LEARNING IN TEMPORALLY STRUCTURED ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Natural environments have temporal structure at multiple timescales, a property that is reflected in biological learning and memory but typically not in machine learning systems. This paper advances a multiscale learning method in which each weight in a neural network is decomposed into a sum of subweights with different learning and decay rates. Thus knowledge becomes distributed across different timescales, enabling rapid adaptation to task changes while avoiding catastrophic interference. First, we prove that previous models that learn at multiple timescales, but with complex coupling between timescales, are formally equivalent to the multiscale learner via a reparameterization that eliminates this coupling. The same analysis also offers a new characterization of momentum learning, as a fast weight with a negative learning rate. Second, we derive a model of Bayesian inference over $1/f$ noise, a common temporal pattern in many online learning domains that involves long-range (power law) autocorrelations. The Bayesian model uses a Kalman filter to jointly infer dynamics at multiple timescales. We then derive a variational approximation to the Bayesian model and show it is a special case of the multiscale learner. Third, we evaluate these methods in synthetic online prediction tasks characterized by $1/f$ noise in the latent parameters. We find that the Bayesian model significantly outperforms stochastic gradient descent and a batch heuristic that predicts each timestep based on a fixed horizon of past observations. Moreover, the multiscale learner with parameters obtained from the variational approximation performs nearly as well as the full Bayesian model, and with memory requirements that are linear in the size of the network. Future work will incorporate the multiscale learner as an optimizer in deep networks to explore their ability to learn in rich, temporally structured environments.

1 INTRODUCTION

Many online tasks facing both biological and artificial intelligence systems involve changes in data distribution over time. Natural environments exhibit correlations at a wide range of timescales, a pattern variously referred to as self-similarity, power-law correlations, and $1/f$ noise (Keshner, 1982). This is in stark contrast with the IID environments assumed by many machine learning (ML) methods. It also differs from diffusion or random-walk environments that (as we show below) are implicitly assumed by many online-learning methods such as temporal-difference learning and stochastic gradient descent (SGD), and which exhibit only short-range correlations. Moreover, biological learning systems are well-tuned to the temporal statistics of natural environments, as seen in phenomena of human cognition including power laws in learning (Anderson, 1982), power-law forgetting (Wixted & Ebbesen, 1997), long-range sequential effects (Wilder et al., 2013), and spacing effects (Cepeda et al., 2008; Anderson & Schooler, 1991). Thus an important goal is to incorporate similar inductive biases into ML systems. This may yield improved performance in tasks with rich temporal structure, as arise in online or continual learning settings with real-world data.

This paper analyzes a simple and general framework for learning in temporally structured environments, *multiscale learning*, which in the context of neural networks (NNs) amounts to a new kind of optimizer. A common explanation for self-similar temporal structure in nature is that it arises from a mixture of events at various timescales. Indeed, many generative models of $1/f$ noise involve summing independent processes (e.g., wavelets or random walks) with varying time constants (Eliazar & Klafter, 2009). Accordingly, the multiscale optimizer assumes multiple learning processes oper-

ating in parallel at different timescales. In a NN, every weight is replaced by a family of subweights, each subweight having its own learning rate and decay rate, that sum to determine the weight as a whole. Learning at multiple timescales is a key idea in several theories in psychology and neuroscience, including conditioning (Staddon et al., 2002), learning (Benna & Fusi, 2016), memory (Howard & Kahana, 2002; Mozer et al., 2009), and motor control (Kording et al., 2007). The multiscale learner isolates and simplifies this idea, by assuming knowledge at different timescales evolves independently and that credit assignment follows standard gradient descent.

The first part of this paper (Sections 2 and 3) presents a set of formal model equivalence results that demonstrate the utility of multiscale learning as a framework for organizing the space of models that learn at multiple timescales. We prove exact correspondences between the multiscale optimizer and three other models: a new variant of fast weights (cf. Ba et al., 2016; Hinton & Plaut, 1987), the model synapse of Benna & Fusi (2016), and momentum learning (Rumelhart et al., 1986; Qian, 1999). The insight behind these proofs is that each of these models can be written in terms of a linear update rule with diagonalizable transition matrix. Thus the eigenvectors of this matrix all evolve independently. By writing the state of the model as a mixture of eigenvectors, we effect a coordinate transformation that exactly yields the multiscale optimizer. These results imply that the complicated coupling among timescales assumed by some models can be superfluous. They also provide a new perspective on momentum learning, with implications for how and when it is beneficial and how it interacts with nonstationarity in the task environment.

Next, in Section 4, we provide a normative grounding for the multiscale learner in terms of Bayesian inference over $1/f$ noise. Our starting point is a construction of $1/f$ noise as a sum of diffusion processes at different timescales, reflecting the assumption that natural environments comprise a variety of events with different magnitudes and durations. Exact Bayesian inference with respect to this generative process is possible using a Kalman filter (KF) that tracks the component processes jointly (Kording et al., 2007). When learning a single environmental parameter θ, such as mean reward for some action in a bandit task, this amounts to modeling $\theta = \sum \theta_i$, where each θ_i evolves with a different timescale τ_i, and doing joint inference over $\Theta = (\theta_1, \ldots, \theta_n)^\top$.

We then generalize this approach to an arbitrary generative statistical model, $h(x, \theta)$, where x is the input and $\theta \in \mathbb{R}^m$ is a latent parameter vector to be estimated. For instance, h might be a NN with parameters θ. Our Bayesian model places a $1/f$ prior on θ (as a stochastic dynamic process), represented as a sum of m-dimensional diffusion processes at different timescales τ_i. This construction subsumes traditional notions of distribution shift, including covariate shift, label shift, and function shift. We extend the exact Bayesian KF above to an extended Kalman filter (EKF) that linearizes h by calculating its gradient at each step. Next, we derive a variational approximation wherein the covariance matrix is constrained to be diagonal, and show that it is a special case of the multiscale optimizer. Specifically, writing w_{ij} as the current mean estimate of θ_{ij} (the jth weight at the ith timescale), the variational update to each w_{ij} follows that of the multiscale optimizer, with decay rate and learning rate both determined by τ_i and the current prior variance s_{ij}^2. Thus the multiscale optimizer is grounded in a Bayesian interpretation as inference over $1/f$ noise.

Finally, in Section 5, we test the methods developed here in synthetic online prediction and classification tasks with nonstationary distributions. In machine learning, nonstationarity often manifests as poorer generalization performance on future data versus on held-out data from within the training interval. Common solutions are to train on a window of fixed length (to exclude older "stale" data) or stochastic gradient descent (SGD) with fixed learning rate and weight decay, which leads older observations to have less influence (Ditzler et al., 2015). Here, we demonstrate that performance can be significantly improved by retaining all data and using a learning model that reflects the temporal structure of the environment. Our simulations assume the latent parameters vary over time according to $1/f$ noise. An important caveat is that the task domains are matched to the Bayesian model in that they are simulated using the same sum-of-diffusion process that the model assumes. Notwithstanding, we test robustness by using a different set of timescales for task generation versus learning and, in Section 5.2, a generative process that mismatches the NN architecture. Results show that the Bayesian methods (KF and EKF) outperform heuristic methods of windowing or online SGD, demonstrating the value of modeling temporal structure. We also find that the variational approximation performs nearly as well as the full model, which speaks to the feasibility of this approach in contemporary applications of large NNs.

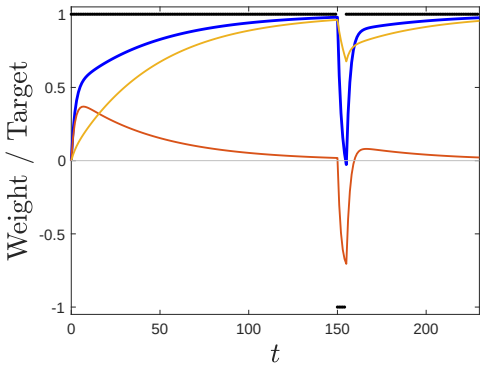

Figure 1: Illustration of fast weights. In this toy example, a single weight w (blue) with constant input ($x \equiv 1$) predicts a target signal T (black) with square loss $\mathcal{L} = \frac{1}{2}(T - w)^2$. The weight is a sum of subweights w_{slow} (yellow) and w_{fast} (red). Initial learning is rapid, due to w_{fast}. Because of decay and the shared error signal, knowledge is gradually transferred to w_{slow} while w_{fast} returns to zero. When the task switches (trial 151), w_{fast} enables rapid adaptation while long-term knowledge is preserved in w_{slow} (compare blue learning curves beginning on trials 1 and 156). Thus the model recovers quickly on the second reversal. The general multiscale learner extends this idea to an array of faster and slower weights.

2 MULTISCALE LEARNING MODEL

Assume a statistical model $\hat{y}(t) = h(x(t), w(t))$ and loss function $\mathcal{L}(y, \hat{y})$, where $x(t)$ is the input on step t, $w(t)$ is the parameter estimate, $\hat{y}(t)$ is the model output, and $y(t)$ is the target output. In a NN, $w(t)$ is the vector of current weights. For exposition, we assume the weights are updated by SGD,

$$w(t+1) = w(t) - \alpha \frac{\partial}{\partial w(t)} \mathcal{L}(y(t), \hat{y}(t)), \tag{1}$$

and we henceforth abbreviate the gradient as $\partial_{w(t)}\mathcal{L}$. However, the following approach naturally generalizes to other optimizers, such as extensions of SGD or Hebbian learning, by replacing $-\alpha\partial_{w(t)}\mathcal{L}$ with the appropriate update term.

Multiscale learning is motivated by the assumption that, in online learning tasks, the true or optimal weights change over time, and change occurs on multiple timescales. The multiscale optimizer expands each weight into a sum of subweights, $w = \sum w_i$, each with a different learning rate α_i and decay rate γ_i. Thus the subweights evolve according to:

$$w_i(t+1) = \gamma_i w_i(t) - \alpha_i \partial_{w(t)}\mathcal{L}. \tag{2}$$

Each w_i has characteristic timescale $\tau_i := (-\log \gamma_i)^{-1}$. Note that $\partial_{w(t)}\mathcal{L} = \partial_{w_i(t)}\mathcal{L}$, so one can think of the gradient for w being apportioned among the subweights (with total learning learning rate $\alpha = \sum \alpha_i$), or equivalently of each subweight following its own gradient.

2.1 FAST WEIGHTS

A potentially important special case of multiscale learning arises with two timescales, $w = w_{\text{slow}} + w_{\text{fast}}$. We assume $\gamma_{\text{slow}} = 1$ (no decay) and $\alpha_{\text{fast}} > \alpha_{\text{slow}}$. Thus w_{slow} can be thought of as the original weight, which is augmented by w_{fast}, a second channel between the same neurons that both learns and decays rapidly. The fast weight enables the system to adapt quickly to distribution shifts while resisting catastrophic forgetting (Figure 1).

This model is conceptually similar to the fast weights approach of Ba, Hinton, and colleagues (Ba et al., 2016; Hinton & Plaut, 1987) in that it implements a form of short-term memory in the weight matrix. In that previous work, the weights are updated by a different mechanism (Hebbian learning) than the primary weights, and they act as a memory of recent hidden states in a recurrent network. In the present conception, fast weights optimize the same loss as the primary weights, only with different temporal properties, and they act as a memory for recent learning signals (e.g., loss gradients). Thus they are perhaps better suited for handling distribution shifts of the sort considered here.

3 EQUIVALENCE RESULTS

3.1 BENNA-FUSI SYNAPSE

Benna & Fusi's (2016) model synapse is designed to capture how biochemical mechanisms in real synapses implement a cascading hierarchy of timescales, and has been adopted in ML for continual reinforcement learning (Kaplanis et al., 2018; 2019). The model assumes that, within each weight w in a network, knowledge is maintained in a 1d hierarchy of variables $u_1, \ldots, u_n$, each dynamically coupled to its immediate neighbors in what is analogous to a flow equation:

$$C_1(u_1(t+1) - u_1(t)) = g_1(u_2(t) - u_1(t)) - \partial_{w(t)}\mathcal{L} \tag{3}$$

$$C_k(u_k(t+1) - u_k(t)) = g_{k-1}(u_{k-1}(t) - u_k(t)) + g_k(u_{k+1}(t) - u_k(t)) \tag{4}$$

for $2 \leq k \leq n$, with $g_n = 0$. The external behavior of the synapse comes from u_1 alone (i.e., $w = u_1$), while $u_{2:n}$ act as stores with progressively longer timescales.

This update rule can be rewritten as

$$u(t+1) = Tu(t) - d(t), \tag{5}$$

with transition matrix T determined by the coefficients in Equations 3 and 4, and external signal $d(t)$ defined by $d_1(t) = \frac{1}{C_1}\partial_{w(t)}\mathcal{L}$ and $d_{2:n} \equiv 0$. It can be shown that the transition matrix is diagonalizable, $T = V\Lambda V^{-1}$, with eigenvalues $\Lambda_{ii} = \lambda_i < 1$ (see Appendix A). We can further enforce $V_{1\cdot} = 1$, for a purpose explained below. We refer to the eigenvectors (columns $V_{\cdot i}$) as *modes* of the system, because they are preserved over time up to a scalar. That is, if the initial state is proportional to mode i, then in the absence of external signal ($d \equiv 0$), the system will remain in that mode, decaying exponentially with rate factor λ_i:

$$u(0) \propto V_{\cdot i} \quad \Longrightarrow \quad \forall_t : u(t) = \lambda_i^t u(0) \tag{6}$$

In general, any state can be written uniquely as a linear combination of modes, $u = \sum w_i V_{\cdot i} = Vw$. Therefore we can reparameterize the model as $w := V^{-1}u$, leading to the simplified update equation:

$$w(t+1) = \Lambda w(t) + V^{-1}d(t) \tag{7}$$

Because Λ is diagonal, there is no cross-talk between the modes, unlike in the original dynamics. Thus we have derived the multiscale learner, with subweights $w_i(t)$, decay rates λ_i, and learning rates $\left[V^{-1}\right]_{i1}$. The assumption above, $V_{1\cdot} = 1$, implies $w = u_1 = \sum w_i$, so the models agree on the external behavior of the weight as a whole. Figure 2 illustrates the translation between the two models.

3.2 MOMENTUM LEARNING

The standard rationale for momentum learning is to smooth updates over time, so that oscillations along directions of high curvature cancel out while progress can be made in directions with consistent gradients (Rumelhart et al., 1986). The momentum is defined as an exponentially filtered running average of gradients, with weight update determined by current momentum:

$$g(t+1) = \beta g(t) + (1-\beta)\partial_{w(t)}\mathcal{L} \tag{8}$$

$$w(t+1) = w(t) - \eta g(t+1). \tag{9}$$

This formulation is equivalent to one in which the update $\Delta w(t) = w(t+1) - w(t)$ includes a portion of the previous update: $\Delta w(t) = -\alpha\partial_{w(t)}\mathcal{L} + \beta\Delta w(t-1)$, with $\alpha = \eta(1-\beta)$.

Paralleling the analysis in Section 3.1, we write the state of the momentum learner as $[w, g]^\top$ and use Equations 8 and 9 to obtain the update rule:

$$\begin{bmatrix} w(t+1) \\ g(t+1) \end{bmatrix} = \begin{bmatrix} 1 & -\eta\beta \\ 0 & \beta \end{bmatrix} \begin{bmatrix} w(t) \\ g(t) \end{bmatrix} + \begin{bmatrix} -\eta(1-\beta) \\ (1-\beta) \end{bmatrix} \partial_{w(t)}\mathcal{L}. \tag{10}$$

The transition matrix has eigenvectors $[1, 0]^\top$ with eigenvalue 1, and $[1, \frac{1-\beta}{\eta\beta}]^\top$ with eigenvalue β. Now use this eigenbasis to define a reparameterization:

$$\begin{bmatrix} w \\ g \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & \frac{1-\beta}{\eta\beta} \end{bmatrix} \begin{bmatrix} w_{\text{slow}} \\ w_{\text{fast}} \end{bmatrix}. \tag{11}$$

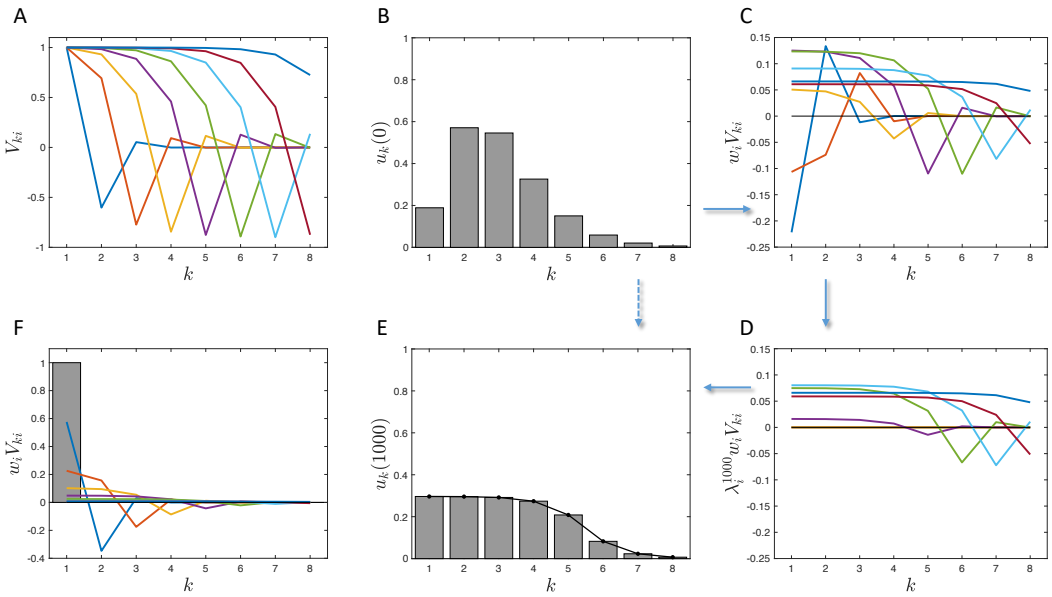

Figure 2: Translation between the model of Benna & Fusi (2016) and the multiscale optimizer works by decomposing the state of the former model into modes, or eigen-patterns of activation that decay independently, which correspond to subweights in the multiscale learner. A: All modes for a default Benna-Fusi model with eight variables ($m = 8$). B: An arbitrary initial state of the model. C: Unique eigen-decomposition of the state in Fig 2B. Implied values of the multiscale learner's subweights can be read off as the values of the curves at $k = 1$. D: Decay of the individual modes or subweights for 1000 steps (with no external input) at rates given by their eigenvalues. E: Reconstruction of the final state exactly matches the result of iterating the Benna-Fusi update (dotted arrow from Fig 2B). F: Decomposition of a unit impulse to u_1 (e.g., loss gradient, shown as grey bar) as a weighted sum of modes. Learning rates for the corresponding subweights can be read off as the values of the curves at $k = 1$.

Substitution into Equation 10 yields the reparameterized update rule:

$$\begin{bmatrix} w_{\text{slow}}(t+1) \\ w_{\text{fast}}(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} w_{\text{slow}}(t) \\ w_{\text{fast}}(t) \end{bmatrix} - \begin{bmatrix} \eta \\ -\eta\beta \end{bmatrix} \partial_{w_t} \mathcal{L}. \tag{12}$$

Thus we have recovered the fast-weight model, with decay $\gamma_{\text{fast}} = \beta$ and learning rates $\alpha_{\text{slow}} = \eta$ and $\alpha_{\text{fast}} = -\eta\beta$. The negative fast learning rate is perhaps surprising and is discussed in Section 6.

4 Bayesian derivation of the multiscale optimizer

We turn now to a normative derivation of the multiscale optimizer, based on Bayesian inference over $1/f$ noise. The Bayesian model assumes that the latent parameters θ governing the observed data in some learning task vary over time approximately according to $1/f$ noise. In the linear case, exact Bayesian inference is possible with a KF that maintains a posterior over an expanded representation of θ. When the relationship between θ and the observable outcome is nonlinear, Bayesian inference is approximated by an EKF. We then show that a variational approximation of the KF or EKF, in which the posterior covariance matrix is constrained to be diagonal, recovers the multiscale optimizer.

4.1 GENERATIVE MODEL FOR $1/f$ NOISE

Let $z_i(t)$ be an Ornstein-Uhlenbeck process (i.e., diffusion with decay), with timescale or inverse decay rate τ_i and diffusion rate σ_i^2:

$$\mathrm{d}z_i = -\tau_i^{-1} z \,\mathrm{d}t + \sigma_i \,\mathrm{d}W, \tag{13}$$

where $W(t)$ is a standard Wiener process (Brownian motion). For a chosen $\gamma > 1$, define $\tau_i = \gamma^i$ and $\sigma_i = \gamma^{-i/2}$. We show in Appendix B that the summed process,

$$\xi(t) = \sum_{i=1}^{n} z_i(t), \tag{14}$$

has a power spectrum that is well-approximated by $1/f$, for frequencies above τ_n^{-1}. Moreover, d independent copies of this process constitute d-dimensional $1/f$ noise, due to the rotational invariance of multidimensional Ornstein-Uhlenbeck processes. We henceforth write $\xi \sim \mathcal{F}_d$ to indicate that the process $\xi(t) \in \mathbb{R}^d$ is generated in this manner.

This construction affords a flexible model of distribution shift in a variety of learning domains, by applying it to the latent parameters governing observable variables, following the causal framework of (Schölkopf et al., 2012). Consider a classification task, with input features $x(t) \in \mathbb{R}^m$ and class labels $y(t) \in \mathbb{Z}_k$. Under a generative causal model where x depends on y, label shift can be modeled by $1/f$ noise in the distribution $p(y)$, for example $y \sim \mathrm{softmax}(\ell)$ with $\ell \sim \mathcal{F}_k$. manifestation shift can be modeled by $1/f$ noise in the distributions $p(x|y)$, for example $x(t)|y(t) \sim \mathcal{N}\left(\mu_{y(t)}, \Sigma\right)$ with $\mu_y \sim \mathcal{F}_m$ for each y. Likewise, under a discriminative causal model where y depends on x, covariate shift can be modeled by $1/f$ noise in $p(x)$, and concept shift can be modeled by $1/f$ noise in $p(y|x)$.

4.2 KALMAN FILTER

We consider Bayesian methods that adopt the above construction as a generative model. Equations 13 and 14 describe a linear dynamic system with state $Z = [z_1, \ldots, z_n]^\top$. If ξ is directly observed at discrete intervals, then optimal Bayesian online prediction of each $\xi(t)$ based on all preceding observations can be implemented by a KF over Z (Kording et al., 2007).

This framework suggests a method for statistical optimization, including training NNs, in $1/f$ environments. Assume we receive observations $x(t), y(t)$ that we wish to model with a statistical model h that is parameterized by $\theta \in \mathbb{R}^m$:

$$y(t) = h(x(t), \theta(t)). \tag{15}$$

For example, h may be a NN with weights θ, input x, and output y. We further assume θ follows a $1/f$ process, modeled as $\theta = \sum_{i=1}^{n} \theta_i$, where each θ_i is an Ornstein-Uhlenbeck process in $\mathbb{R}^m$ with timescale τ_i. Then we can define $\Theta = [\theta_1; \ldots; \theta_n] \in \mathbb{R}^{nm}$ and do inference over Θ.

When h is linear in θ (and hence in Θ), such as in the regression task and 1-layer perceptron of Section 5.1, exact inference is possible with a standard KF. For a general h, such as a multilayer NN, we use an EKF. The EKF makes a local linear approximation of h based on its Jacobian, the matrix of gradients of predictions $\hat{y}$ with respect to θ (Appendix E). We use Ollivier's generalization of the EKF (Ollivier, 2018) that replaces Gaussian observation noise with any exponential family $p(y|\hat{y})$, which is better suited for modeling discrete outcomes such as the classification task of Section 5.2.

4.3 VARIATIONAL APPROXIMATION

Finally, we derive a variational approximation of the EKF that turns out to be equivalent to the multiscale optimizer and hence is suitable for efficient implementation in large NNs. Recall that a KF or EKF maintains an iterative prior based on all previous observations:

$$p\left(\Theta(t)|\boldsymbol{x}_{<t}, \boldsymbol{y}_{<t}\right) \sim \mathcal{N}(w(t), S(t)). \tag{16}$$

The mean, $w(t)$, is the vector of current weight estimates in the network, while $S(t)$ captures their joint uncertainty and hence determines updates (as a preconditioner; see Appendix D). We approximate this distribution by one in which $S(t) \approx \tilde{S}(t)$, where $\tilde{S}(t)$ is constrained to be diagonal (see

Appendix F). This reduces the complexity from $\mathcal{O}(m^2n^2)$ to $\mathcal{O}(mnk)$ (the size of the Jacobian, where k is the size of the output layer). Under this approximation, each w_{ij} (a weight-timescale pair) is updated independently, based on the current loss gradient. Thus the variational method amounts to decomposing every weight in the network as a sum of subweights, $w_j = \sum_i w_{ij}$, with decay rates τ_i and learning rates coming from $\tilde{S}(t)$. Note that this multiplexing is not expensive relative to current optimizers (e.g., Adam (Kingma & Ba, 2015)), which also store multiple variables for each weight.

The simplest case of the $1/f$ KF is a model that tracks a single $1/f$ variable, with no inputs or latent variables. That is, $y = \theta$ and $y \in \mathbb{R}^1$. Appendix F.1 derives the update rule for the variational approximation in this case as

$$w_i(t+1) = e^{-1/\tau_i} w_i(t) + \alpha_i(t)\,(y_i(t) - \hat{y}_i(t)) \tag{17}$$

with learning rates given by

$$\alpha_i(t) = \frac{e^{-1/\tau_i}\tilde{S}^2_{i,i}(t)}{\sum_j \tilde{S}^2_{j,j}(t)}. \tag{18}$$

For the EKF with a general nonlinear model $h(x, \theta)$, the update is derived in Appendix F.3 as

$$w_i(t+1) = e^{-1/\tau_i} s_i'(t)^2 s_i(t)^{-2} w_i(t) - e^{-1/\tau_i} s_i'(t)^2 \partial_{w_i}\mathcal{L} \tag{19}$$

where $\mathrm{diag}(s'^2)$ is the diagonal approximation of the posterior variance after observing $y(t)$.

Thus, in all cases, the subweights learn independently according to SGD, with different decay rates and learning rates. This is a special case of the multiscale optimizer, with and additional mechanism to adapt the learning and decay rates based on current uncertainty.

5 Simulation tests

5.1 Regression task

As a simple demonstration, we created a linear regression task with 10 features (including a bias term), in which the true weights θ vary over time according to $1/f$ noise. The outcome is generated by the model $h(x, \theta) = x^\top \theta$. This is a one-layer perceptron, which we write as $\hat{y} = x^\top w$ in order to distinguish network variables from true environment parameters. We assume square loss, $\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$.

We tested two baseline learning models, representing common heuristic practices with nonstationary data (Ditzler et al., 2015; Parisi et al., 2019). First, we tested a batch model that uses a fixed memory horizon H. To generate a prediction on step t, the batch learner fits the perceptron to trials $t - H$ through $t - 1$. Figure 3A shows performance is U-shaped: accuracy suffers with short horizons because of sampling error, but it also suffers from longer horizons because older observations are less valid. Second, we tested SGD, in which the weights are updated once after each step t, based

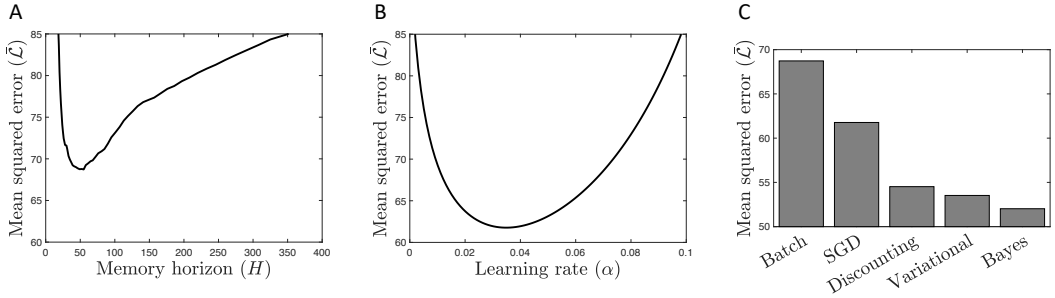

Figure 3: Regression task with $1/f$ dynamics. A: Batch learner. B: Stochastic gradient descent. C: Model comparison.

on $x(t)$ and $y(t)$. Figure 3B shows performance is best with an intermediate learning rate, which roughly corresponds to assuming the environment changes on a single characteristic timescale (see Appendix C).

As applied to the perceptron, the Bayesian $1/f$ model described in Section 4.2 decomposes the weight for each feature j into subweights, $w_j = \sum_i w_{ij}$, and tracks the w_{ij} jointly with a KF (generalizing (Dayan & Kakade, 2000)). The subweights combine to predict the outcome on each step: $\hat{y}(t) = \sum_{ij} x_i w_{ij}$. Relative to blind prediction (guessing the overall mean of y on every trial), this exact Bayesian solution explains 54.0% more variance in the outcome than the best batch learner, and 25.7% more than the best parameterization of SGD (Figure 3C). Moreover, the variational model that constrains the KF covariance matrix to be diagonal performs nearly (96.9%) as well as the full Bayesian model.

Finally, we tested a discounting method, similar to the batch learner except all past trials were used for training, discounted by a function of lag. Because the $1/f$ environment has power-law correlations, we weighted each observation $t - k$ by k^{-a} and optimized a. This method also significantly outperforms windowed batch and SGD, showing that accounting for an environment's autocorrelation function can achieve much of the advantage of the Bayesian approach. Nevertheless, the full KF and variational methods still outperform discounting, by 5.5% and 2.2% respectively.

5.2 Classification task

Next, we investigated a 10-way classification task with 10 features (including a bias term). The data were generated by first sampling the class, $y(t) \sim \mathrm{softmax}(e(t))$, and then sampling the feature vector, $x(t)|y(t) \sim \mathcal{N}(\mu_{y(t)}, I)$. We assumed cross-entropy loss, $\mathcal{L}(y, \hat{y}) = -\log \hat{y}_y$, where $\hat{y}$ is the vector of predicted class probabilities. The logits e_j and the feature-class means μ_{ij} were independently generated from $1/f$ processes, so that there was continual distribution shift in both $p(y)$ and $p(x|y)$. Scaling of e and μ was chosen to equate to maximum possible performance based on perfect knowledge of either one alone (both yielding average loss $\bar{\mathcal{L}} \approx 1$).

The model for the task, h, was a perceptron with a softmax output layer: $\hat{y} = \mathrm{softmax}(x^\top w)$, where w is a matrix of learnable feature-class weights. The nonlinearity of the softmax layer makes an exact KF inapplicable, so instead we use an EKF as a first step toward assessing viability of the method on arbitrary architectures. Note also that the network architecture is deliberately misspecified for the environment: The true posterior logits are quadratic in x, whereas the network is limited to linear functions. This choice enables us to test the robustness of the EKF approach.

The batch method trained the network on trials $t - H$ through $t - 1$ until convergence before predicting $y(t)$. Weight decay was included for regularization, with the decay rate optimized for each H. Figure 4A shows a U-shaped pattern of performance, reflecting the tradeoff between sampling error and stale data. We also used weight decay with SGD, optimized for each learning rate. Figure 4B again shows a U-shaped pattern of performance.

The variational EKF uses a local linear approximation of the network, $\frac{\partial \hat{y}}{\partial w}$, and the variance of $T(y)$ conditioned on $\hat{y}$ (where $T(y)$ is a one-hot encoding) as inputs to the standard KF update equations (see Appendix F.3). We enforce the diagonal approximation of the covariance matrix

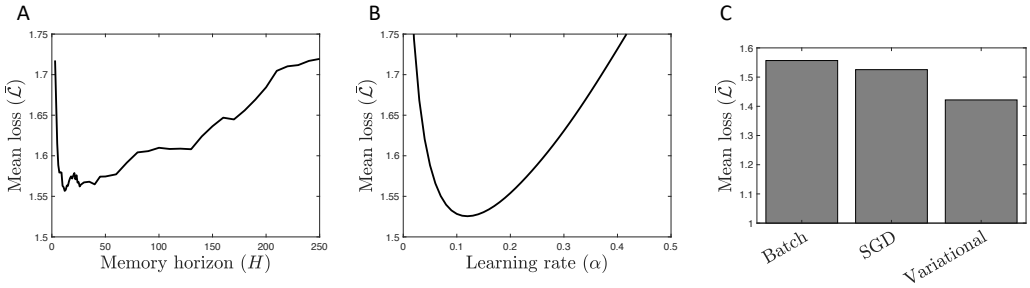

Figure 4: Classification task with $1/f$ dynamics. Loss is negative log-likelihood. A: Batch learner. B: Stochastic gradient descent. C: Model comparison.

when calculating the posterior after each observation. The optimal approximation uses the diagonal of the precision matrix but can be calculated without matrix inversion, which is relevant to scaling up to large networks. We also applied ℓ_2 regularization to the prior for the next timestep, on par with the SGD and batch learners. Figure 4C shows the variational method outperforms the other two.

6 Conclusions

The equivalences proved here, and others provable by similar methods, simplify the space of possible models and provide a unified framework for investigating learning at multiple timescales. A key feature of the multiscale learning model is that the components all evolve independently, both in weight decay between updates and in updating from their individual loss gradients. This makes the model both conceptually and computationally simpler than past ones. Simulations reported in a companion paper, using an online prediction task governed by $1/f$ noise in latent environmental parameters, show that the multiscale learner significantly outperforms SGD and batch learning with finite memory horizon, and nearly matches Bayes-optimal performance without the computational overhead (Anonymous, 2022). Taking all of these results together, the multiscale learner enjoys a combination of normative, heuristic, and biological justification, good performance, and computational efficiency. Future work will incorporate this model into state-of-the-art deep NN architectures, to explore whether it yields ML systems that are better able to exploit temporal structure in natural environments.

A further contribution of this work is the characterization of momentum as a fast weight with negative learning rate. Although this connection may seem counterintuitive, it can be understood as follows: When $\varepsilon_{\text{fast}} < 0$, the subweights learn in opposite directions, with the latent knowledge in w_{slow} overshooting the observable knowledge in $w = w_{\text{slow}} + w_{\text{fast}}$. As w_{fast} decays upward toward 0, w catches up to w_{slow}, so that the model appears to continue learning from past input. Momentum acts in the same manner, as a form of latent knowledge of past input that continues to drive later updates. This correspondence is exact under the translation in Section 3.2. Put differently, learning at multiple timescales is motivated by an expectation of positive autocorrelation in the environment, whereas momentum is effective at smoothing out negative autocorrelation in the gradient signal (e.g., along highly curved directions in weight space). The inherent opposition between these considerations may be reconcilable using the present results that place momentum learning within the broader framework of multiscale learning.

Our results offer an initial demonstration of how online learning performance in nonstationary environments can be improved by incorporating a model of temporal structure. The Bayesian $1/f$ model amounts to distributing knowledge across multiple timescales, and the variational approximation enables implementation in a neural network using subweights with different learning and decay rates. The latter method is closely related to models of learning at multiple timescales that have been proposed in psychology and neuroscience (Anonymous, 2022). It is also similar to power-law discounting of past observations, in that a sum of subweights decaying at different exponential rates approximates power-law decay. Nevertheless, the variational method was seen to perform better, presumably because of the interaction among subweights (via learning from a shared gradient) that approximates Bayesian allocation of knowledge to different timescales. Another possible variational method is to assume a block-diagonal covariance matrix, in which covariance information is maintained only between subweights (timescales) within each weight, so that computational complexity still scales linearly with network size. The next step will be to investigate how well these methods scale up to large networks, and whether they confer an advantage in real online-learning domains.

REFERENCES

John R. Anderson. Acquisition of cognitive skill. *Psychological Review*, 89:369–406, 1982.

John R. Anderson and Lael J. Schooler. Reflections of the environment in memory. *Psychological Science*, 2(6):396–408, 1991.

A. Anonymous. Learning at multiple timescales. In *Advances in Neural Information Processing (Submitted to MemARI workshop)*, 2022.

Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

Marcus K. Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19, 2016. doi: 10.1038/nn.4401. URL https://doi.org/10.1038/nn.4401.

Nicholas J. Cepeda, Edward Vul, Doug Rohrer, John T. Wixted, and Harold Pashler. Spacing effects in learning: A temporal ridgeline of optimal retention. *Psychological Science*, 19(11):1095–1102, 2008.

Peter Dayan and Sham Kakade. Explaining away in weight space. *Advances in neural information processing systems*, 13, 2000.

Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015. doi: 10.1109/MCI.2015.2471196.

Iddo Eliazar and Joseph Klafter. A unified and universal explanation for lévy laws and 1/f noises. *Proceedings of the National Academy of Sciences*, 106(30):12251–12254, 2009.

Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th annual conference of the cognitive science society*, pp. 177–186, 1987.

Marc W. Howard and Michael J. Kahana. A distributed representation of temporal context. *Journal of mathematical psychology*, 46(3):269–299, 2002.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018. PMLR 80.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, 2019. PMLR 97.

Marvin S. Keshner. 1/f noise. *Proceedings of the IEEE*, 70(3):212–218, 1982.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Konrad P. Kording, Joshua B. Tenenbaum, and Reza Shadmehr. The dynamics of memory as a consequence of optimal adaptation to a changing body. *Nature Neuroscience*, 10(6):779–786, 2007. doi: 10.1038/nn1901. URL https://doi.org/10.1038/nn1901.

Michael C. Mozer, Harold Pashler, Nicholas Cepeda, Robert V. Lindsey, and Ed Vul. Predicting the optimal spacing of study: A multiscale context model of memory. *Advances in neural information processing systems*, 22, 2009.

Yann Ollivier. Online natural gradient as a kalman filter. *Electronic Journal of Statistics*, 12(2): 2930–2961, 2018.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12 (1):145–151, 1999.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland (eds.), *Parallel distributed processing, Vol. 1*, pp. 318–362. MIT Press, Cambridge, MA, 1986.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012.

JER Staddon, IM Chelaru, and JJ Higa. Habituation, memory and the brain: The dynamics of interval timing. *Behavioural Processes*, 57(2-3):71–88, 2002.

Richard S Sutton. Gain adaptation beats least squares. In *Proceedings of the 7th Yale workshop on adaptive and learning systems*, volume 161, pp. 166, 1992.

Matthew H. Wilder, Matt Jones, Alaa A. Ahmed, Tim Curran, and Michael C. Mozer. The persistent impact of incidental experience. *Psychonomic bulletin & review*, 20(6):1221–1231, 2013.

John T. Wixted and Ebbe B. Ebbesen. Genuine power curves in forgetting: A quantitative analysis of individual subject forgetting functions. *Memory & Cognition*, 25(5):731–739, 1997.

A DIAGONALIZATION OF BENNA-FUSI TRANSITION MATRIX

This section provides details on diagonalizing the transition matrix of the update rule for the Benna & Fusi (2016) model, and the corresponding reparameterization in terms of the eigenbasis.

First, reparameterize the Benna-Fusi model so that its transition matrix is symmetric, as follows. Recursively define

$$b_k = \begin{cases} 1 & k = 1 \\ \frac{b_{k-1}c_k}{c_{k-1}} & 1 < k \leq n \end{cases} \tag{20}$$

and write the state of the Benna-Fusi synapse as

$$\phi = (\sqrt{b_k}u_k)_{1 \leq k \leq n}. \tag{21}$$

The update becomes

$$\phi(t+1) = \Gamma\phi(t) + d(t) \tag{22}$$

with

$$\Gamma_{k,k} = 1 - \frac{g_{k-1} + g_k}{C_k} \tag{23}$$

$$\Gamma_{k-1,k} = \Gamma_{k,k-1} = \frac{g_{k-1}}{\sqrt{C_{k-1}C_k}}. \tag{24}$$

Symmetry of Γ implies it has an orthonormal eigenbasis, $\{\psi_1, \ldots, \psi_n\}$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Because the scaling of eigenvectors is arbitrary, we can assume $\psi_{i,1} = 1$ for all i.

To translate the eigenbasis back to u, define $B = \mathrm{diag}(\sqrt{b})$ so that $\phi = Bu$ and $T = B^{-1}\Gamma B$. It is then easily verified that

$$V_{\cdot i} = B^{-1}\psi_i, \tag{25}$$

is an eigenvector of T with eigenvalue λ_i. Therefore $T = V\Lambda V^{-1}$ as claimed in the main text. Note that the choice $\psi_{i,1} = 1$ entails $V_{1,i} = 1$ (because $b_1 = 1$). Finally, $\Psi = [\psi_1, \ldots, \psi_n]$ is invertible because it comprises a basis, and therefore so is $V = B^{-1}\Psi$. Therefore the reparameterization $u \mapsto w = V^{-1}u$ is well-defined.

B GENERATIVE MODEL FOR $1/f$ NOISE

Consider a single Ornstein-Uhlenbeck (OU) process $z(t)$ described by Equation 13 with $\sigma = 1$. The covariance function of z is given by

$$\mathbb{E}[z(t)z(t+s)] = \int_{-\infty}^{t} e^{-(t-t')/\tau} e^{-(t+s-t')/\tau} \mathrm{d}t' \tag{26}$$

$$= \frac{\tau}{2} e^{-s/\tau}. \tag{27}$$

Note that this expression decays exponentially with the lag s, yielding short-range correlations. The power spectrum of z is the Fourier transform of the covariance function:

$$P_z(f) = \int_{\mathbb{R}} \frac{\tau}{2} e^{-|s|/\tau} e^{-2\pi i f s} \mathrm{d}s \tag{28}$$

$$= \frac{1}{\tau^{-2} + (2\pi f)^2} \tag{29}$$

where $2\pi f$ is angular frequency.

To model $1/f$ noise, we define a continuous mixture model over a family of OU processes $(z_\tau)_{0<\tau\leq T}$ for some large T:

$$y(t) = \int_0^T 2\tau^{-1} z_\tau(t) \mathrm{d}\tau. \tag{30}$$

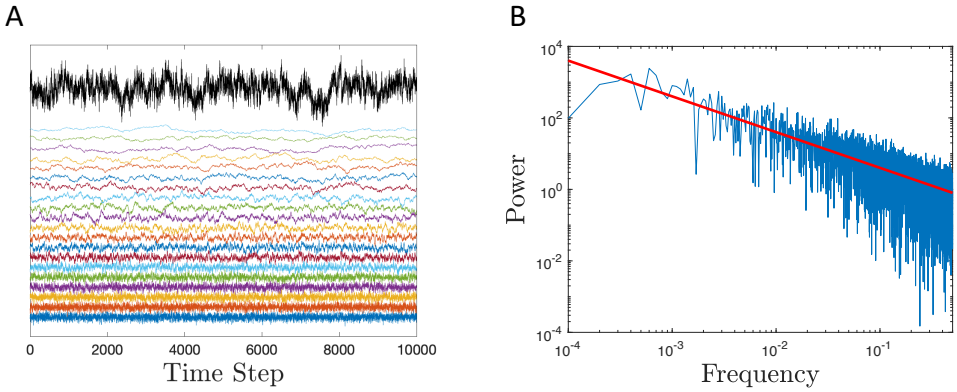

Figure 5: A: Construction of $1/f$ noise (black trajectory at top) as a sum of OU processes with different time constants (colored trajectories). Vertical offsets are applied as a visual aid for discriminating the curves. B: The power spectrum of the aggregate trajectory in log-log coordinates. The red line has slope 1.

The power spectrum of y is then a mixture over the component spectra P_{z_τ}:

$$P_y(f) = \int_0^T 4\tau^{-2} P_{z_\tau}(f) \mathrm{d}\tau \tag{31}$$

$$= \int_0^T \frac{4\tau^{-2}}{\tau^{-2} + (2\pi f)^2} \mathrm{d}\tau \tag{32}$$

$$= \frac{2}{\pi f} \tan^{-1}(2\pi f T) \tag{33}$$

which is approximately $1/f$ for all but the lowest frequencies $f \lesssim 1/T$.

We next define a discrete approximation of the continuum mixture model Equation 30,

$$y(t) = \sum_i z_i(t), \tag{34}$$

where z_i has timescale τ_i and scaling parameter σ_i. To approximate a $1/f$ spectrum, $\sigma_i^2(\tau_{i+1} - \tau_i)$ should scale as τ_i^{-2}, the squared weight density in Equation 30 (because power is additive and proportional to σ^2). For example, the τ_i could be arithmetically spaced with $\sigma_i = \tau^{-1}$. Instead, we assume geometric spacing, with n components defined by $\tau_i = \gamma^i$ and $\sigma_i = \tau_i^{-1/2}$. Figure 5 illustrates this construction and exemplifies the accuracy of the discrete approximation.

To generate $1/f$ noise in $\mathbb{R}^m$, an independent copy of this construction is sampled for each dimension. The resulting joint process is rotationally symmetric, a property inherited from OU processes. That is, any linear combination $\sum_j c_j y_j$ is a 1d $1/f$ process.

C Kalman filter for a single timescale

Start again with a single OU process z with $\sigma = 1$, and assume it is observed at unit intervals $(y_t)_{t\in\mathbb{N}}$ with observation noise η^2. Assume a conjugate iterative prior:

$$z(t)|\boldsymbol{y}_{<t} \sim \mathcal{N}\left(w(t), s(t)^2\right). \tag{35}$$

The posterior after each observation is

$$z(t)|\boldsymbol{y}_{\leq t} \sim \mathcal{N}\left(\frac{\eta^2 w(t) + s(t)^2 y(t)}{s(t)^2 + \eta^2}, \frac{s(t)^2 \eta^2}{s(t)^2 + \eta^2}\right). \tag{36}$$

Evolving the process to time $t+1$ amounts to decay by $e^{-1/\tau}$ and accumulation of new noise. At each time $t' \in [t, t+1]$, variance from the noise appearing at time t' (i.e., from $\mathrm{d}W(t')$ in Equation

13) decays by a factor $e^{-2(t+1-t')/\tau}$ by time $t+1$. Therefore the total accumulated variance is:

$$\int_t^{t+1} e^{-2(t+1-t')/\tau} \mathrm{d}t' = \frac{\tau}{2}\left(1 - e^{-2/\tau}\right). \tag{37}$$

Therefore the prior for the next observation is

$$z(t+1)|\boldsymbol{y}_{\leq t} \sim \mathcal{N}\left(e^{-1/\tau}\frac{\eta^2 w(t) + s(t)^2 y(t)}{s(t)^2+\eta^2}, e^{-2/\tau}\frac{s(t)^2\eta^2}{s(t)^2+\eta^2} + \frac{\tau}{2}\left(1-e^{-2/\tau}\right)\right) \tag{38}$$

$$= \mathcal{N}\left(w(t+1), s(t+1)^2\right). \tag{39}$$

We thus obtain the recursion

$$w(t+1) = e^{-1/\tau}\frac{\eta^2 w(t) + s(t)^2 y(t)}{s(t)^2+\eta^2} \tag{40}$$

$$= e^{-1/\tau}\left(w(t) + \alpha(y(t) - w(t))\right). \tag{41}$$

This is a temporal-difference learning rule, or gradient descent on squared error $\frac{1}{2}(y(t) - w(t))$, with learning rate $\alpha(t) = \frac{s(t)^2}{s(t)^2+\eta^2}$. This connection between temporal-difference learning and the Kalman filter is well known (Sutton, 1992).

The steady state for s^2 is given by the solution to $s(t)^2 = s(t+1)^2$, a quadratic with one positive root. If $s(0)^2$ is initialized to this value, then $\alpha(t)$ will be constant. Note that this model (and the $1/f$ model below) generalizes to irregularly spaced observations, in which case one can derive how the optimal learning rate should vary on each step.

D KALMAN FILTER OVER $1/f$ NOISE

Consider the joint process $Z = [z_1, \ldots, z_n]$, where the z_i are defined as at the end of Appendix B. Assume the summed process is observed at unit time intervals, $y(t)_{t\in\mathbb{N}}$. For simplicity and in contrast to Appendix C, we assume no observation noise, because the shortest timescales (z_1, etc.) can play this role.

Because $y(t)$ is conditionally independent of the history given (and indeed is fully determined by) $Z(t)$, it suffices to compute the posterior for the latter. Write the iterative prior for Z as

$$Z(t)|\boldsymbol{y}_{<t} \sim \mathcal{N}\left(w(t), S(t)\right), \tag{42}$$

implying an optimal (least-squares prediction) of $\hat{y}(t) = \sum_i w_i(t)$. The posterior after observing $y(t)$ is the intersection of the prior with the hyperplane $\sum_i z_i(t) = y(t)$:

$$Z(t)|\boldsymbol{y}_{\leq t} \sim \mathcal{N}\left(w(t) + (y(t) - \hat{y}(t))\frac{S(t)\mathbf{1}}{\mathbf{1}^\top S(t)\mathbf{1}}, \left(PS(t)^{-1}P + \mathbf{1}\mathbf{1}^\top\right)^{-1}P\right). \tag{43}$$

Here, $\mathbf{1}$ is the vector with all elements equal to 1, and $P = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the orthogonal projector.

The prior for the next timestep is then obtained by applying decay and adding variance from the noise accumulated over the intervening interval. Define D as the diagonal matrix of decay factors, e^{-1/τ_i}, and N as the diagonal matrix of added noise $\frac{1}{2}(1 - e^{-2/\tau_i})$, obtained by multiplying the RHS of Equation 37 by $\sigma_i^2 = \tau_i^{-1/2}$. Then we have the update equations for the exact Bayesian model:

$$w(t+1) = D\left(w(t) + (y(t) - \hat{y}(t))\frac{S(t)\mathbf{1}}{\mathbf{1}^\top S(t)\mathbf{1}}\right) \tag{44}$$

and

$$S(t+1) = D\left(PS(t)^{-1}P + \mathbf{1}\mathbf{1}^\top\right)^{-1}PD + N. \tag{45}$$

For the perceptron model in Section 5.1, the state of the environment is described by $\Theta = (\theta_{ij})_{ij}$, where i indexes timescales and j indexes features. We treat ij as a single composite index, so that Θ is a vector. As above, write the iterative prior as

$$\Theta(t)|\boldsymbol{x}_{<t}, \boldsymbol{y}_{<t} \sim \mathcal{N}\left(w(t), S(t)\right). \tag{46}$$

Let X be a multiplexed copy of the input x, so that $X_{ij} = x_j$. The optimal prediction in the regression task is thus $\hat{y}(t) = X(t)^\top w(t)$. The posterior after observing $y(t)$ is the intersection of the prior with the hyperplane $X(t)^\top \Theta(t) = y(t)$:

$$\Theta(t)|\boldsymbol{x}_{\leq t}, \boldsymbol{y}_{\leq t} \sim \mathcal{N}\left(w(t) + (y(t) - \hat{y}(t))\,\frac{S(t)X(t)}{X(t)^\top S(t)X(t)},\right.$$
$$\left.\left(P_{X(t)}S(t)^{-1}P_{X(t)} + X(t)X(t)^\top\right)^{-1} P_{X(t)}\right), \tag{47}$$

where $P_X = I - XX^\top / X^\top X$ is the projector orthogonal to X.

Generalizing the definitions above, let D and N be the diagonal matrices of decay factors and noise accumulation, $D_{ij,ij} = e^{-1/\tau_i}$ and $N_{ij,ij} = \frac{1}{2}(1 - e^{-2/\tau_i})$. Applying these to Equation **??** to obtain the prior for the next time step yields the update equations for the exact Bayesian model of the regression task:

$$w(t+1) = D\left(w(t) + (y(t) - \hat{y}(t))\,\frac{S(t)X(t)}{X(t)^\top S(t)X(t)}\right) \tag{48}$$

$$S(t+1) = D\left(P_{X(t)}S^{-1}(t)P_{X(t)} + X(t)X(t)^\top\right)^{-1} P_{X(t)}D + N. \tag{49}$$

Equation 48 exemplifies how the variance matrix, $S(t)$, can be thought of a preconditioner of the gradient, $X(t)(y(t) - \hat{y}(t))$.

E Extended Kalman filter

Given a nonlinear model $h(x, \theta)$, the EKF maintains an iterative prior over Θ as in Equation 46, and updates that prior by linearizing h about $\Theta = w(t)$:

$$h(x(t), \Theta) \approx h(x(t), w(t)) + J_h(\Theta - w(t)). \tag{50}$$

Here, $J_h = \frac{\partial \hat{y}}{\partial \Theta}$ is the Jacobian matrix of h, evaluated at $\Theta = w(t)$.

Following (Ollivier, 2018), the EKF assumes the observation $y(t)$ is governed by some exponential family $P(y|\eta)$, with sufficient statistics $T(y)$, and it takes the model's output $\hat{y} := h(x(t), \theta(t))$ to encode the predicted mean parameter of that family: $\hat{y}(t) = \mathbb{E}_{y(t)\sim P(\cdot|\eta)}\left[T(y(t))\right]$. It then approximates the conditional distribution of the sufficient statistics as a Gaussian,

$$p(T(y)|\hat{y}) \approx \mathcal{N}\left(\hat{y}, R_{\hat{y}}\right), \tag{51}$$

where $R_{\hat{y}} = \mathrm{Var}\left(y|\hat{y}\right)$ is the conditional variance.

For example, when h is the 2-layer (perceptron+softmax) network in Section 5.2, the output of the network is a vector $\hat{y}$ of class probabilities, and the sufficient statistics $T(y)$ are a one-hot vector. For numerical stability, we exclude the final element of $\hat{y}$ and $T(y)$, which are determined by the other elements. The conditional outcome variance is given by

$$\left[R_{\hat{y}}\right]_{i,j} = \begin{cases} \hat{y}_i(1 - \hat{y}_i) & i = j \\ -\hat{y}_i\hat{y}_j & i \neq j. \end{cases} \tag{52}$$

Under the approximations of Equations 50 and 51, the posterior is given by

$$\Theta(t)|\boldsymbol{x}_{\leq t}, \boldsymbol{y}_{\leq t} \sim \mathcal{N}\Big(w(t) + S(t)J_h^\top\left(J_hS(t)J_h^\top + R_{\hat{y}(t)}\right)^{-1}(T(y(t)) - \hat{y}(t)),$$
$$S(t) - S(t)J_h^\top\left(J_hS(t)J_h^\top + R_{\hat{y}(t)}\right)^{-1} J_hS(t)\Big). \tag{53}$$

Applying decay (D) and accumulated noise (N) as in Appendix D to obtain the prior for the next time step yields the update equations for the EKF:

$$w(t+1) = D\left(w(t) + S(t)J_h^\top\left(J_hS(t)J_h^\top + R_{\hat{y}(t)}\right)^{-1}(T(y(t)) - \hat{y}(t))\right) \tag{54}$$

$$S(t+1) = D\left(S(t) - S(t)J_h^\top\left(J_hS(t)J_h^\top + R_{\hat{y}(t)}\right)^{-1} J_hS(t)\right)D + N. \tag{55}$$

F VARIATIONAL APPROXIMATION

We derive variational approximations of the KF and EKF models, with $S(t) \approx \tilde{S}(t)$ where $\tilde{S}(t) := \mathrm{diag}(s(t)^2)$. Thus the resulting algorithms need only track the individual variance terms in $s(t)$ rather than the full covariance matrix. Moreover, the update equations (62,63,67,68,72,73) all avoid matrix inversion—even though this might initially seem necessary from Equation 56—a property that may be relevant for efficient scaling.

F.1 UNCUED INFERENCE

We begin with the simple Bayesian $1/f$ model in Equations 44 and 45, where there are no predictors and the model merely tracks an observable $y(t)$. Given an arbitrary Gaussian distribution, the variational approximation (in the sense of minimizing Kullback-Leibler divergence) by another Gaussian with diagonal covariance matrix is obtained by taking the diagonal of the original distribution's precision matrix. Thus in the present case we have

$$s_i(t+1)^{-2} = \left[S(t+1)^{-1}\right]_{i,i} \tag{56}$$

where $S(t+1)$ is given by Equation 45 with $S(t)^{-1}$ replaced by $\tilde{S}(t)^{-1}$ (the inductive assumption). To calculate $S(t+1)^{-1}$ under this assumption, we first observe the identity

$$\left(P \ \mathrm{diag}\left(s^{-2}\right) P + \mathbf{1}\mathbf{1}^\top\right)^{-1} P = \mathrm{diag}\left(s^2\right) - \frac{s^2 {s^2}^\top}{\sum_i s_i^2}. \tag{57}$$

Therefore Equation 45 becomes

$$S(t+1) = D\tilde{S}(t)D + N - \frac{(Ds(t)^2)(Ds(t)^2)^\top}{\sum_i s_i(t)^2} \tag{58}$$

$$:= \mathrm{diag}(a) - \frac{bb^\top}{c}. \tag{59}$$

where $\mathrm{diag}(a) = D\tilde{S}(t)D + N$, $b = Ds(t)^2$, and $c = \sum_i s_i(t)^2$. We next use the identity

$$\left(\mathrm{diag}\left(a\right) - \frac{bb^\top}{c}\right)^{-1} = \mathrm{diag}\left(\frac{1}{\sqrt{a}}\right)\left(I + \frac{\left(a^{-1/2} \circ b\right)\left(a^{-1/2} \circ b\right)^\top}{c - \sum \frac{b_i^2}{a_i}}\right)\mathrm{diag}\left(\frac{1}{\sqrt{a}}\right), \tag{60}$$

implying the diagonal elements are

$$\left[\left(\mathrm{diag}\left(a\right) - \frac{bb^\top}{c}\right)^{-1}\right]_{i,i} = \frac{1}{a_i}\left(1 + \frac{\frac{b_i^2}{a_i}}{c - \sum \frac{b_j^2}{a_j}}\right). \tag{61}$$

Combining Equations 56, 59, and 61 gives the final form of the variational update:

$$s_i(t+1)^2 = \frac{\left(s_i(t)^2 e^{-1/\tau_i} + \sinh \frac{1}{\tau_i}\right)^2 Z}{\left(s_i(t)^2 + e^{1/\tau_i} \sinh \frac{1}{\tau_i}\right) Z + s_i(t)^4} \tag{62}$$

with

$$Z = \sum_j \frac{\frac{1}{2} s_j(t)^2 \left(e^{2/\tau_j} - 1\right)}{s_j(t)^2 + \frac{1}{2}\left(e^{2/\tau_j} - 1\right)}. \tag{63}$$

This update converges exponentially to a unique fixed point. Numerical simulations confirm that, in this limit, s_i^2 is larger for smaller τ_i, meaning faster learning rates for shorter timescales. If the prior variances are initialized at the fixed point then they are constant throughout learning.

Under this variational approximation, the update for the mean in Equation 44 becomes

$$w_i(t+1) = e^{-1/\tau_i}\left(w_i(t) + \alpha_i(t)\left(y_i(t) - \hat{y}_i(t)\right)\right) \tag{64}$$

where the learning rates are given by

$$\alpha_i(t) = \frac{s_i(t)^2}{\sum_j s_j(t)^2}. \tag{65}$$

That is, the subweights learn independently according to SGD, with different decay rates and learning rates.

F.2 General Kalman filter

To derive the variational model for the regression task, we apply the analysis of Section F.1 to the KF update in Equations 48 and 49. Paralleling the derivation of Equation 58, the variance update in Equation 49 can be written as

$$S(t+1) = D\tilde{S}(t)D + N - \frac{(D(x(t) \circ s(t)^2))(D(x(t) \circ s(t)^2))^\top}{\sum_k x_k(t)^2 s_k(t)^2}. \tag{66}$$

To avoid confusion, we use the index k to indicate a subweight, which in Appendix D was written as the composite ij. Paralleling the derivation of Equations 62 and 63, the update comes out to be

$$s_k(t+1)^2 = \frac{\left(s_k(t)^2 e^{-1/\tau_k} + \sinh \frac{1}{\tau_k}\right)^2 Z}{\left(s_k(t)^2 + e^{1/\tau_k} \sinh \frac{1}{\tau_k}\right) Z + x_k(t)^2 s_k(t)^4} \tag{67}$$

with

$$Z = \sum_k \frac{\frac{1}{2} x_k(t)^2 s_k(t)^2 \left(e^{2/\tau_k} - 1\right)}{s_k(t)^2 + \frac{1}{2}\left(e^{2/\tau_k} - 1\right)}. \tag{68}$$

Under the assumption of a diagonal covariance matrix, the mean update in Equation 48 becomes

$$w_{ij}(t+1) = e^{-1/\tau_i} w_{ij}(t) - \frac{e^{-1/\tau_i} s_{ij}^2(t)}{\sum_{i'j'} x_{j'}^2(t) s_{i'j'}^2(t)} x_j(t) \left(\hat{y}(t) - y(t)\right) \tag{69}$$

where $x_j(t)\left(y(t) - \hat{y}(t)\right)$ is the loss gradient for w_{ij}. Thus we have recovered a special case of the multiscale optimizer, with an additional mechanism that adapts the learning rates on each time step (via x and s).

F.3 General extended Kalman filter

To analytically derive the variational approximation for the general EKF, such as for classification task in Section 5.2, it turns out that we need to apply the variational approximation to the posterior in Equation 53, rather than to the iterative prior in Equation 55. Algebraic manipulation enables the posterior variance to be rewritten as

$$S(t) - S(t)J_h^\top \left(J_h S(t) J_h^\top + R_{\hat{y}(t)}\right)^{-1} J_h S(t) = (J_h^\top R_{\hat{y}(t)}^{-1} J + S(t)^{-1})^{-1}. \tag{70}$$

This form is convenient because it is in terms of precision, allowing us to read off the diagonal entries directly. That is, the variational approximation for the posterior variance is $\mathrm{diag}(s'(t)^2)$, with

$$s'_k(t)^2 = \left(\left[J_h^\top R_{\hat{y}(t)}^{-1} J_h\right]_{k,k} + s_k(t)^{-2}\right)^{-1}. \tag{71}$$

Here was have used the inductive assumption $S(t) \approx \mathrm{diag}(s(t)^2)$. Applying the transition from posterior on step t to prior on step $t+1$, we obtain the variance update for the variational model, replacing Equation 55:

$$s_k(t+1)^2 = D_{k,k}^2 s'_k(t)^2 + N_{k,k}. \tag{72}$$

The algebraic trick in Equation 70 also enables a more efficient update for the mean, replacing Equation 54:

$$w(t+1) = D\ \mathrm{diag}\left(s'(t)^2\right)\left(\tilde{S}(t)^{-1} w(t) + J_h^\top R_{\hat{y}(t)}^{-1}\left(T(y(t)) - \hat{y}(t) + J_h w(t)\right)\right). \tag{73}$$

Once again, this update rule is a special case of the multiscale optimizer, recognizing that $J_h^\top R_{\hat{y}(t)}^{-1}(\hat{y}(t) - J_h w(t) - T(y(t)))$ equals the gradient $\partial_w \mathcal{L}$.

G IMPLEMENTATION DETAILS

Latent parameters defining the task environments were generated as in Appendix B, using 20 timescales geometrically spaced from $\tau_1 = 1$ to $\tau_{20} = 1000$ (as in Figure 5A). Each component OU process was run for 10τ burn-in steps to ensure stationarity. The regression task was run for 10,000 trials, and the classification task for 1000 trials. The $1/f$ power spectrum was confirmed with a log-log plot (see Figure 5B).

The first feature in both tasks was a constant bias term, $x_1 \equiv 1$. The other 9 features were sampled as $\mathcal{N}(0, I)$ on each time step.

For the classification task, class logits were sampled from $1/f$ processes and then multiplied by 0.886 before entering into softmax to determine class probabilities. Feature-class means were sampled from $1/f$ processes and then multiplied by 0.224. These scaling factors were chosen so that perfect knowledge of either the prior probabilities or the feature-class means on every trial would yield ideal-observer performance of $\bar{\mathcal{L}} \approx 1$. Perfect knowledge of both would yield $\bar{\mathcal{L}} \approx 0.35$. These were merely guidelines for equating prior and likelihood information, as perfect knowledge of each source of information is not possible even with an optimal model of the dynamics.

All Bayesian and variational models assumed 10 timescales, geometrically spaced from $\tau_1 = 2$ to $\tau_{10} = 800$. This deliberate deviation from the generating process enabled testing of robustness, specifically the hypothesis that it is the aggregate $1/f$ character of the environment and of the model that matters, not the choice of component timescales used to approximate that character.