Predictive Coding Networks for Temporal Prediction

Beren Millidge[1], Mufeng Tang[1], Mahyar Osanlouy[2], and Rafal Bogacz[1]

[1]MRC Brain Network Dynamics Unit, University of Oxford, UK
[2]Auckland Bioengineering Institute, University of Auckland, NZ

May 12, 2023

Running title: Temporal predictive coding

Abstract

One of the key problems the brain faces is inferring the state of the world from a sequence of dynamically changing stimuli, and it is not yet clear how the sensory system achieves this task. A well-established computational framework for describing perceptual processes in the brain is provided by the theory of predictive coding. Although the original proposals of predictive coding have discussed temporal prediction, later work developing this theory mostly focused on static stimuli, and key questions on neural implementation and computational properties of temporal predictive coding networks remain open. Here, we address these questions and present a formulation of the temporal predictive coding model that can be naturally implemented in recurrent networks, in which activity dynamics rely only on local inputs to the neurons, and learning only utilises local Hebbian plasticity. Additionally, we show that predictive coding networks can approximate the performance of the Kalman filter in predicting behaviour of linear systems, and behave as a variant of a Kalman filter which does not track its own subjective pos-

terior variance. Importantly, predictive coding networks can achieve similar accuracy as the Kalman filter without performing complex mathematical operations, but just employing simple computations that can be implemented by biological networks. Moreover, we demonstrate how the model can be effectively generalized to non-linear systems. Overall, models presented in this paper show how biologically plausible circuits can predict future stimuli and may guide research on understanding specific neural circuits in brain areas involved in temporal prediction.

Author summary

While significant advances have been made in the neuroscience of how the brain processes static stimuli, the time dimension has often been relatively neglected. However, time is crucial since the stimuli perceived by our senses typically dynamically vary in time, and the cortex needs to make sense of these changing inputs. This paper describes a computational model of cortical networks processing temporal stimuli. This model is able to infer and track the state of the environment based on noisy inputs, and predict future sensory stimuli. By ensuring that these predictions match the following stimuli, the model is able to learn the structure and statistics of its temporal inputs. The model may help in further understanding neural circuits in sensory cortical areas.

Introduction

This paper is concerned with extending the theory of predictive coding to the problem of processing dynamically changing sequences of sensory inputs arriving over time. Predictive coding, which originated from an algorithm for compression in information theory (Spratling, 2017), was initially developed and applied to the analysis of the brain by Srinivasan, Laughlin, and Dubs (1982) and Mumford (1992) and then formalized into a general computational model of the cortex by Rao and Ballard (1999). The core intuition behind predictive coding is that the brain computes predictions of its observed input, and compares these predictions to the actually received input. The difference between the two is called the prediction error and quantifies how incorrect the brain's prediction was. The brain then

adjusts its neural activities and synaptic strengths to minimize prediction errors which ultimately results in more accurate predictions (Clark, 2015; Friston, 2005). Thus, solely by minimizing prediction errors, the brain is forced to learn a general *world model* which can generate accurate predictions of its incoming sensory input (Friston, 2003, 2005). Moreover, these prediction errors are computed *locally* between the local input to the neuron and the predictions it receives. This means that learning in predictive coding model requires only local information and can be accomplished in most cases with purely Hebbian synaptic plasticity (Bogacz, 2017; Buckley, Kim, McGregor, & Seth, 2017; Millidge, Tschantz, Seth, & Buckley, 2020).

Predictive coding has become an influential theoretical model for understanding cortical functions (Friston, 2003, 2005; Rao & Ballard, 1999; Walsh, McGovern, Clark, & O'Connell, 2020). In their original study, Rao and Ballard (1999) trained predictive coding networks to generate static images of natural scenes, and demonstrated that the network learnt receptive fields similar to those found in V1, as well as reproduced extra-classical receptive fields and end-stopping. Since then it has been demonstrated that predictive coding networks can explain many intriguing phenomena such as repetition suppression (Auksztulewicz & Friston, 2016), bistable perception (Hohwy, Roepstorff, & Friston, 2008; Weilnhammer, Stuke, Hesselmann, Sterzer, & Schmack, 2017), illusory motions (Watanabe, Kitaoka, Sakamoto, Yasugi, & Tanaka, 2018), retinal stabilization (Millidge & Shillcock, 2019) and even potentially attentional modulation of perception (Feldman & Friston, 2010; Kanai, Komura, Shipp, & Friston, 2015). Moreover, there has been much work matching the underlying neurophysiology of cortical microcircuits to the fundamental computations required by the predictive coding algorithm (Bastos et al., 2012; Keller & Mrsic-Flogel, 2018; Millidge, Seth, & Buckley, 2021), thus providing a potential low-level basis for the implementation of predictive coding in neural circuitry.

Alongside the aforementioned works that have successfully reproduced many neurophysiological phenomena, recent progresses in predictive coding have been concerned with machine learning tasks such as the classification or generation of static images (Millidge, Tschantz, & Buckley, 2020; Ororbia & Kifer, 2022;

Song, Lukasiewicz, Xu, & Bogacz, 2020; Sun & Orchard, 2020), and multiple lines of research have investigated the relationship between predictive coding and backpropagation, the driving force behind modern machine learning systems (Song et al., 2020; Whittington & Bogacz, 2017). However, most of these works are concerned with inputs that are independent and identically distributed (i.i.d.) samples from some dataset and are presented in batches to the predictive coding model in random order. However, the visual input to the brain is not like this. Instead, the brain receives a continually changing sensory stream conveying an endless sequence of individual images with much correlation and rich structure embedded in the timing of the sequence elements. Therefore, to better describe the information processing in the brain, predictive coding models must take into account one more crucial element: *time*.

There are several established algorithms in statistics and machine learning for sequence processing over time, but they typically require very complex computations that would be difficult for biological circuits to perform. Nevertheless, it is useful to consider them as a reference against which the performance of biologically plausible models can be assessed. When there is a linear relationship between the current and future states (and noise is assumed to be Gaussian), the optimal temporal predictions can be achieved by the Kalman filter (Kalman, 1960). For more complex nonlinear problems, one can employ recurrent neural networks (Jordan, 1990), which contain recurrent connections which maintain and update an internal hidden state over time. While these recurrent networks, and more advanced successors such as the Long-Short-Term-Memory (LSTM) (Hochreiter & Schmidhuber, 1997) can be very expressive and powerful, they are typically trained with backpropagation through time algorithm (BPTT) (Werbos, 1988; Williams, 1992; Williams & Zipser, 1995) which requires storing a history of all computations through a sequence and then 'unrolling' it sequentially backwards through time to make updates. This algorithm is biologically implausible, because the brain can only receive inputs in a sequential stream, and must be able to process them online – i.e. as the inputs are received, and seems unlikely to be able to unfold a precise sequence of computations in reverse. In this work, we focus our comparison to these statistical and machine learning models on the Kalman filter which is an

online algorithm, i.e. it processes data sequentially, as biological systems need to do.

Within the field of predictive coding, there are a few tracks of research that have considered incorporating the temporal dimension. Earlier works (Rao, 1997; Rao & Ballard, 1997) employed Kalman filtering to model the visual processing of dynamical sequences. Instead of using fixed transformation matrices which are assumed to be known as in Kalman filtering, these works introduced learning rules to model synaptic plasticity in neural circuits performing filtering. Friston (2008) have proposed the notion of extending predictive coding to use *generalized coordinates* (Friston, Stephan, Li, & Daunizeau, 2010; Friston, Trujillo-Barreto, & Daunizeau, 2008a) which model a dynamical state by including a set of temporal derivatives into the state vector and making predictions of these derivatives along with the current state. A few recent studies have also developed predictive coding models that perform well in various tasks involving temporal dependencies, such as those commonly examined in machine learning (Jiang & Rao, 2022; Kutschireiter, Surace, Sprekeler, & Pfister, 2017; Ororbia, Mali, Giles, & Kifer, 2020). However, the works mentioned above have departed from the simple and flexible architecture of classical predictive coding for static inputs (Rao & Ballard, 1999) in order to take into account the temporal dimension. For example, the pioneering work that introduced Kalman filtering into predictive coding (Rao, 1997; Rao & Ballard, 1997) has not described how the computations of Kalman filtering could be implemented in biological networks. Mapping of models employing generalized co-ordinates (Friston, 2008) on a neural circuit would require explicit hard-coding of the expected temporal dependencies between different dynamical orders. Other models include specially designed network features to aid the temporal processing, such as hyper-networks (Jiang & Rao, 2022), multiple sub-networks (Kutschireiter et al., 2017) or complex connectivity and neuron types (Ororbia et al., 2020).

In this paper, we propose a simple predictive coding network that also incorporates the temporal dimension, which we call *temporal predictive coding*. This model generates predictions not only about the current inputs but also about its own *future* neural responses, which is achieved by recurrent connections between neurons to transmit the prediction of one time-step to the next. The paper

makes three main contributions: First, we propose a predictive coding model that addresses the problem of temporal prediction, while inheriting from static predictive coding (Rao & Ballard, 1999) the simple and biologically plausible neural network implementations employing only local connectivity and Hebbian plasticity rules. Second, when the model is linear, we show that our model is a close approximation of the Kalman filtering model analytically, and has empirical performance comparable to Kalman filtering in benchmark filtering tasks while being computationally cheaper and requiring only biologically plausible operations. Also, unlike the Kalman filter, our model can learn the parameters of the system online. Finally, we extend the model to the nonlinear case and show promising performance on a number of nonlinear filtering and sequence prediction tasks. Overall, our model provides a possible computational mechanism underlying the cortical processing of dynamic inputs based on predictive coding, suggesting that the brain may learn to represent both static and continuous sensory observations using a single computational framework.

Models

The structure of the exposition in this section is as follows. Firstly, we present the underlying graphical structure of our proposed generative temporal predictive coding network i.e., a Hidden Markov Model (HMM). Next, we show that, with Gaussian assumptions on the HMM and certain parametric assumptions on the nonlinear generative processes, we can derive an objective function that is similar to the original predictive coding network (Rao & Ballard, 1999) but taking into account the temporal dimension. We then demonstrate that neural dynamics and plasticity rules can be derived via minimization of the objective function by gradient descent, and a corresponding training algorithm is presented. Finally, we show that the proposed computations and algorithms afford biologically plausible neural implementations in several different cases and discuss how they may be mapped to neural circuitry in the cortex. Since this paper contains much mathematical notation, for ease of reading, we have collected it in Table 1 for quick reference.

Notation	Meaning
y	Observation
x	Latent state
u	Control input
k	Observation 'frame'
t	Inference time
A	Dynamics matrix
B	Control matrix
C	Observation matrix
Σ_x	Process noise
Σ_y	Observation noise
f	Nonlinear function
$\hat{x}$	Inferred latent state
$q(\cdot)$	Variational distribution
$\mathcal{F}$	Variational free energy
ϵ_x	Dynamics prediction error
ϵ_y	Observation prediction error
η	learning rate
Δt	Inference step size
Σ_k	Posterior variance
K	Kalman Gain matrix

Table 1: Table of mathematical notation used in the paper.

Model foundations: HMM and free energy

The conceptual level of our model is grounded within the Bayesian Brain paradigm (Friston, 2005; Friston & Ao, 2012; Knill & Pouget, 2004; Pouget, Beck, Ma, & Latham, 2013). Specifically, we assume that the problem of perception is fundamentally an *inference* problem, where there exists some real world 'out there', from which we only receive noisy and distorted sensory input. The task of perception is to untangle and counteract the noise in order to reconstruct the real (but hidden) state of the world given only our sensory observations. Thus,

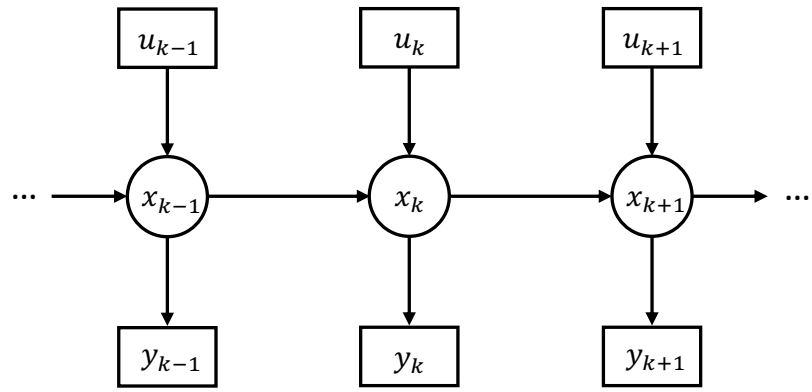

Figure 1: Graphical model of the generative process assumed by the temporal predictive coding networks. x_k correspond to hidden states, y_k to observations, and u_k to control inputs. Circles denote latent variables, squares denote observations, and arrows denote conditional dependence of the variables (the absence of an arrow indicates conditional independence).

mathematically, we can represent the problem of perception as trying to infer a series of latent states of the world x_k $(k = 1, 2, \dots)$ from their corresponding sensory observations y_k $(k = 1, 2, \dots)$. We assume that the underlying graphical structure of our temporal predictive coding network is an HMM, where the hidden states x_k follow a Markov chain. That is, the current hidden state of the world only depends upon the previous hidden state. Also, the current observation is generated by the current state of the world only, with no dependence on past noisy observations. Figure 1 shows the generative process of the temporal predictive coding where, for generality, we have also added 'control' inputs u which can be thought of as known inputs to the system at every time step (such inputs are included in the Kalman filter that we will later use as a benchmark). This control input is useful to model systems where there are known external forces acting on the system (such as an agent's own actions) which we don't necessarily want to model simply as part of the environmental dynamics.

From the generative model in Figure 1, we can also write out specific equations for the dynamics of the states and observations in what is called the state-space

representation:

$$x_k = Af(x_{k-1}) + Bu_k + \omega_x \tag{1}$$

$$y_k = Cf(x_k) + \omega_y \tag{2}$$

where A is the matrix that transitions the previous hidden state x_{k-1} to x_k, B is the matrix governing how the control input u_k affects the current hidden state, and C is the 'emission' matrix that determines how the observation y_k is generated from the hidden state. f is a function transforming x_k that may be nonlinear. The above state-space representation also includes sources of noise ω_x and ω_y. We will assume a white Gaussian noise model such that $\omega_x \sim \mathcal{N}(0, \Sigma_x)$ and $\omega_y \sim \mathcal{N}(0, \Sigma_y)$ are zero-mean Gaussian random variables with covariance matrices Σ_x, Σ_y [1]. Therefore, x_k and y_k can be considered as random variables that follow the Gaussian distributions:

$$x_k | x_{k-1}, u_k \sim \mathcal{N}(Af(x_{k-1}) + Bu_k, \Sigma_x) \tag{3}$$

$$y_k | x_k \sim \mathcal{N}(Cf(x_k), \Sigma_y) \tag{4}$$

The objective of our model is to obtain an estimate $\hat{x}_k$ of the current x_k, given the previously estimated $\hat{x}_{k-1}$, current noisy observation y_k and control input u_k. This objective can be expressed as estimating the posterior distribution $p(x_k | y_k, \hat{x}_{k-1}, u_k)$, which takes an analogous form of the objective used in static predictive coding and more generally, free energy principle (Bogacz, 2017; Friston, 2005). If function f is non-linear, directly estimating this posterior is intractable, and thus variational inference (Beal et al., 2003; Ghahramani, Beal, et al., 2000; Neal & Hinton, 1998) is utilized to find an approximate of the posterior. Specifically, we assume that there exists an approximate posterior $q(x_k)$ and we seek an approximate posterior that is as close as possible to the 'true' posterior $p(x_k | y_k, \hat{x}_{k-1}, u_k)$, and variational

[1] In the control theory literature, the process noise Σ_x is often denoted as Q and the observation noise Σ_y as R. We instead follow the convention that has arisen within the predictive coding literature, which we also believe is more straightforward, by calling them Σ_x and Σ_y, since it makes explicit that these variances are simply the variances of the two distributions composing the generative model.

inference finds such a $q(x_k)$ by minimizing an upper bound on the divergence between the true and approximate posteriors called the variational free energy. If we make an additional simplifying assumption that $q(x_k)$ follows a Gaussian distribution with its density highly concentrated at its mean, the free energy $\mathcal{F}_k$ becomes approximately equal to (Bogacz, 2017; Buckley et al., 2017):

$$\begin{aligned} \mathcal{F}_k &= -\ln p(y_k, x_k | \hat{x}_{k-1}, u_k) \\ &= -\ln p(y_k | x_k) p(x_k | \hat{x}_{k-1}, u_k) \end{aligned} \tag{5}$$

and it is also sufficient to estimate the mode of the approximate posterior (which is the same as its mean) instead of estimating the whole distribution with this assumption. Furthermore, with the Gaussian assumptions in Equations 3 and 4, the free energy becomes:

$$\begin{aligned} \mathcal{F}_k = &(y_k - Cf(x_k))^T \Sigma_y^{-1} (y_k - Cf(x_k)) \\ &+ (x_k - Af(\hat{x}_{k-1}) - Bu_k)^T \Sigma_x^{-1} (x_k - Af(\hat{x}_{k-1}) - Bu_k) \end{aligned} \tag{6}$$

Importantly, we can express this objective as the sum of squared *prediction errors* weighted by their inverse covariances (which are often called precisions in the predictive coding literature). In this model, there are two kinds of prediction errors – 'sensory' prediction errors which are the difference between the observation and predicted observation $y_k - Cf(x_k)$ and 'temporal' prediction errors which are the difference between the inferred current state and the current state predicted from the previous state $x_k - Af(\hat{x}_{k-1}) - Bu_k$. Thus, by finding an estimate $\hat{x}_k = \text{argmin}_{x_k} \mathcal{F}_k$, we effectively minimize the squared sum of these prediction errors while the precision matrices serve to weight the importance of the sensory and temporal prediction errors in accordance with their intrinsic variance (so highly variable prediction errors are weighted less). After the minimization finishes, the estimated $\hat{x}_k$ can then be used to estimate the hidden state at the next step $k + 1$.

Algorithm in the model

With the objective function in Equation 6, it is then possible to derive an iterative algorithm to perform its minimization via gradient descent. Similar to static predictive coding (Rao & Ballard, 1999), the gradient descent is performed on two

sets of values: the hidden states x_k and the weight parameter matrices A, B and C. As we will show in the next subsection, the former can be implemented as neural responses and the latter can be implemented as synaptic connections in a neural circuit in a similar way to static predictive coding. For the hidden state x_k, its dynamics follow:

$$\frac{dx_k}{dt} = -\frac{\partial \mathcal{F}_k}{\partial x_k} \tag{7}$$

At convergence, we can say that the equilibrium value $\hat{x}_k$ represents the optimal inference about the mean of the true hidden state of the world. It is worth mentioning that we now introduced two different indices k and t for distinct time scales: for discrete steps k at which the observations arise and continuous real time t in which computations are made within the model, and we will discuss the relationship between them in detail in the next subsection.

To derive the exact expression for the inferential dynamics, we can define the precision-weighted state and observation prediction errors as ϵ_x and ϵ_y respectively as follows:

$$\epsilon_y = \Sigma_y^{-1}(y_k - Cf(x_k)) \tag{8}$$

$$\epsilon_x = \Sigma_x^{-1}(x_k - Af(\hat{x}_{k-1}) - Bu_k) \tag{9}$$

We can then write Equation 7 as:

$$\frac{dx_k}{dt} = -\frac{\partial \mathcal{F}_k}{\partial x_k} = -\epsilon_x + f'(x_k) \odot C^T \epsilon_y \tag{10}$$

where $\odot$ denotes the element-wise product between vectors. The dynamics have contributions from both the sensory and the temporal prediction errors, and the contribution of the sensory prediction error is 'mapped backwards' through the transpose matrix C^T.

Similarly, if we assume that the A, B and C parameter matrices are learnable, we

can derive update rules also following gradient descent on $\mathcal{F}_k$:

$$\begin{aligned}
\Delta A &= -\eta \frac{\partial \mathcal{F}_k}{\partial A} = \eta \epsilon_x f(\hat{x}_{k-1})^T \\
\Delta B &= -\eta \frac{\partial \mathcal{F}_k}{\partial B} = \eta \epsilon_x u_k^T \\
\Delta C &= -\eta \frac{\partial \mathcal{F}_k}{\partial C} = \eta \epsilon_y f(x_k)^T
\end{aligned} \tag{11}$$

In the above equations, η denotes a scalar learning rate. As we will see below, the iterative updates will correspond to local Hebbian plasticity in the neural implementation of the model.

Typically, if the A, B, and C matrices are learned, then the matrices are updated for a single step according to Equations 11 after the x_k has already converged and using the equilibrium values $\hat{x}_k$. This is because it is often assumed that these variables represent more slowly changing variables in the real world. Moreover, in the neural implementation, these matrices are often assumed to be implemented by synaptic strengths which typically change slowly while the $\hat{x}$ variables are typically mapped to neural firing rates, which change quickly.

The algorithm for learning a temporal predictive coding model is shown in Algorithm 1. The algorithm assumes that sensory observations y_k arrive at discrete steps k. At each step k, we first initialize the x_k values with the previous estimated $\hat{x}_{k-1}$ value from the last step. Then, we iterate Equation 10 until convergence for a given observation y_k. Upon convergence our $\hat{x}_k$ becomes our best estimate of the true state of the world. Given this $\hat{x}_k$ we can update the A, B, and C matrices using Equations 11.

Neural circuit implementation

The update rules and dynamics we have derived in Equations 10 and 11 can be mapped to a recurrent predictive coding network architecture with biologically plausible Hebbian learning rules. In this section we present two examples of networks implementing the algorithm exactly, and then a simplified network that approximates the algorithm.

Algorithm 1: Single training epoch for temporal predictive coding

N: Discrete steps of observations

for $k = 1$ **to** N **do**

  Initialize x_k with previously inferred $\hat{x}_{k-1}$

  while x_k *not converged* **do**

    Perform inference $\frac{dx_k}{dt}$ (Equation 10)

  end

  Update weight matrices ΔA, ΔB, ΔC (Equation 11) using inferred $\hat{x}_k$

end

Figure 2A presents a potential example of how the update rules we have derived can be implemented in a neural network with only local and Hebbian plasticity, which is similar to the standard predictive coding network (Rao & Ballard, 1999). In this network observations y_k enter at the lowest level, and cause sensory prediction errors ϵ_y as the observations are met by top-down predictions. These prediction errors are explicitly represented by the firing rates of 'prediction error neurons'. These prediction error neurons receive top-down inhibitory connections from 'value neurons' in the layer above which, through their firing rates, represent the inferred posterior values $\hat{x}_k$. Similarly, at the layer above there are additional prediction error neurons that represent the difference between the current activity and that predicted based on the previous inference mapped through the dynamics function (Equation 9), and we assume that there are dedicated neurons that 'memorize' latent activities $\hat{x}_{k-1}$ inferred at the previous discrete time step, which is used to make a prediction of the current latent activities and reloaded at the end of inference at each time step.

There are several important aspects to note about this model. First, all the required update rules can be implemented using purely local information. The dynamics of the hidden state estimates x_k (Equation 10) can be reproduced locally since the value neurons receive inhibitory connections from the prediction error neurons ϵ_x at their 'layer' as well as excitatory connections from the prediction error neurons ϵ_y at the layer below. Similarly, the prediction errors ϵ_y can be computed according to Equation 8 as the corresponding prediction error neurons receive excitatory input

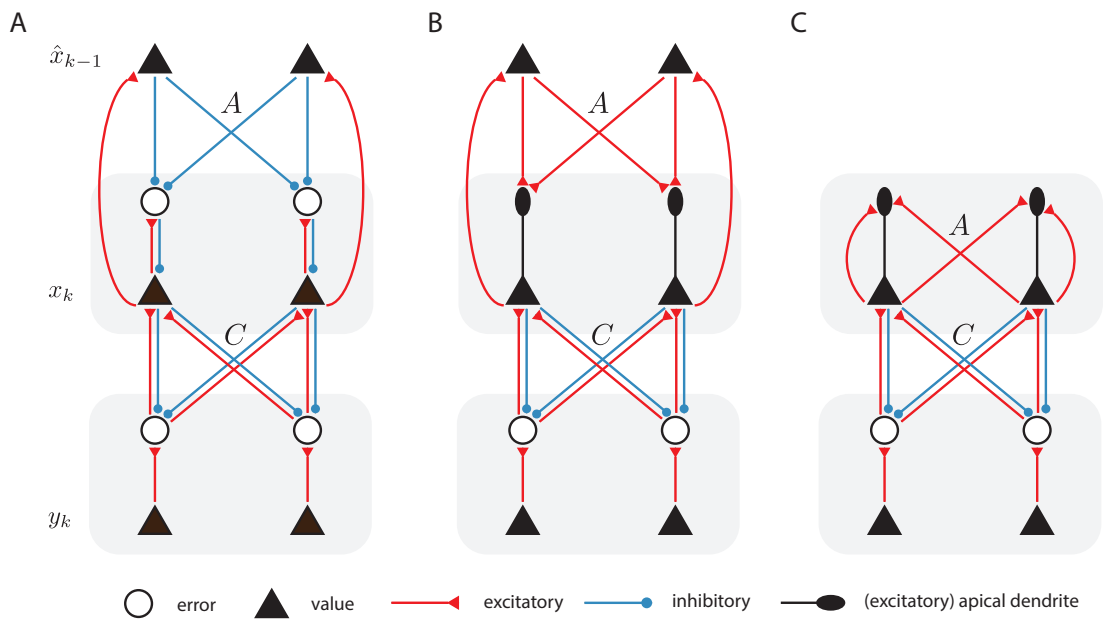

Figure 2: Possible neural implementations of temporal predictive coding. **A**: Potential neural circuit implementing the iterative recurrent predictive coding algorithm. For simplicity, we have depicted each neural 'layer' as possessing only two neurons. **B**: Version of the model where the prediction errors are represented by the difference in membrane potential in soma and at apical dendrites (depicted as rectangles). **C**: Neural circuitry required to implement the single-iteration predictive coding algorithms. This model no longer includes a separate set of neurons explicitly storing the estimate of the previous timestep, but instead the temporal prediction errors are computed naturally through recurrent connections. For simplicity we omitted the control inputs Bu_k, which can be implemented in a similar way to the recurrent inputs $A\hat{x}_{k-1}$ to the error neurons or apical dendrites.

y_k and inhibition from neurons encoding x_k. Scaling by inverse covariance matrix could be achieved through additional inhibitory connections between prediction error neurons (not shown for simplicity) as described by Bogacz (2017), and it was recently demonstrated that the inverse of covariance matrix can also be reparameterised as a recurrent weight matrix to circumvent the implausibility of the inversion (Tang et al., 2023). The prediction error ϵ_x can be computed analogously. The update rules for the A, B, and C weight matrices (Equations 11) are also

precisely Hebbian, since they are outer products between the prediction errors and the value neurons of the layer above which, crucially, are also precisely the pre and post-synaptic activities of the neurons where the synapses implementing these weight matrices are located.

The network shown in Figure 2A follows a standard predictive coding architecture, but it could be simplified because the prediction error neurons encoding ϵ_x only project to corresponding neurons encoding x, and we could thus borrow the idea of dendritic computing similar to the model of Sacramento, Costa, Bengio, and Senn (2018). In particular, substituting Equation 9 into Equation 10, we obtain:

$$\frac{dx_k}{dt} = -x_k + Af(\hat{x}_{k-1}) + Bu_k + f'(x_k) \odot C^T \epsilon_y \tag{12}$$

where, for clarity of explanation, we omitted the precision Σ_x^{-1} that can be implemented in ways mentioned above. By writing the dynamical equation in this way, we assume that there is no building block within the model that encodes the error explicitly; rather, the apical dendrite will encode the inputs $Af(\hat{x}_{k-1}) + Bu_k$ and send the signal to the soma of the pyramidal neuron. This dendritic signal excites the soma and drives the inferential dynamics (Equation 12), together with the decay $-x_k$ that is intrinsic to the soma and feedback signals from the observation layer. The corresponding neural implementation is shown in Figure 2B. Although the architecture of the network becomes simpler, learning parameters A and B is less straightforward because the prediction error ϵ_x is not explicitly represented in activity of any neurons in the network. Nevertheless, it is possible to construct models in which ϵ_x is represented in internal signals (e.g. concentrations of particular ions or proteins) within neurons encoding x: Notice that the error can be represented as the difference between the neural activity and the membrane potential in the apical dendrite. Since both of these quantities are encoded within the same neuron, it is plausible that this computation could be performed, where the error is implicitly encoded in an internal signal. Such a signal could then drive local synaptic plasticity to learn the A and B matrices.

While simulating the model, we update state estimated by numerically solving Equation 12 using the Euler method, i.e. we calculate the state estimates for every

interval Δt:

$$x_k(t+\Delta t) = x_k(t) + \Delta t \left[-x_k(t) + Af(\hat{x}_{k-1}) + Bu_k + f'(x_k(t)) \odot C^T \epsilon_y\right] \quad (13)$$

The above expression highlights that the algorithm has two nested timescales – firstly there is the 'external' timescale which is where sensory inputs y_k are received in a sequence of steps we index by subscript k. Then, for each external step, there is an internal inference of the hidden state that is numerically implemented as a set of recurrent iterations within that external step, which we denote as t. In such a nested framework, the implementations in Figure 2A and B suffer from the need to store and hold fixed the state estimate of the previous step while the iterative inference of the state estimate for the current step is ongoing. Specifically, the estimate from the previous step $\hat{x}_{k-1}$ needs to be held fixed throughout the iterative procedure while the actual current values $x_k(t)$ vary in order to find a balance $\hat{x}_k$ between the demands of matching the prediction from the last step and also the current observation (Algorithm 1). Once the iterations are complete for a step, the new value of $\hat{x}_k$ needs to be loaded into the memory and stored as the last step for the next set of iterations. In situations where the observation are separated in time, it is known that neurons are able to store the representation of stimuli presented a few seconds earlier in their activity (Rainer, Asaad, & Miller, 1998). In the case where sensory input arrives continuously, the two time-scales coincide, and there may be no time for inference between steps of sensory input. In that latter case, the algorithm can be adapted to remove the issue of nested timescales without unduly harming filtering performance by simply using a *single* iteration of the internal inference for each external step. This means that there is effectively no 'inner loop' of the algorithm any more, since the inner loop consists of just a single iteration. This makes the algorithm fully online in the sense that it receives a new sensory input for every step. In particular, note that by equating time indices $t = k\Delta t$ in Equation 13, we obtain:

$$\hat{x}_k = x_k + \Delta t \left[-x_k + Af(\hat{x}_{k-1}) + Bu_k + f'(x_k(t)) \odot C^T \epsilon_y\right] \quad (14)$$

A diagram of a potential neural circuit implementing this single-step algorithm is presented in Figure 2C. This network no longer includes neurons storing past

inferences. Instead, the temporal prediction errors are computed solely using recurrent connections labelled A, which are now assumed to introduce a temporal delay of one step.

The advantage of this approach is that it eschews the challenge of storing and loading the memory of the last step; instead, this memory can be dynamically maintained across a single external step simply through recurrent connections via their intrinsic synaptic delays. The disadvantage of this approach is that the update rules of the algorithm were derived as gradient descent on an objective function, and this approach is equivalent to taking only a single gradient step for each example. Clearly, in many cases, such an algorithm simply would not work because a single step is nowhere near enough to approach the optimum. However, there are two features of the problem that ameliorate much of this difficulty in practice. The first is that, when f is a linear function, the objective is actually convex, and thus the loss landscape is extremely well-behaved. This allows for the use of relatively high integration steps to move large distances in a single, or a few steps, without fear of overshooting the optimum or running into divergences. The other factor is due to the nature of the external world: typically, visual scenes change relatively slowly on a microsecond-by-microsecond level, and thus the optimum estimated hidden state for a single step is likely extremely close to the optimum hidden state for the next. In this case, since we initialize the inference of each step with the optimum of the last step, this will usually be extremely close to the optimum for the current step as well, thus meaning that the algorithm simply does not have to make many iterations to achieve the optimum since it already starts close by. In the next section, we show that in practice on standard tracking and filtering tasks, these two factors can often simplify the inference problem enough that this single iteration approach often works successfully, although it usually does not perform quite as well as multi-step methods.

Results

The results of this paper are partitioned into a theoretical section and experimental sections. In the theoretical section, we examine the relationship between the

temporal predictive coding model and Kalman filtering and demonstrate that the temporal predictive coding network, under certain assumptions, is equivalent to a Kalman filter with a fixed posterior variance. In simulations, we demonstrate that despite not correctly representing the posterior variance, the temporal predictive coding network nevertheless exhibits strong and robust tracking performance on both linear and nonlinear filtering tasks while also being capable of online system identification through the learning of the A and C matrices, unlike the Kalman filter. All code required for replicating the simulation presented in this paper can be found freely online at https://github.com/C16Mftang/temporal-predictive-coding.

Relationship to Kalman filtering

Here we show that both Kalman filtering (Kalman, 1960) and temporal predictive coding can be derived as special cases of the *Bayesian filtering* problem. Bayesian filtering concerns the problem of inferring the *sequence* of hidden 'causes' $x_1, ..., x_K$ of the observations $y_1, ..., y_K$. This problem can be effectively factorised into a sequence of online inference problems i.e., inferring the hidden state x_k at time step k, given the whole history of observations $y_1, ..., y_k$ (Jazwinski, 2007; Stengel, 1986). For simplicity of notation, we denote $x_{1:k} = x_1, ..., x_k$ and $y_{1:k} = y_1, ..., y_k$. The Bayesian filtering problem can thus be formulated as inferring the following posterior distribution:

$$p(x_k|y_{1:k}) \tag{15}$$

We show in Appendix A that this posterior distribution is proportional to:

$$p(x_k|y_{1:k}) \propto p(y_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{16}$$

where the integral is effectively the marginal distribution $p(x_k|y_{1:k-1})$ of the joint $p(x_k, x_{k-1}|y_{1:k-1})$ and can be considered as the prior on x_k. Notice that the term $p(x_{k-1}|y_{1:k-1})$ is exactly the posterior inferred from the previous time step $k-1$, making Bayesian filtering a recursive method (Chen et al., 2003). As a special case of Bayesian filtering, Kalman filtering assumes that the conditional distributions $p(y_k|x_k)$ and $p(x_k|x_{k-1})$ can be parameterized *linearly* as follows:

$$y_k|x_k \sim \mathcal{N}(Cx_k, \Sigma_y); \quad x_k|x_{k-1} \sim \mathcal{N}(Ax_{k-1} + Bu_k, \Sigma_x) \tag{17}$$

Further, it assumes that the posterior estimated at the previous step $k-1$ follows:

$$x_{k-1}|y_{1:k-1} \sim \mathcal{N}(\hat{x}_{k-1}, \Sigma_{k-1}) \tag{18}$$

where $\hat{x}_{k-1}$ is the MAP estimate from the previous step $k-1$ and is the mode (or mean) of the Gaussian posterior with covariance Σ_{k-1} at step $k-1$. Under the above Gaussian assumptions, the prior $p(x_k|y_{1:k-1})$ on x_k (i.e., the integral in Equation 16) can be written as (Bishop & Nasrabadi, 2006):

$$p(x_k|y_{1:k-1}) = \mathcal{N}(A\hat{x}_{k-1} + Bu_k, A\Sigma_{k-1}A^T + \Sigma_x) \tag{19}$$

Kalman filtering then performs maximum a posteriori (MAP) to find $\hat{x}_k$:

$$\begin{aligned}\hat{x}_k &= \underset{x_k}{\operatorname{argmax}} \log p(x_k|y_{1:k}) \\ &= \underset{x_k}{\operatorname{argmin}} (y_k - Cx_k)^T \Sigma_y^{-1} (y_k - Cx_k) \\ &+ (x_k - A\hat{x}_{k-1} - Bu_k)^T (A\Sigma_{k-1}A^T + \Sigma_x)^{-1} (x_k - A\hat{x}_{k-1} - Bu_k)\end{aligned} \tag{20}$$

Since all the transformation functions in this optimization problem are linear, an analytical expression for $\hat{x}_k$ can be derived, which will result in the well-known algorithm for Kalman filtering i.e., the 'projection':

$$\begin{aligned}\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\ \Sigma_k^- &= A\Sigma_{k-1}A^T + \Sigma_x\end{aligned} \tag{21}$$

and the 'correction' step where we then incorporate the new information we have received from the environment to correct our estimates:

$$\begin{aligned}\hat{x}_k &= \hat{x}_k^- + K(y_k - C\hat{x}_k^-) \\ \Sigma_k &= (I - KC)\Sigma_k^- \\ K &= \Sigma_k^- C^T \left[C\Sigma_k^- C^T + \Sigma_y\right]^{-1}\end{aligned} \tag{22}$$

where K is known as the Kalman Gain matrix and is central to the Kalman Filter update rules for the estimated mean and variance in the correction step (Welch, Bishop, et al., 1995). $\hat{x}_k$ and Σ_k are then our estimated mean and covariance of the posterior Gaussian distribution. The derivation of the projection and correction

rules can be found in prior works (Chen et al., 2003; Millidge, Tschantz, Seth, & Buckley, 2021), while we also provide the derivation in Appendix B as to compare to the update rules of our temporal predictive coding model, which is demonstrated below.

Our temporal predictive coding model also aims to solve the Bayesian filtering problem in Equation 16, and it can also make the linear and Gaussian assumptions in Equation 17. Notice that Equation 17 is identical to Equations 3 and 4, but with a linear f. Temporal predictive coding differs from Kalman filtering by making a different assumption on the distribution on the previous-step posterior $p(x_{k-1}|y_{1:k-1})$. Instead of assuming it as a Gaussian distribution in Equation 18, it assumes:

$$x_{k-1}|y_{1:k-1} \sim \delta(x_{k-1} - \hat{x}_{k-1}) \tag{23}$$

where $\delta(x_{k-1} - \hat{x}_{k-1})$ denotes a Dirac distribution with its density concentrated at $\hat{x}_{k-1}$. The prior on x_k for predictive coding thus becomes:

$$p(x_k|y_{1:k-1}) = p(x_k|\hat{x}_{k-1}) = \mathcal{N}(A\hat{x}_{k-1} + Bu_k, \Sigma_x) \tag{24}$$

since the density of $p(x_{k-1}|y_{1:k-1})$ is concentrated at $\hat{x}_{k-1}$. The MAP estimation of $\hat{x}_k$ performed by predictive coding is thus:

$$\begin{aligned}
\hat{x}_k &= \underset{x_k}{\operatorname{argmax}} \log p(x_k|y_{1:k}) \\
&= \underset{x_k}{\operatorname{argmin}} (y_k - Cx_k)^T \Sigma_y^{-1} (y_k - Cx_k) \\
&+ (x_k - A\hat{x}_{k-1} - Bu_k)^T \Sigma_x^{-1} (x_k - A\hat{x}_{k-1} - Bu_k)
\end{aligned} \tag{25}$$

which is the free energy in Equation 5 with a linear f. The above derivation thus provides another interpretation of the temporal predictive coding model i.e., a special case of the Bayesian filtering problem that assumes at each step the variance estimated at the previous step is 0. Again, as all transformations in the optimization objective are linear, we can derive an analytical expression for $\hat{x}_k$:

$$\begin{aligned}
\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \\
\hat{x}_k &= \hat{x}_k^- + K(y_k - C\hat{x}_k^-) \\
K &= \Sigma_x C^T \left[C\Sigma_x C^T + \Sigma_y\right]^{-1}
\end{aligned} \tag{26}$$

which is similar to the projection and correction steps for Kalman filtering (Equations 21 and 22), but with Σ_{k-1} assumed to be 0 i.e., temporal predictive coding does not propagate the uncertainty estimations. The derivation of these equations can be found in Appendix B.

It is also worth mentioning that in the temporal predictive coding model, although we did not specify the estimation of uncertainty Σ_k at each step, not *propagating* the uncertainty Σ_{k-1} from the previous step is not equivalent to not *estimating* it. In fact, as we show in Appendix C, even when we choose to estimate Σ_k in the temporal predictive coding model, it is still not propagated. Therefore, our MAP estimation $\hat{x}_k$ will not be affected.

To summarize, by assuming a linear f, here we have shown that both Kalman filtering and temporal predictive coding are special cases of the Bayesian filtering problem, while the key difference is that predictive coding ignores the uncertainty estimated at the previous step when estimating the most likely hidden 'cause' at the current time step, whereas Kalman filtering always estimates this uncertainty. However, as we will show in the experimental results section, in the benchmark tracking tasks, predictive coding performs on par with Kalman filtering, albeit not estimating the posterior uncertainty. Importantly, there are several advantages of predictive coding over Kalman filtering as a model of dynamical processing in the brain. Firstly, the projection and correction steps of Kalman filtering require complicated matrix algebra and are challenging to compute in neural circuitry, especially the Kalman Gain matrix K. On the other hand, although we can derive analytical results for the estimates of predictive coding as well, these estimates can be obtained via the iterative methods mentioned above, which afford plausible circuitry implementations (Figure 2). Secondly, the iterative nature of our predictive coding model also makes it adaptable to nonlinear f, where there are no analytical solutions to the Bayesian filtering problem. In contrast, extending the Kalman Filter to nonlinear systems is challenging and standard methods such as the extended Kalman filter (Ruck, Rogers, Kabrisky, Maybeck, & Oxley, 1992) work by linearizing around the nonlinearity and thus require knowledge of the Jacobian of the nonlinearity at every state, which is also challenging to implement in neural circuits. Finally, the Kalman filter assumes knowledge of the correct A, B,

and C matrices while these must presumably be learned from sensory observations in the brain.

Simulations of the models

As well as these theoretical results, we empirically quantify how the predictive coding network performs on Bayesian filtering tasks in both the linear and nonlinear setting. This allows us to directly compare our algorithm to the Kalman filter to assess how important representing the posterior variance is, as well as to investigate the success of the online learning of the A and C matrices. Since other recent studies of related temporal predictive coding models have simulated them on tasks used in machine learning (Ororbia et al., 2020) and visual neuroscience (Jiang & Rao, 2022), in this paper we focus on simulating tasks following the setup assumed by the Kalman filter.

Linear model

Here we first present results for the linear model on a simple tracking task of the kind to which the Kalman filter is commonly applied in industry, and which also shares a close relationships with those the brain must solve – such as tracking a moving object with visual saccades during smooth pursuit. Here, the goal is simply to infer the unknown hidden state (position, velocity, acceleration) of an object which is undergoing an unknown acceleration. We receive noisy observations of the position, state, and acceleration of the object which are mapped through a random C matrix with additional observation noise. We use a random C matrix for the observation mapping to simulate and test the most difficult scenario where the observations are entirely scrambled.

Mathematically, the generative process of this task can be represented according to a linear state space model. We assume that the position, velocity, and acceleration of the object follow the usual laws of Newtonian physics, while the control input

only affects the acceleration, giving us the following A and B matrices,

$$A = \begin{bmatrix} 1 & \Delta k & \frac{1}{2}\Delta k^2 \\ 0 & 1 & \Delta k \\ 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{27}$$

In Equation 27, Δk denotes the duration of the interval between successive sensory observations (we used $\Delta k = 0.001$). Additionally, we draw a C matrix from a random Gaussian distribution $C \sim \mathcal{N}(0, 1)$. The process and observation noise Σ_x and Σ_y were set to identity matrices. We then generate the true latent states x_k and the noisy observations y_k using Equations 1 and 2, initialized with a zero vector when $k = 0$. The performance of the models is then measured as the mean squared error (MSE) between the estimated $\hat{x}_k$ and true x_k across all observation time steps. Figure 3A shows an example of the true system state that we generated across 1000 time steps, and Figure 3B shows its corresponding noisy observations. As can be seen, the projected observations are completely scrambled by matrix C, making it a challenging task for the models to retrieve the true system states.

We then investigate tracking performance using the predictive coding model compared to the Kalman filter for both 5 steps of inference between observations ($\Delta k = 5\Delta t$) and 1 step ($\Delta k = \Delta t$). Since the problem is linear, we also investigate the performance of a predictive coding model when its inference dynamics have reached the equilibrium, using the equilibrium condition derived in Equation 26. Since tracking performance is extremely good when zoomed out over 1000 timesteps, in Figure 3C, we plot the estimates of acceleration (x_3) on the 40 timesteps between 560 and 600 steps in. It can be seen from Figure 3C that both the predictive coding model with 5 inference steps and that with fully converged inferential dynamics could achieve comparable performance to Kalman filter. Interestingly, the estimates of predictive coding tend to be closer to those of the Kalman filtering, rather than the true values. Although the predictive coding model with a single inference step (corresponding to the neural implementation in Figure 2C) has worse tracking performance, it is able to estimate a smoothed version of the trajectory of the system state. We hypothesize that the smoothed estimate with a single inference step is likely due to the fact that the predictive coding algorithm does

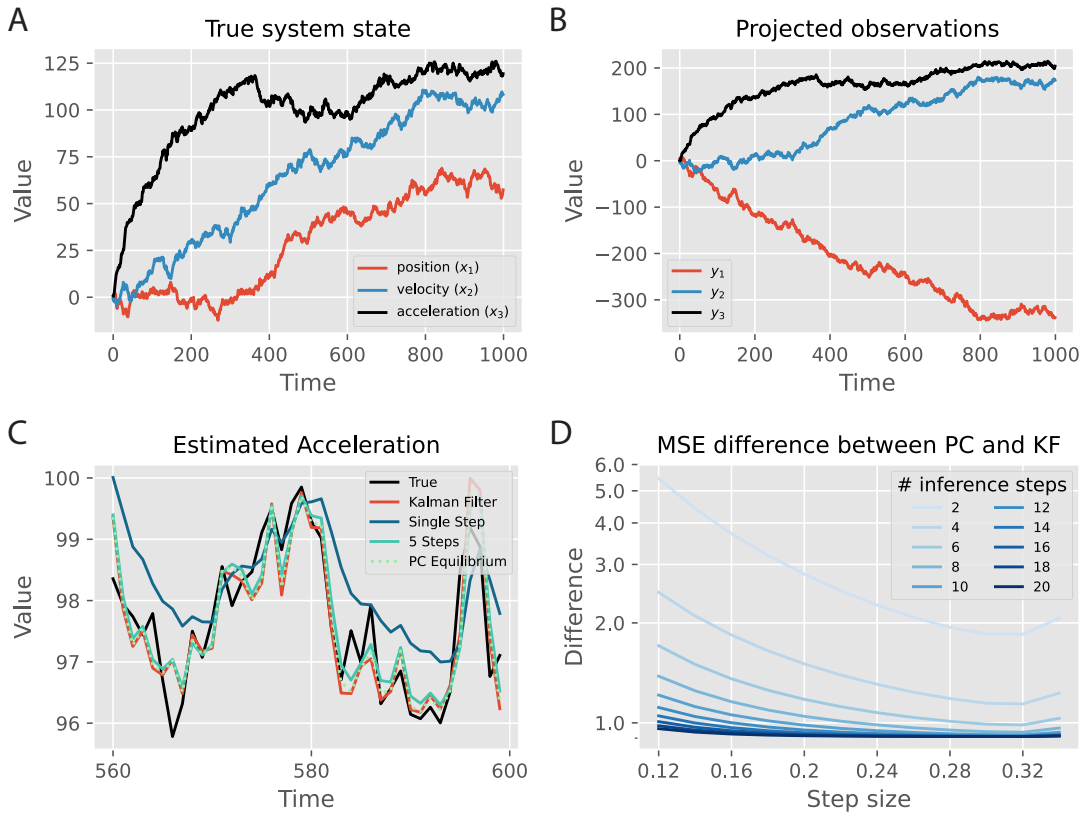

Figure 3: The tracking task and the impact of inference step size and the number of inference steps on performance. **A**. The dynamics of the true hidden state represented as a 3-dimensional vector at each time step, with entries corresponding to position (x_1), velocity (x_2) and acceleration (x_3). **B**. The projected noisy observations from the true system state in A. **C**: Estimates of the acceleration with different models, zoomed in at the interval between 560 and 600 time steps. **D**: MSE difference between predictive coding and Kalman filter, with varying number of inference steps and step sizes for predictive coding. PC stands for predictive coding and KF stands for Kalman filter.

not completely converge in 1 iteration, and so does not completely optimize its estimate on every timestep, with the effect that the estimate is less sensitive to new information and effectively averages over recent experiences rather than optimally solving each one independently. A similar performance comparison is obtained on the position (x_1) and velocity (x_2) and is thus not shown.

To quantify the effect of the number of inference iterations and integration step size Δt upon performance, in Figure 3D we plotted the MSE difference between predictive coding models with various inference iterations and inference step sizes and the Kalman filter, which is the optimal solution to the tracking problem. The MSE is calculated as the mean squared difference between the estimated system state $\hat{x}_k$ and true state x_k, averaged across time steps and trials. We find that with a small number of inference steps, the performance of the predictive coding model is worse, indicating that additional steps of inference aid the tracking performance of the algorithm. Moreover, although increasing the step size will initially improve the performance, the MSE will start to increase if the step size is too large. It can also be seen that with more inference steps and appropriate step sizes, the performance of predictive coding will be able to approximate that of the (optimal) Kalman filter.

Learning

In the previous investigations, we fixed the parameters A, B and C to the true values and only performed the inference dynamics. However, in many cases, simply inferring the hidden states of the world is not enough because we cannot assume that we know a-priori the structure of the dynamics or observation functions of the world. That is, in most real-world situations, the A, B, and C matrices are unknown. Instead, we must *learn* these matrices from observations. In our recurrent predictive coding model, we have a natural Hebbian plasticity-based learning rule which we can use to learn these matrices directly (Equation 11). Here, we investigate how learning these parameters affects the performance of our predictive coding model. Specifically, we investigate three different ways of setting the values for A and C: 1) fixing them to the true values used for generating the data; 2) learning them using Equation 11 and Algorithm 1; 3) fixing them to random values. We then examine the performance of these models on two levels, the latent state level (x) and the observation level (y), by measuring how well the model estimates the activities on both levels. It is worth noting that the observation estimates are calculated by performing a forward pass at each time step i.e., $\hat{y}_k = C(A\hat{x}_{k-1} + Bu_k)$, where the value of C is obtained at each time step

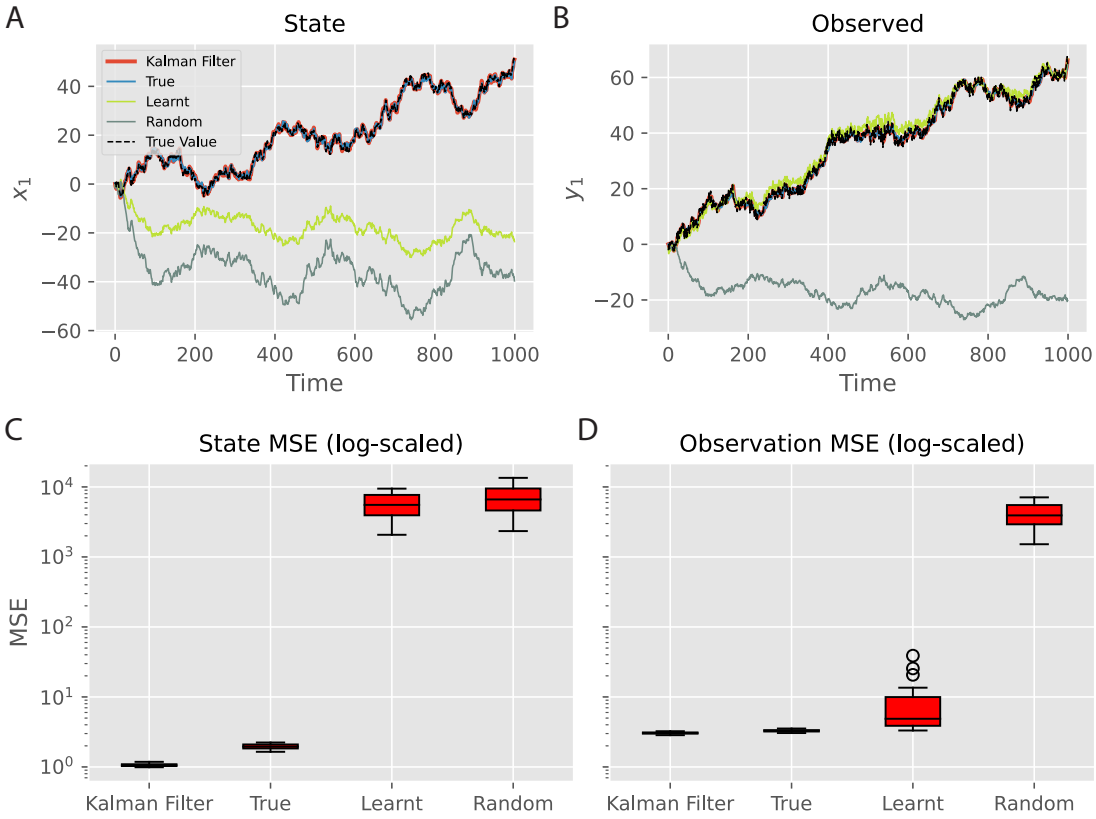

Figure 4: Effects of learning parameters A and C. **A, B**: Estimation of the state and observation trajectories respectively by different models. 'True', 'Learnt' and 'Random' denote the predictive coding model with true, learned and random A and C respectively. Only the first dimension of the latent and observation is shown for simplicity. The other two dimensions have similar performance. **C, D**: MSE of the predictions on the hidden and observation levels respectively. Boxplots obtained with 40 trials on each model.

k via the aforementioned three approaches. The results are shown in Figure 4.

For this set of results, we use 20 inference steps with a step size 0.2 to get the optimal performance for the predictive coding models, based on Figure 3D. Figure 4A shows that, while our predictive coding model estimates the latent state well with true A and C, when asked to learn the parameters, the model fails to accurately estimate the latent state. Quantitatively, as shown in Figure 4C, the estimation MSE of the learning model on the state level is similar as for the predictive coding

model with totally random parameters, which is much higher than those of the Kalman filter and predictive coding model with true parameters. On the other hand, however, we find that the model learning A and C can accurately estimate the observations even with the incorrectly estimated latent state (Figure 4B). This effect arises because the problem of inferring the true hidden state from the data is fundamentally under-determined. There are many possible hidden states that, given a flexible learnt mapping, could result in an identical predicted observation. Importantly, despite inferring a different representation of the hidden state, the network is able to learn A and C that correctly predicts the incoming observations (Figure 4D).

Simple non-linear model

We tested the nonlinear predictive coding model in two different experiments. In this section we focus on a simple simulation in which observations are generated from a probabilistic model of the same form as that of non-linear predictive coding network. In such setting we expected the non-linear temporal predictive coding network to learn predict these data well, and we wished to test if it can do it better than a linear model.

We generated a time-series of 100, two-dimensional stimuli according to Equations 1 and 2 where the nonlinear function f is chosen to be the hyperbolic tangent function, and ω_y and ω_x correspond to standard Gaussian noise sampled from an i.i.d. Gaussian distribution with a mean of 0 and a variance of 0.01. In this 2D example, parameters of C and A were set to a rotation matrix and the identity matrix multiplied by 3, respectively, which are as follows,

$$
\begin{aligned}
A &= \begin{bmatrix} \frac{-1}{2}\Delta k & 1 \\ -1 & \frac{-1}{2}\Delta k \end{bmatrix} \times 3 \\
C &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times 3
\end{aligned}
\tag{28}
$$

The multiplication by 3 in both matrices above was performed to produce a system that operates beyond the linear range of the hyperbolic tangent function (we

used $\Delta k = 0.5$). The observed stimuli are periodic signals which are simply noisy versions of the hidden variables.

Both a liner and non-linear models were trained to predict these stimuli. The models were simulated with single state of inference between samples ($\Delta k = \Delta t$). For the learning process, the parameters of the C matrix were initialized as identity matrix while the weights between hidden units were initialized to zeros. Additionally, each neuron $\hat{x}$ was initialized to 0, and the model was trained according to Equations 11.

Figures 5A,B show zoomed-in time-series with predictions of nonlinear and linear models (for the sake of visualization, only one dimension is shown). Figure 5A demonstrates that in the first 50 time steps of the simulation both models significantly move their predictions towards true time-series. Figure 5B shows the final 50 time steps, where the nonlinear model performed relatively better than the linear model.

The observation prediction error of each model is depicted in Figure 5C. Within the first 50 to 100 time steps, the linear model has a much higher rate of reduction in its prediction error compared with the nonlinear model; however, this rapid drop in the error is followed by a stationary line and even a slow divergence from the optimum toward the end of the simulation. On the other hand, the nonlinear model shows a slower, but relatively steady, rate of error reduction which ultimately leads to convergence. The lower loss achieved by the nonlinear model could be explained by the fact that the ground-truth simulation data has a different oscillatory pattern as a result of the nonlinearity in the Equations 1 and 2 that were used to generate such data. Therefore, the shape of the signal is different than one would expect to observe from a simple linear system. To show this, we generated a simulation data based on the following equation:

$$\hat{y}_k = a \sin(bk + c), \tag{29}$$

where a, b, and c were approximated by a least squares optimisation so they match the ground-truth data. The comparison is depicted in Figure 5D.

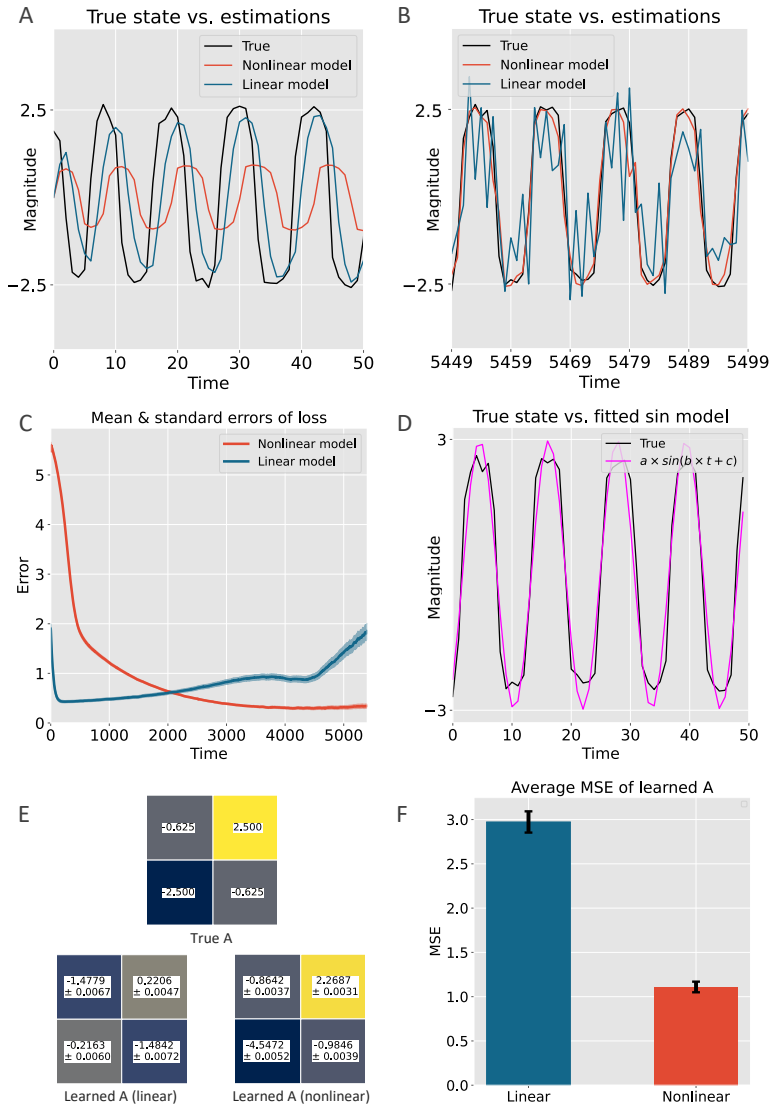

Figure 5: Comparison between nonlinear and linear model predictions for randomly generated time-series. **A**: Time-series and the corresponding predictions of the nonlinear and linear models during the first 50 time steps of learning. **B**: Analogous data and predictions during the last 50 time steps of the simulation. **C**: Observation prediction error during training. The solid lines indicate the average errors from the 100 simulations with the shaded area as standard errors. **D**: Comparison of the shape of the ground-truth signal versus the shape obtained from simulating a parametric *sin* data with coefficients and constants optimised to match the ground-truth. **E**: Comparison of the learned A matrices (mean values) of both models against the true matrix. **F**: Mean error of the learned A parameters (with standard errors).

To better understand the behaviour of each model, we investigated how well the transition matrix A is learnt in each case. In Figure 5E, we plotted the mean learnt matrix by colour-coding the elements and normalising it to the minimum and maximum values of the true matrix. As expected, the parameters of the A matrix are closer to the true values when compared against their corresponding values in the linear model. To quantify it, we also plotted the average errors of the learned parameters for both models, showing that the linear model has a much higher error (almost by three folds) versus the nonlinear model - see Figure 5F.

Pendulum

In order to investigate if temporal predictive coding model can learn to predict time-series for which the generative process does not explicitly follow that of the model, we trained the model on the simulated motion of a pendulum. Figure 6A shows a free-body diagram of the pendulum that we simulated in this experiment, demonstrating the mass, length L, and force vectors acting on the system. We describe the state of the pendulum by the angle of the pendulum θ_1 and its angular velocity θ_2. According Newton's second Law of Motion, the angle of the pendulum evolves according to:

$$\ddot{\theta}_1 = -\frac{g}{L}\sin(\theta_1) \tag{30}$$

where g is gravity. We can express this motion in a set of first-order equations:

$$\dot{\theta}_1 = \theta_2 \tag{31}$$

$$\dot{\theta}_2 = -\frac{g}{L}\sin(\theta_1) \tag{32}$$

In simulation we used the following parameters: $g = 9.81$ m/s^2, $L = 3.0$ m. We then simulated the system as an initial value problem by numerically integrating the equations using the explicit Runge-Kutta method for 2500 seconds with $\Delta k = 0.1$ second time steps and initial values of $\theta_1 = 1.8$ rad and $\theta_2 = 2.2$ rad/s. These values

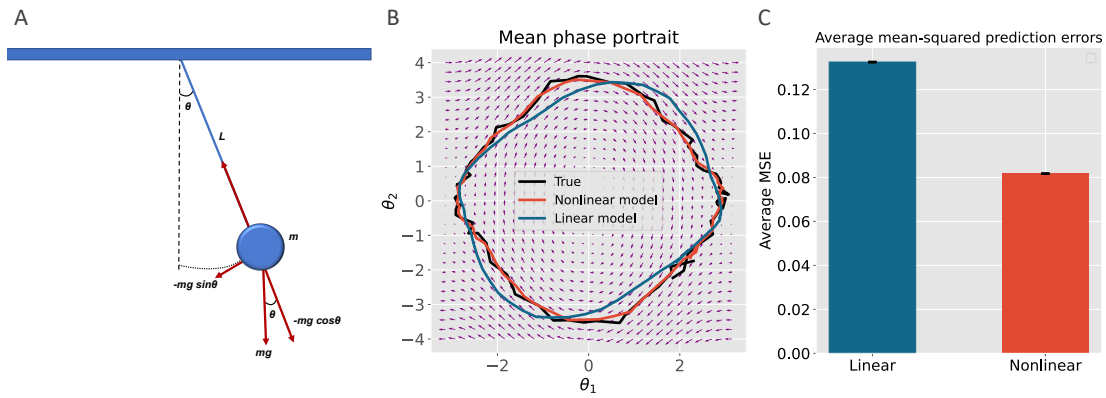

Figure 6: Simulations of the pendulum. **A**: A free-body diagram of a simple pendulum that has a mass m attached to a string with length L. Also shown are the forces applied to the mass. The restoring force $-mg\sin\theta$ is a net force toward the equilibrium position. **B**: A phase portrait of the pendulum simulation showing the result of our linear versus nonlinear models prediction for the ground-truth data. The vector field (i.e. set of small arrows) was created by computing the derivatives of $\frac{d\theta_1}{dt}$ and $\frac{d\theta_2}{dt}$ at $t = 0$ on a grid of 30 points over the range of $-\pi$ to $+\pi$ and -4 to $+4$ for θ_1 and θ_2, respectively. **C**: The barplot shows the difference between the mean prediction errors of the linear model versus the nonlinear model from 100 simulations with varying noise profiles. The mean errors are significantly different ($p << 0.001$).

were chosen to simulate the pendulum motion with a large amplitude of oscillation to shift the system into a more nonlinear regime. The time-series presented to the models y_k were created by adding these numerical solutions and Gaussian noise with zero-mean and standard deviation of 0.1.

We trained non-linear and linear temporal predictive coding models to predict these time-series. Parameters of C and A were initialized as identity matrix and zero, respectively. The learning was performed as in the previous section.

Figure 6B shows the results from the pendulum simulations using the phase portrait of the system. The solutions of the ground-truth simulation and our nonlinear model prediction are plotted on the vector field for the final 80 seconds. Even though both models performed relatively well by correctly predicting the behaviour

of the pendulum motion, the linear model performed worse when the pendulum reached its highest angular displacements (see the noisy prediction around the extremities of the θ_1 axis in Figure 6B). On the other hand, the nonlinear model stayed relatively stable throughout the entire prediction. Figure 6C shows the mean squared error of prediction averaged over 100 separate simulations, and the non-linear model consistently outperformed the linear one.

Discussion

In this paper, we have analysed the recurrent predictive coding architecture for temporal prediction and filtering. This task is important because processing time-varying sequences of inputs and using them to infer dynamically changing hidden states of the world is the core task of the sensory regions of the brain. As such, it is likely that these regions have an architecture heavily specialised for performing such filtering tasks. Here we have shown that the filtering problem can be tackled with a simple and biologically plausible algorithm with a straightforward implementation in neural circuitry.

We have derived our temporal predictive coding model from first principles as a variational filtering algorithm, provided a clear algorithmic derivation in terms of gradient descents on the resulting free energy functionals, and proposed a direct implementation in neural circuitry which relies on only local information transmission as well as purely Hebbian plasticity that inherits the simple neural implementation of static predictive coding networks. We have also demonstrated that in the linear case, the algorithm is closely related to Kalman filtering, and is capable of robustly solving filtering and tracking tasks at a level close to the optimal linear solution. Moreover, and unlike the Kalman filter, we have demonstrated that our model can perform online *learning* of the parameters using Hebbian plasticity, which works rapidly and effectively in predicting the correct observations. We have also extended the algorithm to the nonlinear case, and demonstrated that, when presented with noisy nonlinear stimuli, the nonlinear model has a superior performance over a comparable linear model in both learning the dynamics and predicting the behaviour of the input sensory observation both spatially and

temporally.

Related Work

Several earlier works have tried to approach the problem of Kalman filtering in the brain. Wilson and Finkel (2009) repurpose a line attractor network and show that it recapitulates the dynamics of a Kalman filter in the regime of low prediction error. However their model only works for a single-dimensional stimulus, does not encode uncertainty, and also only works when a linearisation around zero prediction error holds. Deneve, Duhamel, and Pouget (2007) encoded Kalman filter dynamics in a recurrent attractor network. Their approach however encodes stimuli by means of basis functions, which leads to an exponentially growing number of basis functions required to tile the space as the dimensionality of the input grows. In the predictive coding approach, neurons directly encode the mean of the estimated posterior distribution, which means that the network size scales linearly with the number of dimensions. Our gradient method also completely eschews the direct computation of the Kalman gain, which simplifies the required computations significantly. Additionally, Beck, Ma, Latham, and Pouget (2007) show that probabilistic population coding approaches can compute posteriors for the exponential family of distributions of which the Gaussian distribution is a member. However, no explicitly worked-out application of the population coding approach to Kalman filtering exists, to our knowledge. Recent work has also addressed the question of how biological recurrent neural networks can be trained for temporal prediction (Bellec et al., 2020). They proposed that synapses maintain eligibility traces encoding to what extent they contributed to neural activity over time, and when combined with error signals, such traces enable effective credit assignment. It would be interesting to investigate how such eligibility traces could be incorporated into temporal predictive coding networks.

Moreover, other works have also explored predictive coding for temporal predictions. For instance, early works Rao (1997, 1999); Rao and Ballard (1997) utilized a Kalman filter combined with sparse image representations to make future predictions of visual stimuli, but these works did not describe how the Kalman filter can be implemented in biological circuits, and how their Kalman filter-based models

can be extended to the nonlinear case. More recently, Jiang and Rao (2022) trained temporal predictive coding networks on sequences of natural video (filmed by a person walking through a forest) and observed that neurons have spatiotemporal receptive fields resembling those in the primary visual cortex. Temporal predictive coding networks have been extended to include multiple levels of hierarchy (Jiang & Rao, 2022; Ororbia et al., 2020). Analysis of these networks revealed that neurons on higher levels change their activity with a slower time scale than the neurons at the lower levels of the hierarchy (Jiang & Rao, 2022). It has been also demonstrated that hierarchical temporal predictive coding networks can achieve performance comparable to BPTT in standard machine learning benchmarks (Ororbia et al., 2020). However, these models require complex neural network implementations to perform these temporal tasks. It is thus an interesting future direction to see whether our temporal predictive coding model, which inherits the simple neural implementation of the static predictive coding network, can present similar performance and neural responses.

Work by Lotter, Kreiman, and Cox (2016) adapted deep recurrent neural networks to perform a kind of predictive coding whereby the network was trained to predict future *prediction errors* of each layer. They demonstrated that the resulting network was capable of correctly predicting sequences of video frames. While substantially scaling up predictive coding architectures to challenging machine learning tasks, the networks of Lotter et al. (2016) diverged in many ways from classical predictive coding architectures, and also utilized many non-biologically plausible components from machine learning such as convolutional and LSTM layers as well as training their network with BPTT.

Kutschireiter et al. (2017) addressed the question of how temporal predictive coding networks can be extended so they represent posterior uncertainty. They demonstrated that if multiple copies of the network are made, and the dynamics of each network include noise with an appropriate magnitude, then each network can represent a sample from the posterior distribution $p(x_k|y_1 \dots y_k)$ and the collection of networks as a whole can represent the posterior distribution of state in a sampling-based manner. Their model is particularly interesting because the posterior uncertainty can be decoded from the differences in the activity of

individual networks. However, the encoding of posterior uncertainty comes with a cost of a larger number of neurons required to form multiple networks.

Relationship to Kalman filtering

The similarities and differences between the temporal predictive coding algorithms and classical filtering algorithms like Kalman filtering are of significant theoretical interest, as earlier works by Rao and Ballard (1997) have already used Kalman filtering as a model of dynamical processing in the brain. We have found that the crucial distinction is in the representation of the model's uncertainty, mathematically represented in the posterior variance Σ_k. Specifically, in the temporal predictive coding model, although it represents the two 'objective' uncertainties Σ_x and Σ_y of the dynamical system it is inferring, it does not represent the uncertainty in the estimated posterior distribution (unlike the extended model of Kutschireiter et al. (2017)). Crucially, it is the assumption of the prior at each time step that prevents the temporal predictive coding model from successfully propagating the posterior uncertainty through time.

Although temporal predictive coding does not optimally update and represent the dynamically changing posterior uncertainty of the agent, it has other computational advantages over Kalman filtering, which may render it more suitable for implementation in neural circuitry. The key advantage of the predictive coding approach is its computational simplicity compared to the Kalman filtering update rules which require complex matrix algebra and especially matrix inversions to compute the Kalman Gain matrix which are unlikely to be implementable in neural circuitry directly, while the predictive coding equations are simple and only require local and Hebbian updates and can be directly translated into relatively straightforward neural circuits. Moreover, as seen in Figure 3, the predictive coding estimate and the Kalman filter estimates of a given dynamical system end up closely converging anyway, which means that temporal predictive coding networks could provide the brain an efficient and cheap way to approximate the highly effective Kalman filter using only simple circuitry.

One reason why predictive coding networks achieve performance similar to the

Kalman filter is that the posterior uncertainty decays rapidly over trials (for an illustration see Figure 5A in a paper by Moeller, Manohar, and Bogacz (2022)). Furthermore, for deterministic transition processes (with $\Sigma_x = 0$), the posterior uncertainty decays to $\Sigma_k = 0$ (Moeller et al., 2022). Therefore, as the learning progresses, the Kalman filter becomes more similar or even identical (for deterministic processes) to the temporal predictive coding.

Our work thus raises an interesting empirical question as to what kinds of uncertainties various agents – such as humans or animals – actually appear to represent in dynamical inference tasks. For instance, it is not clear that in the literature that subjective confidence ratings are highly correlated with the true dynamical uncertainty of the decisions in a task (Navajas et al., 2017). Although the Kalman filter has been used to describe reinforcement learning (Daw, O'Doherty, Dayan, Seymour, & Dolan, 2006), direct comparison with simpler reinforcement learning models did not favour the Kalman filter (Howard-Jones, Bogacz, Yoo, Leonards, & Demetriou, 2010). It may be interesting, therefore, to compare predictions of the Kalman filter (and the extended model representing posterior uncertainty (Kutschireiter et al., 2017)) and the original temporal predictive coding models to experimental data directly. For example, one could compare if learning rate in reinforcement learning tasks is better described by the Kalman gain or the value from temporal predictive coding.

Relationship with Generalized coordinates

A further interesting theoretical property of the model is its potential to autonomously learn to represent the dynamics of systems in generalized coordinates of motion (Friston, 2008; Friston et al., 2008a) if provided with generalized coordinates as inputs. Generalized coordinates, introduced into the predictive coding literature by Friston et al. (2008a), and well known in engineering practice, involve representing the n'th order derivatives of a state as additional coordinate dimensions. In effect, a point in generalized coordinates of motion to n'th order reflects the approximate trajectory of the state given by the n'th order Taylor expansion around that point. Original predictive coding models involving generalized coordinates typically required hardcoding the relationships between the

coordinates, and the relative precisions between different dynamical orders (Friston, 2008; Friston, Trujillo-Barreto, & Daunizeau, 2008b; Parr & Friston, 2018) which results in intricate and complex hardcoded connectivity, reducing the ultimate biological plausibility of such models. However, our model's capability to directly learn the A and C matrices from data allows the model to simply receive a generalized coordinate state as input and learn the required connectivity online, as demonstrated in our pendulum simulations.

Neural Implementation

There are several interesting questions regarding the biological plausibility of the multi-step neural implementation. Initially these schemes, while they arise directly from the gradient descent derivation, appear biologically implausible for two reasons. The first is the issue of storage. Iterative schemes require the initial conditions (state estimate $\hat{x}_{k-1}$) to be held fixed throughout multiple iterations, and this means that this state must be stored somewhere accessible to be utilized multiple times during the iterative inference phase. It is not clear where or how this information could be stored in the brain, especially in low-level sensory systems. This storage must be local and ubiquitous as a naive implementation of the multi-step algorithms proposed in this paper would require separate storage for every single value neuron.

The second issue relates to the time it takes for an iterative algorithm to converge. Specifically, if we model the brain as receiving visual input as a continuous stream, then multiple iterations based upon a single stimulus would necessitate ignoring the data arriving in the intervals while the iterations are taking place. Moreover, an iterative approach would also take more time to update upon information newly received, which could be crucial for survival in some cases. There are also multiple potential solutions to this problem – firstly, the cortex may implement both an iterative and an amortized feedforward pass solution simultaneously (Tschantz, Millidge, Seth, & Buckley, 2022), and there is substantial evidence for precisely this. Firstly, core visual functions can often occur within 100-200 ms (Carlson, Tovar, Alink, & Kriegeskorte, 2013; Keysers, Xiao, Földiák, & Perrett, 2001; Thorpe, Fize, & Marlot, 1996; Thunell & Thorpe, 2019) which is too short to allow multiple steps

of recurrent optimization, thus demonstrating that some kind of rapid single-step inference is possible. Conversely, there is much evidence that increased viewing time allows for the refinement of representations, reduction of uncertainty, and improvement in accuracy over time, which strongly speaks to the existence of some iterative recurrent processing occurring as well.

Finally, there is some interesting evidence that most brain regions, including the visual cortex, operate on a characteristic frequency (Buzsaki, 2006; Buzsaki & Draguhn, 2004). In the case of the visual cortex, the dominant rhythm is the alpha band at 5-15 Hz. Experiments have found that information presented in phase with these oscillations is processed normally; however, if the information is presented out of phase, then a drop of accuracy ensues, suggesting that the information has not been fully or successfully processed (Ruzzoli, Torralba, Fernández, & Soto-Faraco, 2019). These findings are consistent with the iterative convergence algorithms proposed here being implemented in the cortex.

A further avenue for future work relates to the challenge of learning long-term dependencies which span over many time steps. This has long been a central challenge with these recurrent models, and emerges essentially due to the fact that information is permuted or lost at every step of the recurrent pass, and thus tracking dependencies across many recurrent loops becomes increasingly difficult (Hochreiter, 1998; Hochreiter & Schmidhuber, 1997; Tallec & Ollivier, 2018). Numerous solutions to this have been suggested in the literature, ranging from specially designed cell architectures that can explicitly store or pass along information unaltered (Hochreiter & Schmidhuber, 1997; Tallec & Ollivier, 2018) to having a nested hierarchy of recurrent models which allow for the propagation of information over longer and longer timescales (Koutnik, Greff, Gomez, & Schmidhuber, 2014).

The recurrent temporal predictive coding model we propose can also be implemented when representing prediction errors in dendrites as in Sacramento et al. (2018) and Guerguiev, Lillicrap, and Richards (2017) instead of using explicit prediction error neurons as in Figure 2A. Such an architecture can reconcile predictive coding networks with the lack of strong evidence for there being explicit prediction error neurons in the cortex (Walsh et al., 2020), unlike the dopaminergic reward

prediction error neurons in the mid-brain whose existence has been established for decades (Schultz, 1998; Schultz, Dayan, & Montague, 1997).

Conclusion

In this paper, we have proposed neural implementations of temporal predictive coding model in recurrent networks which utilize only biologically plausible Hebbian learning rules. We demonstrate that the model can successfully perform both linear and nonlinear tracking tasks successfully, along with online system identification of both transition and observation dynamics, and can come close to the optimal tracking performance of the Kalman filter. Moreover, by analyzing closely the mathematical relationship of the model to the Kalman filter, we have identified representation of subjective uncertainty as the key difference between the Kalman filter and the temporal predictive coding networks.

References

Auksztulewicz, R., & Friston, K. (2016). Repetition suppression and its contextual determinants in predictive coding. *cortex*, *80*, 125–140.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., & Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, *76*(4), 695–711. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0896627312009592`

Beal, M. J., et al. (2003). *Variational algorithms for approximate bayesian inference*. university of London London.

Beck, J., Ma, W., Latham, P., & Pouget, A. (2007). Probabilistic population codes and the exponential family of distributions. *Progress in brain research*, *165*, 509–519.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *bioRxiv*, 738385.

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4) (No. 4). Springer.

Bogacz, R. (2017). A tutorial on the free-energy framework for modelling perception and learning. *Journal of mathematical psychology*, *76*, 198–211.

Buckley, C. L., Kim, C. S., McGregor, S., & Seth, A. K. (2017). The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, *81*, 55–79.

Buzsaki, G. (2006). *Rhythms of the brain.* Oxford University Press.

Buzsaki, G., & Draguhn, A. (2004). Neuronal oscillations in cortical networks. *science*, *304*(5679), 1926–1929.

Carlson, T., Tovar, D. A., Alink, A., & Kriegeskorte, N. (2013). Representational dynamics of object vision: the first 1000 ms. *Journal of vision*, *13*(10), 1–1.

Chen, Z., et al. (2003). Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, *182*(1), 1–69.

Clark, A. (2015). *Surfing uncertainty: Prediction, action, and the embodied mind.* Oxford University Press. Retrieved from `https://books.google.co.uk/books?hl=en&lr=&id=TnqECgAAQBAJ&oi=fnd&pg=PP1&dq=andy+clark+surfing+uncertainty&ots=aurm6iDbJR&sig=A5uoJIteAk4JDCEpnQaa2KAbfg4&redir_esc=y#v=onepage&q=andy%20clark%20surfing%20uncertainty&f=false`

Daw, N. D., O'Doherty, J. P., Dayan, P., Seymour, B., & Dolan, R. J. (2006). Cortical substrates for exploratory decisions in humans. *Nature*, *441*(7095), 876–879.

Deneve, S., Duhamel, J.-R., & Pouget, A. (2007). Optimal sensorimotor integration in recurrent cortical networks: a neural implementation of kalman filters. *Journal of Neuroscience*, *27*(21), 5744–5756.

Feldman, H., & Friston, K. (2010). Attention, uncertainty, and free-energy. *Frontiers in human neuroscience*, *4*, 215. Retrieved from `https://www.frontiersin.org/articles/10.3389/fnhum.2010.00215/full`

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

Friston, K. (2005). A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, *360*(1456), 815–836.

Friston, K. (2008). Hierarchical models in the brain. *PLoS computational biology*, *4*(11).

Friston, K., & Ao, P. (2012). Free energy, value, and attractors. *Computational and mathematical methods in medicine*, *2012*. Retrieved from `https://www.hindawi.com/journals/cmmm/2012/937860/`

Friston, K., Stephan, K., Li, B., & Daunizeau, J. (2010). Generalised filtering. *Mathematical Problems in Engineering*, *2010*. Retrieved from `https://www.hindawi.com/journals/mpe/2010/621670/`

Friston, K., Trujillo-Barreto, N., & Daunizeau, J. (2008a). Dem: a variational treatment of dynamic systems. *Neuroimage*, *41*(3), 849–885.

Friston, K., Trujillo-Barreto, N., & Daunizeau, J. (2008b). Dem: a variational treatment of dynamic systems. *Neuroimage*, *41*(3), 849–885.

Ghahramani, Z., Beal, M. J., et al. (2000). *Graphical models and variational methods*. Advanced mean field methods-theory and practice. MIT Press.

Guerguiev, J., Lillicrap, T. P., & Richards, B. A. (2017). Towards deep learning with segregated dendrites. *Elife*, *6*, e22901.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hohwy, J., Roepstorff, A., & Friston, K. (2008). Predictive coding explains binocular rivalry: An epistemological review. *Cognition*, *108*(3), 687–701.

Howard-Jones, P. A., Bogacz, R., Yoo, J. H., Leonards, U., & Demetriou, S. (2010). The neural mechanisms of learning from competitors. *Neuroimage*, *53*(2), 790–799.

Jazwinski, A. H. (2007). *Stochastic processes and filtering theory*. Courier Corporation.

Jiang, L. P., & Rao, R. P. (2022). Dynamic predictive coding: A new model of hierarchical sequence learning and prediction in the cortex. *bioRxiv*.

Jordan, M. I. (1990). Attractor dynamics and parallelism in a connectionist sequential machine. In *Artificial neural networks: concept learning* (pp. 112–127).

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kanai, R., Komura, Y., Shipp, S., & Friston, K. (2015). Cerebral hierarchies:

predictive processing, precision and the pulvinar. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *370*(1668), 20140169. Retrieved from `https://royalsocietypublishing.org/doi/full/10.1098/rstb.2014.0169`

Keller, G. B., & Mrsic-Flogel, T. D. (2018). Predictive processing: a canonical cortical computation. *Neuron*, *100*(2), 424–435.

Keysers, C., Xiao, D.-K., Földiák, P., & Perrett, D. I. (2001). The speed of sight. *Journal of cognitive neuroscience*, *13*(1), 90–101.

Knill, D. C., & Pouget, A. (2004). The bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, *27*(12), 712–719.

Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork rnn. In *International conference on machine learning* (pp. 1863–1871).

Kutschireiter, A., Surace, S. C., Sprekeler, H., & Pfister, J.-P. (2017). Nonlinear bayesian filtering and learning: a neuronal dynamics for perception. *Scientific reports*, *7*(1), 1–13.

Lotter, W., Kreiman, G., & Cox, D. (2016). Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*. Retrieved from `https://arxiv.org/abs/1605.08104`

Millidge, B., Seth, A., & Buckley, C. L. (2021). Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*.

Millidge, B., & Shillcock, R. (2019). Fixational eye movements: Data augmentation for the brain? *PsyArXiv*.

Millidge, B., Tschantz, A., & Buckley, C. L. (2020). Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*. Retrieved from `https://arxiv.org/abs/2006.04182`

Millidge, B., Tschantz, A., Seth, A., & Buckley, C. (2021). Neural kalman filtering. *arXiv preprint arXiv:2102.10021*.

Millidge, B., Tschantz, A., Seth, A., & Buckley, C. L. (2020). Relaxing the constraints on predictive coding models. *arXiv preprint arXiv:2010.01047*.

Moeller, M., Manohar, S., & Bogacz, R. (2022). Uncertainty–guided learning with scaled prediction errors in the basal ganglia. *PLoS computational biology*, *18*(5), e1009816.

Mumford, D. (1992). On the computational architecture of the neocortex. *Biological cybernetics*, *66*(3), 241–251.

Navajas, J., Hindocha, C., Foda, H., Keramati, M., Latham, P. E., & Bahrami, B. (2017). The idiosyncratic nature of confidence. *Nature human behaviour*, *1*(11), 810–818.

Neal, R. M., & Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models* (pp. 355–368). Springer.

Ororbia, A., & Kifer, D. (2022). The neural coding framework for learning generative models. *Nature communications*, *13*(1), 1–14.

Ororbia, A., Mali, A., Giles, C. L., & Kifer, D. (2020). Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Transactions on Neural Networks and Learning Systems*.

Parr, T., & Friston, K. J. (2018). The discrete and continuous brain: from decisions to movement—and back again. *Neural computation*, *30*(9), 2319–2347.

Pouget, A., Beck, J. M., Ma, W. J., & Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nature neuroscience*, *16*(9), 1170.

Rainer, G., Asaad, W. F., & Miller, E. K. (1998). Selective representation of relevant information by neurons in the primate prefrontal cortex. *Nature*, *393*(6685), 577–579.

Rao, R. P. (1997). Correlates of attention in a model of dynamic visual recognition. *Advances in neural information processing systems*, *10*.

Rao, R. P. (1999). An optimal estimation approach to visual perception and learning. *Vision research*, *39*(11), 1963–1989.

Rao, R. P., & Ballard, D. H. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural computation*, *9*(4), 721–763.

Rao, R. P., & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, *2*(1), 79–87. Retrieved from `https://www.nature.com/articles/nn0199_79`

Ruck, D. W., Rogers, S. K., Kabrisky, M., Maybeck, P. S., & Oxley, M. E. (1992). Comparative analysis of backpropagation and the extended kalman filter for

training multilayer perceptrons. *IEEE Transactions on Pattern Analysis & Machine Intelligence*(6), 686–691.

Ruzzoli, M., Torralba, M., Fernández, L. M., & Soto-Faraco, S. (2019). The relevance of alpha phase in human perception. *Cortex*, *120*, 249–268.

Sacramento, J., Costa, R. P., Bengio, Y., & Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in neural information processing systems* (pp. 8721–8732).

Schultz, W. (1998). Predictive reward signal of dopamine neurons. *Journal of neurophysiology*, *80*(1), 1–27.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*(5306), 1593–1599.

Song, Y., Lukasiewicz, T., Xu, Z., & Bogacz, R. (2020). Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *Advances in Neural Information Processing Systems*, *33*. Retrieved from `https://proceedings.neurips.cc/paper/2020/hash/fec87a37cdeec1c6ecf8181c0aa2d3bf-Abstract.html`

Spratling, M. W. (2017). A review of predictive coding algorithms. *Brain and cognition*, *112*, 92–97.

Srinivasan, M. V., Laughlin, S. B., & Dubs, A. (1982). Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *216*(1205), 427–459. Retrieved from `https://royalsocietypublishing.org/doi/abs/10.1098/rspb.1982.0085?casa_token=gdNrGbAlmC8AAAAA%3Ac1xArFgNym4QLB0vI-dDd0ywIS0ozVZjzjnhogf4CVpFZi2zIW8cMU3OIZwvV8cFCoVqAaDOFo_IFDY`

Stengel, R. F. (1986). *Stochastic optimal control: theory and application.* John Wiley & Sons, Inc.

Sun, W., & Orchard, J. (2020). A predictive-coding network that is both discriminative and generative. *Neural Computation*, *32*(10), 1836–1862. Retrieved from `https://www.mitpressjournals.org/doi/abs/10.1162/neco_a_01311`

Tallec, C., & Ollivier, Y. (2018). Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*.

Tang, M., Salvatori, T., Millidge, B., Song, Y., Lukasiewicz, T., & Bogacz, R. (2023). Recurrent predictive coding models for associative memory employing

covariance learning. *PLOS Computational Biology*, *19*(4), e1010719.

Thorpe, S., Fize, D., & Marlot, C. (1996). Speed of processing in the human visual system. *nature*, *381*(6582), 520–522.

Thunell, E., & Thorpe, S. J. (2019). Memory for repeated images in rapid-serial-visual-presentation streams of thousands of images. *Psychological science*, *30*(7), 989–1000.

Tschantz, A., Millidge, B., Seth, A. K., & Buckley, C. L. (2022). Hybrid predictive coding: Inferring, fast and slow. *arXiv preprint arXiv:2204.02169*.

Walsh, K. S., McGovern, D. P., Clark, A., & O'Connell, R. G. (2020). Evaluating the neurophysiological evidence for predictive processing as a model of perception. *Annals of the New York Academy of Sciences*, *1464*(1), 242. Retrieved from `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7187369/`

Watanabe, E., Kitaoka, A., Sakamoto, K., Yasugi, M., & Tanaka, K. (2018). Illusory motion reproduced by deep neural networks trained for prediction. *Frontiers in psychology*, *9*, 345.

Weilnhammer, V., Stuke, H., Hesselmann, G., Sterzer, P., & Schmack, K. (2017). A predictive coding account of bistable perception-a model-based fmri study. *PLoS computational biology*, *13*(5), e1005536.

Welch, G., Bishop, G., et al. (1995). *An introduction to the kalman filter*. Citeseer.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, *1*(4), 339–356.

Whittington, J. C., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, *29*(5), 1229–1262. Retrieved from `https://www.mitpressjournals.org/doi/full/10.1162/NECO_a_00949`

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229–256.

Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent. *Backpropagation: Theory, architectures, and applications*, *433*, 17.

Wilson, R., & Finkel, L. (2009). A neural implementation of the kalman filter. In *Advances in neural information processing systems* (pp. 2062–2070).

A Appendix: Derivation of recursive Bayesian estimation

Here we provide detailed derivations of Equation 16, showing that the original Bayesian filtering problem can be simplified as a recursive estimation process that finally leads to Kalman filtering and our own temporal predictive coding:

$$
\begin{aligned}
p(x_k|y_{1:k}) &= \int p(x_{1:k}|y_{1:k})dx_1, \ldots, dx_{k-1} \\
&\overset{(a)}{\propto} \int p(y_{1:k}|x_{1:k})p(x_{1:k})dx_1, \ldots, dx_{k-1} \\
&\overset{(b)}{=} \int \prod_{i=1}^{k} p(y_i|x_i)p(x_i|x_{i-1})dx_1, \ldots, dx_{k-1} \\
&= p(y_k|x_k) \int p(x_k|x_{k-1}) \prod_{i=1}^{k-1} p(y_i|x_i)p(x_i|x_{i-1})dx_1, \ldots, dx_{k-1} \\
&= p(y_k|x_k) \int p(x_k|x_{k-1}) \left[\int \prod_{i=1}^{k-1} p(y_i|x_i)p(x_i|x_{i-1})dx_1, \ldots, dx_{k-2} \right] dx_{k-1} \\
&\overset{(c)}{\propto} p(y_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}
\end{aligned}
\tag{33}
$$

where step (a) is achieved by Bayes' rule, step (b) is achieved by the Markov assumption we made on the hidden states, and step (c) is achieved by observing that the integral in the square bracket in the penultimate line is proportional to the Bayesian filtering posterior at the previous step $k-1$. This can be observed by comparing it with the expression on the third line of the above equations. Importantly, this expression provides a recursive solution to the problem i.e., knowing the posterior at step $k-1$, we could estimate the posterior at step k.

B Appendix: Derivation of the update rules for the models

Here we derive the update rule for $\hat{x}_k$ that underlies both predictive coding and Kalman filtering (Equations 22 and 26). Notice that the objective functions for

these two models follow a unified form:

$$\mathcal{F}_k = (y_k - Cx_k)^T \Sigma_y^{-1} (y_k - Cx_k) + (x_k - \hat{x}_k^-)^T S^{-1} (x_k - \hat{x}_k^-) \tag{34}$$

where $S = \Sigma_x$ for predictive coding and $S = A\Sigma_{k-1}A^T + \Sigma_x$ for Kalman filtering. To obtain $\hat{x}_k$ that minimizes $\mathcal{F}_k$, we first take the derivative of $\mathcal{F}_k$ with respect to x_k:

$$\frac{\partial \mathcal{F}_k}{\partial x_k} = (C^T \Sigma_y^{-1} C + S^{-1}) x_k - (C^T \Sigma_y^{-1} y_k + S^{-1} \hat{x}_k^-) \tag{35}$$

Then, by setting the derivative to 0 we have the optimal $\hat{x}_k$:

$$\begin{aligned}
\hat{x}_k &= (C^T \Sigma_y^{-1} C + S^{-1})^{-1} (C^T \Sigma_y^{-1} y_k + S^{-1} \hat{x}_k^-) \\
&\overset{(a)}{=} [S - SC^T (\Sigma_y + CSC^T)^{-1} CS](C^T \Sigma_y^{-1} y_k + S^{-1} \hat{x}_k^-) \\
&\overset{(b)}{=} [S - KCS](C^T \Sigma_y^{-1} y_k + S^{-1} \hat{x}_k^-) \\
&= \hat{x}_k^- - KC\hat{x}_k^- + [SC^T \Sigma_y^{-1} - KCSC^T \Sigma_y^{-1}] y_k \\
&= \hat{x}_k^- - KC\hat{x}_k^- + [KK^{-1} SC^T \Sigma_y^{-1} - KCSC^T \Sigma_y^{-1}] y_k \\
&= \hat{x}_k^- - KC\hat{x}_k^- + K[(\Sigma_y + CSC^T) C^{-T} S^{-1} SC^T \Sigma_y^{-1} - CSC^T \Sigma_y^{-1}] y_k \\
&= \hat{x}_k^- - KC\hat{x}_k^- + K y_k \\
&= \hat{x}_k^- + SC^T (\Sigma_y + CSC^T)^{-1} (y_k - C\hat{x}_k^-)
\end{aligned} \tag{36}$$

In step (a) we use the Woodbury matrix inversion identity and in step (b) we replace $SC^T(\Sigma_y + CSC^T)^{-1}$ with K. Substituting $S = \Sigma_x$ for predictive coding and $S = A\Sigma_{k-1}A^T + \Sigma_x$ for Kalman filtering we get Equations 22 and 26 respectively.

C Appendix: Estimating uncertainty in temporal predictive coding

In the previous derivations, we have shown that temporal predictive coding differs from Kalman filtering in that it does not *propagate* the uncertainty estimated at the previous time step. Initially, this may seem to result from the fact that temporal predictive coding does not *model* the uncertainty at any step at all

since it performs MAP estimation and does not have a separate estimation of the posterior covariance. However, as we show in this appendix, even if we choose to model the posterior covariance i.e., model the full Gaussian posterior, the optimal covariance Σ_k^* at step k will still be independent of the previously estimated Σ_{k-1}.

As was shown in Equation 25, the objective of temporal predictive coding simplifies to maximizing the following joint probability with respect to x_k:

$$p(y_k, x_k|\hat{x}_{k-1}) = p(x_k|y_k, \hat{x}_{k-1})p(y_k|\hat{x}_{k-1}) \tag{37}$$

Since $p(y_k|\hat{x}_{k-1})$ does not depend on x_k, this is equivalent to maximizing the posterior probability $p(x_k|y_k, \hat{x}_{k-1})$. If, instead of finding the maximum of this posterior, we want to model the full posterior distribution, a common approach is to perform variation inference, where we assume an approximate posterior distribution $q(x_k)$ that approximates this true posterior. Here, we assume that $q(x_k)$ is Gaussian:

$$q(x_k) = \mathcal{N}(\mu_k, \Sigma_k) \tag{38}$$

We want to find the distribution parameters μ_k and Σ_k that minimizes the divergence between the approximate and true posteriors:

$$q^* = \underset{q}{\operatorname{argmin}}\, D_{\mathrm{KL}}\left[q(x_k)||p(x_k|y_k, \hat{x}_{k-1})\right] \tag{39}$$

which is equivalent to minimizing the following KL divergence (see (Bogacz, 2017) for details):

$$q^* = \underset{q}{\operatorname{argmin}}\, D_{\mathrm{KL}}\left[q(x_k)||p(x_k, y_k|\hat{x}_{k-1})\right] \tag{40}$$

Following the definition of KL divergence, we can explicitly write this KL divergence as:

$$\begin{aligned} D_{\mathrm{KL}}\left[q(x_k)||p(x_k, y_k|\hat{x}_{k-1})\right] &= \mathbb{E}_q[\log q(x_k)] - \mathbb{E}_q[\log p(x_k, y_k|\hat{x}_{k-1})] \\ &= \mathbb{E}_q[\log q(x_k)] - \mathbb{E}_q[\log p(y_k|x_k)] - \mathbb{E}_q[\log p(x_k|\hat{x}_{k-1})] \end{aligned} \tag{41}$$

We now evaluate the three expection terms. For the first term $\mathbb{E}_q[\log q(x_k)]$, notice that it is simply the negative entropy of the distribution $q(x_k)$. Since $q(x_k)$ is

Gaussian, this term is simply:

$$\begin{aligned}\mathbb{E}_q[\log q(x_k)] &= -\frac{D_x}{2}\left(1+\log 2\pi\right) - \frac{1}{2}\log|\Sigma_k| \\ &= c_0 - \frac{1}{2}\log|\Sigma_k|\end{aligned} \tag{42}$$

where we assume x_k has dimension D_x for any k. Since the first term is independent of Σ_k, we denote it as a constant c_0.

For the second term, since we assume that the generative process $y_k|x_k \sim \mathcal{N}(Cx_k, \Sigma_y)$, we can write it as:

$$\begin{aligned}\mathbb{E}_q[\log p(y_k|x_k)] &= \mathbb{E}_q\left[-\frac{D_y}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_y| - \frac{1}{2}(y_k - Cx_k)^T\Sigma_y^{-1}(y_k - Cx_k)\right] \\ &= -\frac{D_y}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_y| - \frac{1}{2}\mathbb{E}_q\left[(y_k - Cx_k)^T\Sigma_y^{-1}(y_k - Cx_k)\right] \\ &\overset{(a)}{=} c_1 + y_k^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}\mathbb{E}_q\left[x_k^TC^T\Sigma_y^{-1}Cx_k\right] \\ &= c_1 + y_k^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}\mathbb{E}_q\left[tr(x_k^TC^T\Sigma_y^{-1}Cx_k)\right] \\ &= c_1 + y_k^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}tr\left(C^T\Sigma_y^{-1}C\mathbb{E}_q\left[x_kx_k^T\right]\right) \\ &\overset{(b)}{=} c_1 + y_k^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}tr\left(C^T\Sigma_y^{-1}C\left[\Sigma_k + \mu_k\mu_k^T\right]\right) \\ &= c_1 + y_k^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}\mu_k^TC^T\Sigma_y^{-1}C\mu_k - \frac{1}{2}tr\left(C^T\Sigma_y^{-1}C\Sigma_k\right)\end{aligned} \tag{43}$$

where we assume the observations $y_k \in \mathbb{R}^{D_y}$, and $tr()$ denotes the trace of matrix. In step (a) we use the fact that the first two terms in the second line are independent of μ_k and Σ_k and thus can be denoted as a constant c_1. In step (b) we use the fact that $cov(x,x) = \mathbb{E}[xx^T] - \mathbb{E}[x]\mathbb{E}[x]^T$.

Similarly, for the third term, since we assumed that $x_k|\hat{x}_{k-1} \sim \mathcal{N}(A\hat{x}_{k-1}, \Sigma_x)$, we

can write it as:

$$
\begin{aligned}
\mathbb{E}_q[\log p(x_k|\hat{x}_{k-1})] &= -\frac{D_x}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_x| \\
&\quad - \mathbb{E}_q\left[\frac{1}{2}(x_k - A\hat{x}_{k-1})^T\Sigma_x^{-1}(x_k - A\hat{x}_{k-1})\right] \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}\mathbb{E}_q\left[x_k^T\Sigma_x^{-1}x_k\right] \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}\mathbb{E}_q\left[tr(x_k^T\Sigma_x^{-1}x_k)\right] \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}\mathbb{E}_q\left[tr(\Sigma_x^{-1}x_k x_k^T)\right] \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}tr(\Sigma_x^{-1}\mathbb{E}_q\left[x_k x_k^T\right]) \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}tr(\Sigma_x^{-1}\left[\Sigma_k + \mu_k\mu_k^T\right]) \\
&= c_2 + \hat{x}_{k-1}^T A^T\Sigma_x^{-1}\mu_k - \frac{1}{2}\mu_k\Sigma_x^{-1}\mu_k - \frac{1}{2}tr(\Sigma_x^{-1}\Sigma_k)
\end{aligned}
\tag{44}
$$

Putting these three terms together, we can define the objective function $\mathcal{F}_k$ as:

$$
\begin{aligned}
\mathcal{F}_k &:= D_{\mathrm{KL}}\left[q(x_k)||p(x_k, y_k|\hat{x}_{k-1})\right] \\
&= c + \frac{1}{2}\mu_k^T(\Sigma_x^{-1} + C^T\Sigma_y^{-1}C)\mu_k - (y_k^T\Sigma_y^{-1}C + \hat{x}_{k-1}^T A^T\Sigma_x^{-1})\mu_k \\
&\quad - \frac{1}{2}\log|\Sigma_k| + \frac{1}{2}tr\left(C^T\Sigma_y^{-1}C\Sigma_k\right) + \frac{1}{2}tr\left(\Sigma_x^{-1}\Sigma_k\right)
\end{aligned}
\tag{45}
$$

Taking the derivative of $\mathcal{F}_k$ with respective to μ_k and Σ_k and set them to 0 we get (notice that $\mathcal{F}_k$ is a convex function with respect to both μ_k and Σ_k):

$$
\mu_k^* = (\Sigma_x^{-1} + C^T\Sigma_y^{-1}C)^{-1}(C^T\Sigma_y^{-1}y_k + \Sigma_x^{-1}A\hat{x}_{k-1}) \tag{46}
$$

$$
\Sigma_k^* = (\Sigma_x^{-1} + C^T\Sigma_y^{-1}C)^{-1} \tag{47}
$$

Notice that μ_k^* is exactly the MAP estimate $\hat{x}_k$ of x_k we obtained in the main text, and Σ_k^* is independent of the previously estimated Σ_{k-1}.