)]:

$$\Delta W_{r,BPTT} = \Delta W_{r,\mathrm{sig}} + \Delta W_{r,\mathrm{noise}},$$
$$\Delta W_{r,\mathrm{sig}} = 2\eta\alpha \sum_{t=2}^{T} (W_\alpha^{T-t})^\top W_o^\top \mathbf{e}_T \mathbf{m}_{t-1}^\top, \tag{49}$$
$$\Delta W_{r,\mathrm{noise}} = -2\eta\alpha \sum_{t=2}^{T} (W_\alpha^{T-t})^\top W_o^\top C A^{T-t+1} P_{t-1}. \tag{50}$$

The 1st term $\Delta W_{r,\mathrm{sig}}$ depends on $\mathbf{e}_T$ and is induced by the task-relevant input-target mapping. The 2nd term $\Delta W_{r,\mathrm{noise}}$ is contributed by the state noise in the model and is independent of the tasks.

The task-dependent learning signal thus decays exponentially with the delay T,

$$\|\Delta W_{r,\mathrm{sig}}\| \le 2\eta\alpha^2(T-1)\rho(W_\alpha)^{T-2}\|W_i\|\|C\|\|\mathbf{e}_T\|_2\|\mathbf{x}\|_2 = O(Tr^T). \tag{51}$$

The error $\mathbf{e}_T$ remains bounded for a fixed target, as well as other norms. Thus the task-dependent learning signal decays exponentially, for any fixed stable recurrent matrix $\rho(A) < 1$. This term also resembles the product-of-Jacobians mechanism in BPTT-trained deterministic networks, which is the usual source of vanishing gradients (Pascanu et al., 2013).

As a final comment, the total noisy BPTT update need not vanish. In fact, assuming stability, the stationary covariance matrix converges to a finite value,

$$P_t \longrightarrow P = W_\alpha P W_\alpha^\top + Q, \tag{52}$$

where P denotes the unique stationary covariance. Let $\mathcal{O} = \sum_{k=0}^{\infty} (W_\alpha^k)^\top W_o^\top W_o W_\alpha^k$ be the observability Gramian. The task-independent learning signal [Eq. (50)] gives

$$\Delta W_{r,\text{noise}} \longrightarrow -2\eta\alpha\mathcal{O}AP. \tag{53}$$

This is generally a nonzero limit but is independent of the input and target. Therefore, this term alone does not support learning the task. In contrast, the *task-dependent component* of the learning signal is exponentially small and thus prevent effective task learning.

D.2 EM

Following the results in the last section, we will first derive the result for single-neuron networks and then extend to networks with arbitrary number of neurons.

D.2.1 Filtering statistics

For $1 \le t < T$, there are no observations. The filtered estimates are equivalent to the predictions. Define the scaled state-noise variance

$$Q = \alpha\sigma_h^2. \tag{54}$$

Then

$$m_t = m_t^- = w_\alpha m_{t-1} + \delta_{t,1}\alpha w_i x = \alpha w_\alpha^{t-1} w_i x,$$
$$P_t = P_t^- = w_\alpha^2 P_{t-1} + Q = Q\sum_{k'=1}^{t} w_\alpha^{2(t-k')} = Q\frac{1-w_\alpha^{2t}}{1-w_\alpha^2}, \quad |w_\alpha| < 1.$$

At $t = T$, the update involves the terminal observation y_T,

$$e_T = y_T - w_o m_T^-, \quad S_T = w_o^2 P_T^- + \sigma_y^2, \quad K_T = \frac{P_T^- w_o}{S_T}, \quad m_T = m_T^- + K_T e_T, \quad P_T = P_T^- - \frac{(P_T^- w_o)^2}{w_o^2 P_T^- + \sigma_y^2}.$$

D.2.2 Smoothing statistics

The smoothing gain is $G_t = \frac{P_t w_\alpha}{P_{t+1}^-}$. Since the gain is applied recursively from T, we utilize the telescoping property, and define J_t:

$$J_t = \prod_{k=t}^{T-1} G_k = w_\alpha^{T-t}\frac{P_t}{P_T^-}. \tag{55}$$

Substituting the analytical variance expression

$$J_t = w_\alpha^{T-t}\left(\frac{1-w_\alpha^{2t}}{1-w_\alpha^{2T}}\right). \tag{56}$$

The smoothed mean and variance for $t < T$ are:

$$m_t^s = m_t^- + J_t(m_T - m_T^-) = \alpha w_\alpha^{t-1} w_i x + J_t K_T e_T,$$
$$P_t^s = P_t^- + J_t^2(P_T - P_T^-) = Q\frac{1-w_\alpha^{2t}}{1-w_\alpha^2} - J_t^2\Delta_T.$$

where

$$\Delta_T := \frac{(w_o P_T^-)^2}{w_o^2 P_T^- + \sigma_y^2} = \frac{w_o^2 Q^2\left(\frac{1-w_\alpha^{2T}}{1-w_\alpha^2}\right)^2}{w_o^2 Q\left(\frac{1-w_\alpha^{2T}}{1-w_\alpha^2}\right) + \sigma_y^2}, \qquad P_T^- = Q\frac{1-w_\alpha^{2T}}{1-w_\alpha^2}. \tag{57}$$

D.2.3 WEIGHT UPDATE

The exact M-step update [Eq.(13)] can be written as (assume no regularization $\lambda \to 0^+$),

$$w_\alpha^{(n+1)} = \frac{C}{\Phi}. \tag{58}$$

$$C = \sum_{t=1}^{T} \Big(P_t^s \, G_{t-1} + m_t^s \, m_{t-1}^s \Big), \quad \Phi = \sum_{t=1}^{T} \Big(P_{t-1}^s + (m_{t-1}^s)^2 \Big). \tag{59}$$

For convenience, we introduce the new variables

$$C_1 = \sum_{t=1}^{T} P_t^s G_{t-1}, \qquad C_2 = \sum_{t=1}^{T} m_t^s m_{t-1}^s, \qquad \Phi_1 = \sum_{t=1}^{T} P_{t-1}^s, \qquad \Phi_2 = \sum_{t=1}^{T} (m_{t-1}^s)^2. \tag{60}$$

The smoothing gain can be written as

$$G_{t-1} = w_\alpha \, \frac{1 - w_\alpha^{2(t-1)}}{1 - w_\alpha^{2t}}, \tag{61}$$

We split C_1 in Eq.(60) into two terms,

$$C_1 = \underbrace{C_1^{(1)}}_{\text{from } Q\frac{1-w_\alpha^{2t}}{1-w_\alpha^2} G_{t-1}} + \underbrace{C_1^{(2)}}_{\text{from } -J_t^2 \Delta_T G_{t-1}}. \tag{62}$$

The two terms can be simplified as

$$C_1^{(1)} = \sum_{t=1}^{T} \left[Q \frac{1 - w_\alpha^{2t}}{1 - w_\alpha^2} \frac{1 - w_\alpha^{2(t-1)}}{1 - w_\alpha^{2t}} w_\alpha \right] = \frac{Q w_\alpha}{1 - w_\alpha^2} \sum_{t=1}^{T} \Big(1 - w_\alpha^{2(t-1)} \Big) = \frac{Q w_\alpha}{1 - w_\alpha^2} \left(T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right). \tag{63}$$

$$\begin{aligned} C_1^{(2)} &= -\Delta_T \sum_{t=1}^{T} w_\alpha^{2(T-t)} \left(\frac{1 - w_\alpha^{2t}}{1 - w_\alpha^{2T}} \right)^2 \frac{1 - w_\alpha^{2(t-1)}}{1 - w_\alpha^{2t}} w_\alpha \\ &= -\Delta_T \left[\frac{w_\alpha^{2T+1}}{(1 - w_\alpha^{2T})^2} \right] \left[\frac{w_\alpha^{-2}(1 - w_\alpha^{-2T})}{1 - w_\alpha^{-2}} - T(1 + w_\alpha^{-2}) + \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right]. \end{aligned} \tag{64}$$

Similarly, for C_2 in Eq.(60), it can be split into four terms,

$$\begin{aligned} C_2 &= \sum_{t=1}^{T} m_t^s m_{t-1}^s \\ &= \underbrace{C_2^{(1)}}_{\text{from } \alpha w_\alpha^{t-1} w_i x \cdot \alpha w_\alpha^{t-2} w_i x} + \underbrace{C_2^{(2)}}_{\text{from } (J_t K_T e_T) \cdot (J_{t-1} K_T e_T)} + \underbrace{C_2^{(3)}}_{\text{from } \alpha w_\alpha^{t-1} w_i x \cdot (J_{t-1} K_T e_T)} + \underbrace{C_2^{(4)}}_{\text{from } \alpha w_\alpha^{t-2} w_i x \cdot (J_t K_T e_T)}. \end{aligned} \tag{65}$$

We compute the four terms separately. Since $h_0 = 0$, the terms involving m_{t-1}^s have no contribution from $t = 1$:

$$C_2^{(1)} = \sum_{t=2}^{T} \Big(\alpha w_\alpha^{t-1} w_i x \Big) \Big(\alpha w_\alpha^{t-2} w_i x \Big) = \alpha^2 w_i^2 x^2 \sum_{t=2}^{T} w_\alpha^{2t-3} = \alpha^2 w_i^2 x^2 w_\alpha \left[\frac{1 - w_\alpha^{2T-2}}{1 - w_\alpha^2} \right]. \tag{66}$$

$$
\begin{aligned}
C_2^{(2)} &= \sum_{t=1}^{T} \Big(J_t K_T e_T\Big)\Big(J_{t-1} K_T e_T\Big) \\
&= K_T^2 e_T^2 \sum_{t=1}^{T} \left(w_\alpha^{T-t} \frac{1-w_\alpha^{2t}}{1-w_\alpha^{2T}}\right)\left(w_\alpha^{T-t+1} \frac{1-w_\alpha^{2(t-1)}}{1-w_\alpha^{2T}}\right) \\
&= K_T^2 e_T^2 \frac{w_\alpha^{2T+1}}{(1-w_\alpha^{2T})^2} \sum_{t=1}^{T} w_\alpha^{-2t}(1-w_\alpha^{2t})(1-w_\alpha^{2t-2}) \\
&= K_T^2 e_T^2 \frac{w_\alpha^{2T+1}}{(1-w_\alpha^{2T})^2} \left(\frac{w_\alpha^{-2}(1-w_\alpha^{-2T})}{1-w_\alpha^{-2}} - T - T w_\alpha^{-2} + \frac{1-w_\alpha^{2T}}{1-w_\alpha^{2}}\right). \qquad (67)
\end{aligned}
$$

$$
\begin{aligned}
C_2^{(3)} &= \sum_{k=1}^{T} \Big(\alpha w_\alpha^{k-1} w_i x\Big)\Big(J_{k-1} K_T e_T\Big) \\
&= K_T e_T \sum_{k=1}^{T} \alpha w_\alpha^{k-1} w_i x \left[w_\alpha^{T-k+1} \frac{1-w_\alpha^{2(k-1)}}{1-w_\alpha^{2T}}\right] \\
&= \frac{\alpha w_i x\, w_\alpha^{T}}{1-w_\alpha^{2T}} K_T e_T \sum_{k=1}^{T} \Big(1-w_\alpha^{2(k-1)}\Big) = \frac{\alpha w_i x\, w_\alpha^{T}}{1-w_\alpha^{2T}} K_T e_T \left[T - \frac{1-w_\alpha^{2T}}{1-w_\alpha^{2}}\right]. \qquad (68)
\end{aligned}
$$

$$
\begin{aligned}
C_2^{(4)} &= \sum_{k=2}^{T} \Big(\alpha w_\alpha^{k-2} w_i x\Big)\Big(J_k K_T e_T\Big) \\
&= K_T e_T \sum_{k=2}^{T} \alpha w_\alpha^{k-2} w_i x \left[w_\alpha^{T-k} \frac{1-w_\alpha^{2k}}{1-w_\alpha^{2T}}\right] \\
&= \frac{\alpha w_i x\, w_\alpha^{T-2}}{1-w_\alpha^{2T}} K_T e_T \sum_{k=2}^{T} \Big(1-w_\alpha^{2k}\Big) = \frac{\alpha w_i x\, w_\alpha^{T-2}}{1-w_\alpha^{2T}} K_T e_T \left[(T-1) - \frac{w_\alpha^{4}(1-w_\alpha^{2T-2})}{1-w_\alpha^{2}}\right]. \qquad (69)
\end{aligned}
$$

We perform similar computation for the Φ terms in Eq.(60),

$$
\Phi = \underbrace{\Phi_1}_{\sum_{t=1}^{T} P_{t-1}^{s}} + \underbrace{\Phi_2}_{\sum_{t=1}^{T} (m_{t-1}^{s})^2}. \qquad (70)
$$

$$
\Phi_1 = \underbrace{\Phi_1^{(1)}}_{\text{from } P_{t-1}^{-}} + \underbrace{\Phi_1^{(2)}}_{\text{from } -J_{t-1}^2 \Delta_T}. \qquad (71)
$$

$$
\Phi_1^{(1)} = \sum_{t=1}^{T} P_{t-1}^{-} = \sum_{t=1}^{T} Q \frac{1-w_\alpha^{2(t-1)}}{1-w_\alpha^{2}} = \frac{Q}{1-w_\alpha^{2}} \sum_{t=1}^{T} \Big(1-w_\alpha^{2(t-1)}\Big) = \frac{Q}{1-w_\alpha^{2}} \left[T - \frac{1-w_\alpha^{2T}}{1-w_\alpha^{2}}\right]. \qquad (72)
$$

$$\Phi_1^{(2)} = \sum_{t=1}^{T} \Big(- J_{t-1}^2 \Delta_T \Big) = -\Delta_T \sum_{t=1}^{T} J_{t-1}^2$$
$$= -\Delta_T \sum_{t=1}^{T} w_\alpha^{2(T-t+1)} \left(\frac{1 - w_\alpha^{2(t-1)}}{1 - w_\alpha^{2T}} \right)^2$$
$$= -\Delta_T \frac{w_\alpha^{2T+2}}{(1 - w_\alpha^{2T})^2} \sum_{t=1}^{T} w_\alpha^{-2t} \Big(1 - w_\alpha^{2(t-1)} \Big)^2$$
$$= -\Delta_T \frac{w_\alpha^{2T+2}}{(1 - w_\alpha^{2T})^2} \left[\frac{w_\alpha^{-2}\big(1 - w_\alpha^{-2T}\big)}{1 - w_\alpha^{-2}} - 2T w_\alpha^{-2} + \frac{w_\alpha^{-2}\big(1 - w_\alpha^{2T}\big)}{1 - w_\alpha^{2}} \right]. \tag{73}$$

$$\Phi_2 = \sum_{t=1}^{T} (m_{t-1}^s)^2 = \underbrace{\Phi_2^{(1)}}_{\text{from } (\alpha w_\alpha^{t-2} w_i x)^2} + \underbrace{\Phi_2^{(2)}}_{\text{from } 2(\alpha w_\alpha^{t-2} w_i x)(J_{t-1} K_T e_T)} + \underbrace{\Phi_2^{(3)}}_{\text{from } (J_{t-1} K_T e_T)^2}.$$

$$\Phi_2^{(1)} = \sum_{t=2}^{T} \alpha^2 w_\alpha^{2(t-2)} w_i^2 x^2 = \alpha^2 w_i^2 x^2 \sum_{t=2}^{T} w_\alpha^{2(t-2)} = \alpha^2 w_i^2 x^2 \left(\frac{1 - w_\alpha^{2T-2}}{1 - w_\alpha^2} \right).$$

$$\Phi_2^{(2)} = \sum_{t=1}^{T} 2\Big(\alpha w_\alpha^{t-2} w_i x \Big) \Big(J_{t-1} K_T e_T \Big) = \frac{2\alpha w_\alpha^{T-1} w_i x}{1 - w_\alpha^{2T}} K_T e_T \sum_{t=1}^{T} \Big(1 - w_\alpha^{2(t-1)} \Big)$$
$$= \frac{2\alpha w_\alpha^{T-1} w_i x}{1 - w_\alpha^{2T}} K_T e_T \left[T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^{2}} \right].$$

$$\Phi_2^{(3)} = \sum_{t=1}^{T} \Big(J_{t-1} K_T e_T \Big)^2 = (K_T e_T)^2 \sum_{t=1}^{T} J_{t-1}^2$$
$$= (K_T e_T)^2 \frac{w_\alpha^{2T+2}}{(1 - w_\alpha^{2T})^2} \left[\frac{w_\alpha^{-2}\big(1 - w_\alpha^{-2T}\big)}{1 - w_\alpha^{-2}} - 2T\, w_\alpha^{-2} + \frac{w_\alpha^{-2}\big(1 - w_\alpha^{2T}\big)}{1 - w_\alpha^{2}} \right].$$

Now that we have established the necessary terms, we want to analyze how the weight update behaves. First, we plug the above terms into the weight update formula:

$$w_\alpha^{(n+1)} - w_\alpha = \frac{C}{\Phi} - w_\alpha = \frac{C - w_\alpha \Phi}{\Phi} = \frac{\mathcal{N}}{\mathcal{D}}. \tag{74}$$

$$\begin{aligned}
\mathcal{N} = {} & \frac{Q w_\alpha}{1 - w_\alpha^2} \left(T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right) - \Delta_T \frac{w_\alpha^{2T+1}}{(1 - w_\alpha^{2T})^2} \left[\frac{w_\alpha^{-2}(1 - w_\alpha^{-2T})}{1 - w_\alpha^{-2}} - T(1 + w_\alpha^{-2}) + \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right] \\
& + \alpha^2 w_i^2 x^2 w_\alpha \left(\frac{1 - w_\alpha^{2T-2}}{1 - w_\alpha^2} \right) + K_T^2 e_T^2 \frac{w_\alpha^{2T+1}}{(1 - w_\alpha^{2T})^2} \left[\frac{w_\alpha^{-2}(1 - w_\alpha^{-2T})}{1 - w_\alpha^{-2}} - T(1 + w_\alpha^{-2}) + \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right] \\
& + K_T e_T \frac{\alpha w_i x w_\alpha^T}{1 - w_\alpha^{2T}} \left[T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right] + K_T e_T \frac{\alpha w_i x\, w_\alpha^{T-2}}{1 - w_\alpha^{2T}} \left[(T-1) - \frac{w_\alpha^4 (1 - w_\alpha^{2T-2})}{1 - w_\alpha^2} \right] \\
& - \frac{Q w_\alpha}{1 - w_\alpha^2} \left[T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right] + \Delta_T \left[\frac{w_\alpha^{2T+3}}{(1 - w_\alpha^{2T})^2} \right] \left[\frac{w_\alpha^{-2}(1 - w_\alpha^{-2T})}{1 - w_\alpha^{-2}} - 2T\, w_\alpha^{-2} + \frac{w_\alpha^{-2}(1 - w_\alpha^{2T})}{1 - w_\alpha^2} \right] \\
& - \alpha^2 w_i^2 x^2 w_\alpha \left(\frac{1 - w_\alpha^{2T-2}}{1 - w_\alpha^2} \right) - K_T e_T \frac{2\alpha w_\alpha^T w_i x}{1 - w_\alpha^{2T}} \left[T - \frac{1 - w_\alpha^{2T}}{1 - w_\alpha^2} \right] \\
& - K_T^2 e_T^2 \frac{w_\alpha^{2T+3}}{(1 - w_\alpha^{2T})^2} \left[\frac{w_\alpha^{-2}(1 - w_\alpha^{-2T})}{1 - w_\alpha^{-2}} - 2T\, w_\alpha^{-2} + \frac{w_\alpha^{-2}(1 - w_\alpha^{2T})}{1 - w_\alpha^2} \right].
\end{aligned} \tag{75}$$

$$\mathcal{D} = \frac{Q}{1-w_\alpha^2}\left(T - \frac{1-w_\alpha^{2T}}{1-w_\alpha^2}\right) - \Delta_T \frac{w_\alpha^{2T+2}}{(1-w_\alpha^{2T})^2}\left[\frac{w_\alpha^{-2}(1-w_\alpha^{-2T})}{1-w_\alpha^{-2}} - 2T\, w_\alpha^{-2} + \frac{w_\alpha^{-2}(1-w_\alpha^{2T})}{1-w_\alpha^2}\right]$$
$$+ \alpha^2 w_i^2 x^2 \left(\frac{1-w_\alpha^{2T-2}}{1-w_\alpha^2}\right) + K_T e_T \frac{2\alpha w_\alpha^{T-1} w_i x}{1-w_\alpha^{2T}}\left(T - \frac{1-w_\alpha^{2T}}{1-w_\alpha^2}\right)$$
$$+ K_T^2 e_T^2 \frac{w_\alpha^{2T+2}}{(1-w_\alpha^{2T})^2}\left[\frac{w_\alpha^{-2}(1-w_\alpha^{-2T})}{1-w_\alpha^{-2}} - 2T\, w_\alpha^{-2} + \frac{w_\alpha^{-2}(1-w_\alpha^{2T})}{1-w_\alpha^2}\right]. \tag{76}$$

Then we take the limit $T \to \infty$, under the regime $|w_\alpha| < 1$:

$$w_r^{(n+1)} - w_r = \frac{1}{\alpha}(w_\alpha^{(n+1)} - w_\alpha) = \frac{1}{\alpha}\frac{\Delta_T\left[\frac{w_\alpha^{-1}}{1-w_\alpha^{-2}}\right] - K_T^2 e_T^2\left[\frac{w_\alpha^{-1}}{1-w_\alpha^{-2}}\right] - \Delta_T\left[\frac{w_\alpha}{1-w_\alpha^{-2}}\right] + K_T^2 e_T^2\left[\frac{w_\alpha}{1-w_\alpha^{-2}}\right]}{\frac{Q}{1-w_\alpha^2}\left[T - \frac{1}{1-w_\alpha^2}\right] + \frac{\alpha^2 w_i^2 x^2}{1-w_\alpha^2} + \frac{(K_T^2 e_T^2 - \Delta_T) w_\alpha^2}{1-w_\alpha^2}}.$$

After some algebra,

$$w_r^{(n+1)} - w_r = \frac{1}{\alpha}\frac{(K_T^2 e_T^2 - \Delta_T)(1-w_\alpha^2) w_\alpha}{Q\left(T - \frac{1}{1-w_\alpha^2}\right) + \alpha^2 w_i^2 x^2 + (K_T^2 e_T^2 - \Delta_T) w_\alpha^2}$$
$$\to \frac{w_\alpha Q w_o^2\left[(y_T^2 - \sigma_y^2)(1-w_\alpha^2) - Q w_o^2\right]}{\alpha T\left[Q w_o^2 + \sigma_y^2(1-w_\alpha^2)\right]^2}, \quad T \gg 1. \tag{77}$$

Equivalently, since $Q = \alpha\sigma_h^2$,

$$w_r^{(n+1)} - w_r \to \frac{w_\alpha \sigma_h^2 w_o^2\left[(y_T^2 - \sigma_y^2)(1-w_\alpha^2) - \alpha\sigma_h^2 w_o^2\right]}{T\left[\alpha\sigma_h^2 w_o^2 + \sigma_y^2(1-w_\alpha^2)\right]^2}, \quad T \gg 1.$$

The exact EM update therefore depends inversely on the delay T, which differs from the exponentially decaying BPTT update in Eq. (45).

D.2.4 Extension to N neurons

Consider the N-neuron model in Eq. (46), with the same fixed-stability and positive-noise assumptions as in Appendix D.1.2. Define

$$q_T = p_W(\mathbf{h}_{1:T} \mid \mathbf{y}_T, \mathbf{x}), \qquad S_T = W_o P_T W_o^\top + R, \qquad \mathbf{e}_T = \mathbf{y}_T - \alpha W_o W_\alpha^{T-1} W_i \mathbf{x},$$
$$H_T = W_o^\top S_T^{-1}(\mathbf{e}_T \mathbf{e}_T^\top - S_T) S_T^{-1} W_o. \tag{78}$$

Here H_T is a symmetric matrix.

The Gaussian transition likelihood gives the regression M-step for the matrix $W_\alpha = (1-\alpha)I + \alpha W_r$,

$$W_\alpha^{\text{new}} = \left(\sum_{t=2}^T \mathbb{E}_{q_T}[\mathbf{h}_t \mathbf{h}_{t-1}^\top]\right)\left(\sum_{t=2}^T \mathbb{E}_{q_T}[\mathbf{h}_{t-1}\mathbf{h}_{t-1}^\top]\right)^{-1},$$
$$W_\alpha^{\text{new}} - W_\alpha = \left(\sum_{t=2}^T \mathbb{E}_{q_T}[\boldsymbol{\xi}_t \mathbf{h}_{t-1}^\top]\right)\left(\sum_{t=2}^T \mathbb{E}_{q_T}[\mathbf{h}_{t-1}\mathbf{h}_{t-1}^\top]\right)^{-1}. \tag{79}$$

We derive the explicit form of the numerator. For $t \geq 2$, the noise $\boldsymbol{\xi}_t$ and the state $\mathbf{h}_{t-1}$ are independent under the prior, and their covariance with the output are

$$\text{Cov}(\boldsymbol{\xi}_t, \hat{\mathbf{y}}_T) = Q(W_\alpha^{T-t})^\top W_o^\top, \qquad \text{Cov}(\hat{\mathbf{y}}_T, \mathbf{h}_{t-1}) = W_o W_\alpha^{T-t+1} P_{t-1}.$$

So the joint posterior distribution between $\boldsymbol{\xi}_t$ and $\mathbf{h}_{t-1}$ can be obtained through Gaussian conditioning. Specifically, the expectation over the posterior becomes

$$\mathbb{E}_{q_T}[\boldsymbol{\xi}_t \mathbf{h}_{t-1}^\top] = Q(W_\alpha^{T-t})^\top W_o^\top S_T^{-1} \mathbf{e}_T \mathbf{m}_{t-1}^\top + Q(W_\alpha^{T-t})^\top H_T W_\alpha^{T-t+1} P_{t-1}. \tag{80}$$

The denominator can be computed in a similar way. Set

$$L_{s,T} = P_s (W_\alpha^{T-s})^\top W_o^\top S_T^{-1}. \tag{81}$$

The smoothed second moments in the denominator are exactly

$$\mathbb{E}_{q_T}[\mathbf{h}_s \mathbf{h}_s^\top] = P_s + \mathbf{m}_s \mathbf{m}_s^\top + L_{s,T}(\mathbf{e}_T \mathbf{e}_T^\top - S_T) L_{s,T}^\top + \mathbf{m}_s \mathbf{e}_T^\top L_{s,T}^\top + L_{s,T} \mathbf{e}_T \mathbf{m}_s^\top. \tag{82}$$

In the limit of long delay $T \gg 1$, let

$$\begin{aligned} P_T \to P = W_\alpha P W_\alpha^\top + Q, \qquad S_T \to S = W_o P W_o^\top + R, \\ H_T \to H = W_o^\top S^{-1}(\mathbf{y}_T \mathbf{y}_T^\top - S) S^{-1} W_o, \qquad \mathcal{G} = \sum_{k=0}^{\infty} (W_\alpha^k)^\top H W_\alpha^k. \end{aligned} \tag{83}$$

Here the target $\mathbf{y}_T$ is held fixed as the delay is increased. The series in $\mathcal{G}$ is absolutely convergent due to the stability assumption $\rho(W_\alpha) < 1$.

The first term of Eq. (80), summed over t, is $O(T\rho(W_\alpha)^{T-2})$ and vanishes exponentially. The second term gives

$$B_T \longrightarrow Q \mathcal{G} W_\alpha P. \tag{84}$$

On the other hand, the denominator satisfies

$$\sum_{t=2}^{T} \mathbb{E}_{q_T}[\mathbf{h}_{t-1} \mathbf{h}_{t-1}^\top] = TP + O(1), \qquad \left(\sum_{t=2}^{T} \mathbb{E}_{q_T}[\mathbf{h}_{t-1} \mathbf{h}_{t-1}^\top] \right)^{-1} = T^{-1} P^{-1} + O(T^{-2}). \tag{85}$$

Combining these results yields

$$W_r^{\text{new}} - W_r = \frac{W_\alpha^{\text{new}} - W_\alpha}{\alpha} = \frac{1}{\alpha T} Q \mathcal{G} W_\alpha + o(T^{-1}). \tag{86}$$

Since all the matrices have finite norms, the leading EM update depends *inversely* on the delay T. Plugging in $Q = \alpha \sigma_h^2 I$ gives the equations displayed in the main text.

E Posterior inference preserves modularity under distribution shifts

This appendix shows using the posterior distribution to update the weight matrix reduces interference between different modules formed in the network.

Suppose that the phase 1 solution is approximately organized into K context-specific modules,

$$\mathbf{h}_t = (\mathbf{h}_t^{(1)}, \ldots, \mathbf{h}_t^{(K)}), \qquad \sum_{j=1}^{K} n_j = N. \tag{87}$$

Assuming the output units also form K groups, each only readout by one module

$$\mathbf{y}_T^{(j)} = V_j \mathbf{a}_T^{(j)} + \boldsymbol{\epsilon}_T^{(j)}, \qquad \boldsymbol{\epsilon}_T^{(j)} \sim \mathcal{N}(0, \sigma_y^2 I). \tag{88}$$

After the distribution shift, training examples come only from context 1, and the other observed output blocks have targets $\mathbf{y}_t^{(j)} = 0$ for $j \neq 1$.

For an inactive module $j \neq 1$ at a readout time $t \in \Theta$, the observation likelihood has the factor

$$p(\mathbf{y}_t^{(j)} = 0 | \mathbf{h}_t^{(j)}) \propto \exp\left[-\frac{\| V_j \phi(\mathbf{h}_t^{(j)}) \|_2^2}{2\sigma_y^2} \right]. \tag{89}$$

Therefore, the posterior distribution given the readout is

$$p(\mathbf{h}_t^{(j)} | \mathbf{y}_t^{(j)} = 0, \mathbf{x}) \propto p(\mathbf{h}_t^{(j)} | \mathbf{x}) \exp\left[-\frac{\| V_j \phi(\mathbf{h}_t^{(j)}) \|_2^2}{2\sigma_y^2} \right] \tag{90}$$

The variance of the prior distribution of the states $p(\mathbf{h}_t^{(j)}|\mathbf{x})$ is generated by the state noise, which has the magnitude σ_h^2. When $\sigma_h \gg \sigma_y$, the prior distribution has much more uncertainty than the observation likelihood $\exp\left[-\frac{\|V_j\phi(\mathbf{h}_t^{(j)})\|_2^2}{2\sigma_y^2}\right]$ in the above equation. So the posterior distribution is mainly determined by the observation likelihood, which strongly suppresses the activities in the jth module, $\phi(\mathbf{h}_t^{(j)}) \ll 1$ and leads to negligible update on the readout weights.

In fact, for GEM weight update (Eq.(8)), we can make the comparison between BPTT and EM more explicitly as

$$\begin{aligned}
\Delta V^{GEM} &\propto -\frac{1}{\sigma_y^2} V_j \mathbb{E}_{p(\mathbf{h}_t^{(j)}|\mathbf{y}_t^{(j)},\mathbf{x})}\left[\phi(\mathbf{h}_t^{(j)})\phi(\mathbf{h}_t^{(j)})^T\right] \\
&\propto -\frac{1}{\sigma_y^2}\mathbb{E}_{p(\mathbf{h}_t^{(j)}|\mathbf{x})}\left[V_j\phi(\mathbf{h}_t^{(j)})\phi(\mathbf{h}_t^{(j)})^T \exp\left(-\frac{\|V_j\phi(\mathbf{h}_t^{(j)})\|_2^2}{2\sigma_y^2}\right)\right], \\
\Delta V^{BPTT} &\propto -\frac{1}{\sigma_y^2}\mathbb{E}_{p(\mathbf{h}_t^{(j)}|\mathbf{x})}\left[V_j\phi(\mathbf{h}_t^{(j)})\phi(\mathbf{h}_t^{(j)})^T\right]. \qquad (91)
\end{aligned}$$

Because of the existence of the exponential factor, the update by GEM is generally smaller than one by BPTT, $\|\Delta V^{GEM}\| \ll \|\Delta V^{BPTT}\|$.

F BPTT GENERATES TASK-INDEPENDENT LEARNING SIGNALS IN STRUCTURAL SEQUENCE GENERATION

We analyze the structural sequence generation task in Section 3.5 and show that the expected update for the BPTT objective contains no information about the temporal correlation between targets at different time points (whether the two target outputs in Fig. 3 are the same or not). More precisely, its target-dependent component vanishes in expectation. This result applies to arbitrary finite network sizes and nonlinear activations.

Let $c \in \{+1, -1\}$ encode the rule specified by the two input conditions: $c = +1$ for input $(0, 1)$ and $c = -1$ for input $(1, 0)$. The targets at the two observation times are

$$\mathbf{y}_{T_1} = \mathbf{z}, \qquad \mathbf{y}_{T_2} = c\mathbf{z}, \qquad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, I_{d_Y}). \qquad (92)$$

Both conditional target means vanish, but the information of the rule is encoded in their correlations,

$$\mathbb{E}[\mathbf{y}_{T_1} \mid c] = \mathbb{E}[\mathbf{y}_{T_2} \mid c] = \mathbf{0}, \qquad \mathbb{E}[\mathbf{y}_{T_2}\mathbf{y}_{T_1}^\top \mid c] = cI_{d_Y}. \qquad (93)$$

Let W be all trainable weights. Write the network outputs before additive observation noise as

$$\mathbf{f}_i(c, \boldsymbol{\xi}; W) = W_o\phi(\mathbf{h}_{T_i}) + \mathbf{b}_o, \qquad i = 1, 2, \qquad (94)$$

where $\boldsymbol{\xi}$ collects all state noise. Thus, conditional on c, the state noise is independent of the targets $\mathbf{z}$.

Up to a constant, the BPTT objective can be written as

$$L_{BPTT}(W) = \frac{1}{2}\mathbb{E}_{\xi,\mathbf{z}}\|\mathbf{f}_1 - \mathbf{z}\|_2^2 + \frac{1}{2}\mathbb{E}_{\xi,\mathbf{z}}\|\mathbf{f}_2 - c\mathbf{z}\|_2^2. \qquad (95)$$

We have averaged out the independent observation noise: for the fixed σ_y in Eq. equation 1, it adds the constant $d_Y\sigma_y^2$ to the loss.

Let $J_i = \partial\mathbf{f}_i/\partial W$ be the output Jacobian, including all temporal derivatives computed by backpropagation through time (BPTT). For a fixed condition c, a gradient-descent step with learning rate η is

$$\begin{aligned}
\Delta W &= -\eta\mathbb{E}_{\xi,\mathbf{z}}\left[J_1^\top(\mathbf{f}_1 - \mathbf{z}) + J_2^\top(\mathbf{f}_2 - c\mathbf{z})\right] \\
&= -\eta\mathbb{E}_{\xi,\mathbf{z}}\left(J_1^\top\mathbf{f}_1 + J_2^\top\mathbf{f}_2\right) + \eta\mathbb{E}_\xi(J_1^\top + cJ_2^\top)\mathbb{E}_\mathbf{z}[\mathbf{z}]. \qquad (96)
\end{aligned}$$

Since the mean of $\mathbf{z}$ is zero, we have

$$\Delta W = -\eta\,\mathbb{E}_{\boldsymbol{\xi}}[J_1^\top\mathbf{f}_1 + J_2^\top\mathbf{f}_2]. \qquad (97)$$

The cancellation holds separately within each condition. Therefore, the gradient update under BPTT is independent of the targets in this task and thus the network would not be able to learn at all.

The property that the BPTT update will ignore the temporal correlations between different output time points hold generally. In fact, the above BPTT loss can be written as

$$\begin{aligned} L_{BPTT}(W) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{\xi}}[\|\mathbf{f}_1\|_2^2 + \|\mathbf{f}_2\|_2^2] + \mathbb{E}_{\mathbf{z}}\|\mathbf{z}\|_2^2 - \mathbb{E}_{\boldsymbol{\xi},\mathbf{z}}[\mathbf{f}_1^\top \mathbf{z} + c\mathbf{f}_2^\top \mathbf{z}] \\ &= \frac{1}{2}\mathbb{E}_{\boldsymbol{\xi}}[\|\mathbf{f}_1\|_2^2 + \|\mathbf{f}_2\|_2^2] + \mathbb{E}_{\mathbf{z}}\|\mathbf{z}\|_2^2. \end{aligned} \tag{98}$$

This loss does not depend on the temporal correlations between targets at different output times. Thus, any task structure that relies on this correlation rather than the mean $\mathbb{E}_{\mathbf{z}}\|\mathbf{z}\|_2^2$ will be lost through BPTT training.

G Details on numerical simulations

At training iteration n, we estimate the sufficient statistics of the training data by using running averages

$$S_{\cdot\cdot}^{n+1} = (1-\gamma_{EM})S_{\cdot\cdot}^{n} + \eta_{EM} S_{\cdot\cdot,batch}^{n}, \tag{99}$$

where the dots ... represent all the possible statistics in Eq.(12). And $S_{\cdot\cdot,batch}$ is the corresponding statistics computed from the mini-batch data at this training iteration. All the statistics are initialized to be zero. Unless specified, the parameter α in the stochastic RNN [Eq.(1)] is 0.5. All simulations are performed on CPUs. All error bars in the figure measure the standard deviations across repeats. All experiments use ReLU activations.

For the teacher-student task in Figure 1 in Section 3.2, the teacher weights are randomly sampled from zero-mean Gaussian distribution whose variance $\sim \frac{1}{\text{fan-in}}$. The network size for both teacher and student is $n = 5$, input dimension $d_X = 2$ and output dimension $d_Y = 2$. Each component of input is white noise with unit variance and output is measured only at the last time step $T = 10$. The observation noise is $\sigma_y = 0.1$ and batch size is 10. For BPTT, we use stochastic gradient descent with learning rate 0.03. The parameters in EM are $\gamma_{EM} = \eta_{EM} = 0.03$. The simulations are repeated 5 times to generate the error bars. The number of Monte Carlo samples at each training step is the same as the batch size.

For the delayed XOR task in Figure 2 in Section 3.3, the input and output dimensions are $d_X = d_Y = 2$. The mean of the 2D input is sampled from a Gaussian mixture that consists of four Gaussian components with standard deviation 1.5. The mean of the 4 Gaussian components are $(1, 1)$, $(-3, -3)$, $(1, -1)$ and $(-3, 3)$. The first two components correspond to output $(1, 0)$ and the last two correspond to output $(0, 1)$, forming a XOR configuration. The input is on for the initial $T_{input} = 5$ time steps, followed by a delay period ΔT without any inputs. The network is asked to generate the output for two time steps based on XOR rule right after the delay period. The network size is $n = 32$ and the decaying parameter is $\alpha = 1$. Batch size is 32 and the same initialization is used as in the teacher-student task. We used Adam for BPTT training, and swept different learning rates by choosing $P = 100$, $\Delta T = 2$, $\sigma_h = 0.126$, and found the optimal learning rate to be equal to 0.001. For the EM, we applied the same procedure and found optimal $\gamma_{EM} = 0.03$, but we did not optimize for the learning rate η_{EM} and simply set it equal to γ_{EM}. l^2-weight regularization strength is $\lambda = 0.0002$. Since we are using a nonlinear decoder, we crudely approximated error forcing by not forcing the $\mathbf{h}_t$ but the $\phi(\mathbf{h}_t)$. Since error forcing introduces no extra hyperparameters beyond those of BPTT, we did not perform any separate hyperparameter tuning for error forcing for a fair comparison with BPTT. The simulations are repeated 8 times to generate the error bars.

For the context-dependent evidence accumulation task in Figure 3 in Section 3.4, the input dimension is 4. Two of these dimensions are for the one-hot contextual cues. The remaining two input dimensions represent the actual inputs that the network needs to integrate over time. Before the distribution shift, each dimension of the actual inputs has mean randomly chosen from ± 1, with white noise (standard deviation is 0.3) added. The input period lasts for $T = 5$. The output dimension is also 2. In each context, the network needs to output the mean of the corresponding input dimension in the corresponding output dimension. After the distribution shift, the network only sees that training data from context 1 and moreover the mean of the corresponding input dimension is halved with the white noise level increased to three times. The performance of network is evaluated based on

the data sampled from this new post-shift context 1 and the original pre-shift context 2. Batch size is 5 and the same initialization is used as in the teacher-student task. Stochastic gradient descent is used for BPTT training with learning rate 0.04 and no l^2-weight regularization. For EM, the learning rate parameters are $\eta_{EM} = 0.05$. The initial $\gamma_{EM} = 0.05$ and gradually annealed to zero during phase (1) training. After the distribution shift, γ_{EM} and η_{EM} are fixed at their final phase (1) values. This EM implementation therefore includes an adjustable inertia in the running sufficient statistics, a feature absent from BPTT. Thus, part of EM's robustness to distribution shift may reflect this inertia effect, in addition to posterior inference. Disentangling these two effects is an important direction for future work. The simulations are repeated 5 times to generate the error bars.

For the structural sequence generation task in Figure 3 in Section 3.5, the 2D inputs are randomly sampled from two choices $(0, 1)$ or $(1, 0)$. The network is asked to match output to the training data at two time points $T_1 = 3$ and $T_2 = 6$. The output is also 2D. If the input is $(0, 1)$, the two outputs have the same value, which is sampled from the standard 2D Gaussian distribution. If the input is $(1, 0)$, the two outputs have the opposite values (Figure 3D, top), which is sampled from the standard 2D Gaussian distribution. The network size is $n = 10$ and the decaying parameter is $\alpha = 0.1$. Batch size is 3 and the same initialization is used as in the teacher-student task. Stochastic gradient descent is used for BPTT training with learning rate 0.01 and l^2-weight regularization strength $\lambda = 0.0001$. For EM, the learning rate parameters are $\gamma_{EM} = 0.07 = 5\eta_{EM}$. The simulations are repeated 5 times to generate the error bars. The test performance of the EM-trained network is computed by using the posterior distribution of the latent variables $\mathbf{h}$ conditioned on the input and first output element, in the MSE and BL metrics [rather than the prior distribution as in Eq.(9) and (10)]. As BPTT fundamentally lacks the ability to perform output-conditional inference, we use prior distribution in Eq.(9) and (10) to compute its performance on test data.