The covariance perceptron: A new paradigm for classification and processing of time series in recurrent neuronal networks

M Gilson*, D Dahmen*, R Moreno-Bote, A Insabato, M Helias

Abstract

Learning in neuronal networks has developed in many directions, from image recognition and speech processing to data analysis in general. Most theories that rely on gradient descents tune the connection weights to map a set of input signals to a set of activity levels in the output of the network, thereby focusing on the first-order statistics of the network activity. Fluctuations around the desired activity level constitute noise in this view. Here we propose a conceptual change of perspective by employing temporal variability to represent the information to be learned, rather than merely being the noise that corrupts the mean signal. The new paradigm tunes both afferent and recurrent weights in a network to shape the input-output mapping for covariances, the second-order statistics of the fluctuating activity. When including time lags, covariance patterns define a natural metric for time series that capture their propagating nature. Notably, this viewpoint differs from recent studies that focused on noise correlation and (de)coding, because the activity variability here is the basis for stimulus-related information to be learned by neurons. We develop the theory for classification of time series based on their spatio-temporal covariances, which reflect dynamical properties. Closed-form expressions reveal identical pattern capacity in a binary classification task compared to the ordinary perceptron. The information density, however, exceeds the classical counterpart by a factor equal to the number of input neurons. We finally demonstrate the crucial importance of recurrent connectivity for transforming spatio-temporal covariances to spatial covariances.

1 Introduction

A fundamental cognitive task that is commonly performed by humans and animals is the classification of time-dependent signals. For example, in the perception of auditory signals, the listener needs to distinguish the meaning of different sounds: The neuronal system receives a series of pressure values, the stimulus, and needs to assign a category, for example whether the sound indicates the presence of a predator or a prey.

Neuronal information processing systems are set apart from traditional paradigms of information processing by their ability to be trained, rather than being algorithmically programmed. The same architecture, a network composed of neurons connected by synapses, can be adapted to perform different classification tasks. The physical implementation of learning predominantly consists of adapting the connection strengths between neurons —a mechanism termed synaptic plasticity. Learning in artificial neuronal networks is often formulated as a gradient descent for an objective function that measures the mismatch between the desired and the actual outputs. This idea forms the basis of supervised learning [8]. The most prominent examples of such synaptic update rules are the delta rule for the perceptron neuronal network [37, 28, 40] and error back-propagation [38]. These led to modern classification machines, like deep learning and convolutional networks [24, 39]. Their success was only unleashed rather recently by the increased computational power of modern computers and large amounts of available training data, both required for successful training. A key problem to be solved in the improvement of a neuronal information processing is thus to devise new and efficient paradigms for training.

A central feature of the training design is how the physical stimulus is represented in terms of neuronal activity. The traditional view regards the time series of neuronal activity as a succession of snap shots, each of which is possibly corrupted by noise. Thus, the mean activity is regarded as the relevant information of the signal; the variance that measures departures from this mean quantifies the noise. The task of the neuronal network is to robustly classify time-varying input signals despite their variability within

each category. This view has led to efficient technical solutions to train neuronal networks by recurrent back-propagation [34] or by back-propagation through time [33].

The representation of information by the mean activity is, however, challenged by two observations in biological neuronal networks. First, neuronal activity in cortex shows a considerable amount of variability even if the very same experimental paradigm is repeated multiple times [2]; neurons also tend to respond more reliably to transients, than to steady states [26]. Previous studies have proposed that this variability may be related to probabilistic representations of the environment in a Bayesian fashion [6, 32]. Second, synaptic plasticity, the biophysical implementation of learning, has been shown to depend upon the temporal activity of the presynaptic and the postsynaptic neurons [27, 7], which can be formalized using the covariance of the neuronal activity [22, 16]. Experimental and theoretical evidence thus points to a relation between the variability of neuronal activity and the representation of the stimulus.

These observations raise several questions: How can a neuronal system perform its function despite this large amount of variability? Moving a step further, can variability even be employed to represent information in its covariance structure, as suggested by covariance-dependent synaptic plasticity and by the preferred response of neurons to transients? If so, how to train networks that employ such representations? Finally, one may wonder if covariance-based learning is superior to technical solutions that employ a mean-based representation, providing a reason why it may be used by neuronal circuits.

We here present a novel paradigm that employs the covariances of fluctuating activity to represent stimulus information. We show how the input-output mapping for covariances can be learned in a recurrent network architecture by efficiently trained the connectivity weights by a gradient-descent learning rule. We find that covariance-based classification is at least as robust as with the mean perceptron. Analyzing the capacity of the network in terms of the maximum number of correctly classifiable stimuli shows that it is en par with the traditional architecture; In terms of memory capacity in bits, however, it largely exceeds the traditional paradigm by a factor m, the number of input neurons. Our work thus provides evidence that covariance-based information processing in a biological context can reach superior performance compared to paradigms that have so far been used in artificial neuronal networks.

The remainder of the article is organized as follows: Section 2 formalizes the main idea of this article, the use of the covariance of stochastic fluctuations to represent information. Section 3 considers a network with feed-forward connectivity that is trained in an online manner, as a time-varying process, to implement a desired mapping from the input covariance to the output covariance. We derive a gradient-descent learning rule that adapts the feed-forward connections and examine the network training in theory, for infinite observation time, as well as for time series of limited duration. Section 4 focuses on the capacity of the covariance perceptron in the case of assigning a binary class label to a bipartite set of noiseless input covariance patterns. This capacity is also compared with the classical perceptron. Section 5 extends the online training of Section 3 to a network with both afferent and recurrent connections. We show how recurrent connections allow us to exploit the temporal structure of input covariances as an additional dimension for stimulus representation that can be mapped to output representations. Importantly, we demonstrate the specific role played by the recurrent network connectivity when the information to learn is in the temporal dimension of covariances, but not in its spatial dimension.

2 Covariance-based representation of information

The present paper considers the problem of information transmission conveyed by a time series in a neuronal network, as illustrated in Fig. 1A. To fix ideas, consider a discrete-time network dynamics as defined by a multivariate autoregressive (MAR) process [25]. The activity of the m inputs $x^t_{1\leq k\leq m}$ is described by a stochastic process in discrete time $t \in \mathcal{Z}$. The inputs drive the activity $y^t_{1\leq i\leq n}$ of the n output neurons via connections $B \in \mathbb{R}^{n\times m}$, which form the afferent connectivity. The outputs also depend on their own immediate past activity (i.e. with a unit time shift) through the connections $A \in \mathbb{R}^{n\times n}$, the recurrent connectivity, as

$$y^t_i = \sum_{1\leq j\leq n} A_{ij} y^{t-1}_j + \sum_{1\leq k\leq m} B_{ik} x^t_k \ , \tag{1}$$

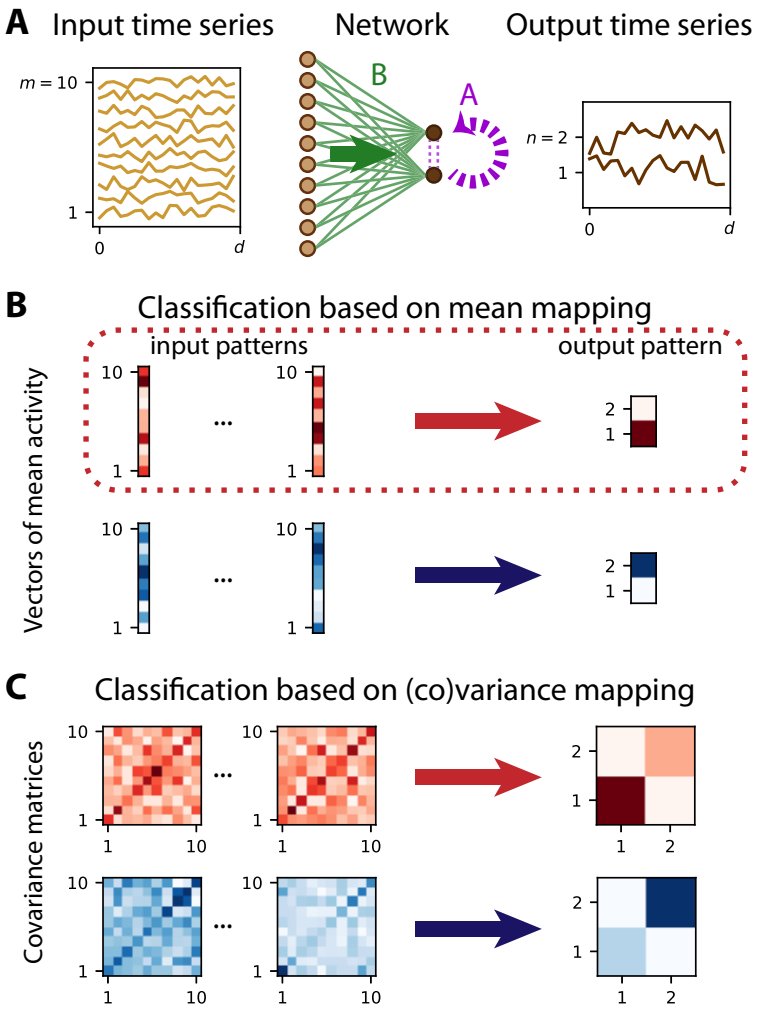

Figure 1: **From mean-based to covariance-based classification of a time series. A** Network with $n = 2$ output nodes generates a time series (in dark brown on the right) from the noisy time series of $m = 10$ input nodes (in light brown on the left). The afferent (feed-forward) connections B (green links and green arrow) and, when existing, recurrent connections A (purple dashed links and arrow) determine the input-output mapping. We observe the time series over a window of duration d. **B** The time series in panel A determines the m-dimensional vector of mean activities, averaged over the observation window d (darker pixels indicate higher values). The classification scheme is implemented by tuning the connectivity weights A and B such that several input patterns of mean activity (m-dimensional vectors on the left-hand side) are mapped to the same output pattern (n-dimensional vector of the right-hand side), thereby representing two categories in red (dotted rectangle, neuron 1 highly active) and blue (neuron 2 highly active). The mapping between input and output (mean) vectors in Eq. (2) corresponds to the classical perceptron. **C** The covariance perceptron maps the covariance patterns of an input time series ($m \times m$ matrices on the left-hand side) to covariance patterns of the output time series ($n \times n$ matrices on the right-hand side), see Eq. (3) for their formal definition. Here the two classes are represented by larger variance of either of the two nodes.

illustrated in Fig. 1A. We define the mean activities

$$\begin{aligned} X_k &\equiv \langle x_k^t \rangle \\ Y_i &\equiv \langle y_i^t \rangle \ , \end{aligned} \tag{2}$$

where the angular brackets indicate the average over realizations and over a period of duration d.

A classical assumption is that the information is represented by the mean of each input (see Fig. 1B). By tuning the connection weights, A and B, patterns in the mean input activity can be mapped to desired patterns in the mean activity of the output. The example in Fig. 1B maps a bipartite set of patterns to either of two output patterns, each of which representing one class; the network performs a binary classification of the incoming stimuli. Applying a threshold function to the output yields the classical 'mean perceptron' [28].

The present study proposes a different representation of information that employs temporal fluctuations, rather than the mean activity. We thus move from the first-order statistics, the mean, to the second-order statistics, the covariance of the statistical fluctuations of the network activity. The input and output covariances, with $\tau \in \mathcal{Z}$ being the time lag, are defined as

$$\begin{aligned} P_{kl}^\tau &\equiv \langle x_k^{t+\tau} x_l^t \rangle - \langle x_k^{t+\tau} \rangle \langle x_l^t \rangle \\ Q_{ij}^\tau &\equiv \langle y_i^{t+\tau} y_j^t \rangle - \langle y_i^{t+\tau} \rangle \langle y_j^t \rangle \ . \end{aligned} \tag{3}$$

Here we implictly assume stationarity of the inputs over the window of duration d in Fig. 1A. In this study we consider the case of vanishing mean for covariance-based classification, so the second terms on the right-hand sides disappear in Eq. (3); considerations about a mixed scenario based on both means and covariances will be discussed at the end of the article.

In this setting, the goal of learning is to shape the mapping from the input covariance P to the output covariance Q in the network in Fig. 1A. Building up on the classical 'mean perceptron' (Fig. 1B), we use classification as an example to illustrate our theory. The ansatz is that correlated fluctuations across neurons —as defined by covariances in Eq. (3)— convey information that can be used to train the network weights and then classify input time series into categories. Fig. 1C shows the concept of classifying a time series based on patterns in the covariance: The 'red class' of input covariance matrices P is mapped by the network to an output, where neuron 1 has larger variance than neuron 2. For the 'blue class' of input covariances matrices, the variance of neuron 2 exceeds that of neuron 1.

In particular, we aim to use the 'covariance perceptron' to discriminate time series that have a covariance structure that results from the input activity obeying a network dynamics itself. In this case, input and output information are of the same type, which makes the scheme represent and process information in a self-consistent manner. This opens the way to successive stages of information processing as in multilayer perceptrons. This viewpoint on signal variability radically differs from that in Fig. 1B, where the information is conveyed by the mean signal and fluctuations are noise. Conceptually, taking the second statistical order as the basis of information is an intermediate description between the detailed signal waveform and the (oversimple) mean signal. The switch from means to covariances implies that richer representations can be realized with the same number of nodes. We assess in this study how to make use of this enlarged representation space for training and classification.

3 Online learning input-output covariance mappings in feedforward networks

This section presents the concepts underlying the covariance perceptron with afferent connections B only (meaning absent recurrent connectivity $A = 0$) and compares it with the classical perceptron. The classical perceptron for means, shown in Fig. 1B, corresponds to observing the output mean vector Y for the classification of the input mean vector X in Eq. (2). It relies on the input-output mapping

$$X \mapsto Y = BX \ . \tag{4}$$

The derivation of this consistency equation —with $A = 0$ in Eq. (1)— assumes stationarity for the inputs. Under the same assumption of (second-order) stationarity, the novel proposed scheme relies on

the mapping between the input and output covariance matrices, P^0 and Q^0 in Eq. (3), namely

$$P^0 \mapsto Q^0 = BP^0B^{\mathrm{T}} \ , \tag{5}$$

where T denotes the matrix transpose. Details can be found with the derivation of the consistency equation Eq. (23) in Appendix A, which also assumes stationarity. The common property of Eqs. (4) and (5) is that both mappings are linear in the respective inputs (X and P^0). However, the second is bilinear in the weight B while the first is simply linear. Note also that this section ignores temporal correlations (i.e. we consider that $P^1 = P^{-1\mathrm{T}} = 0$); time-lagged covariances, however, do not play any role in Eq. (23) when $A = 0$.

3.1 Theory for learning of spatial covariance structure by tuning afferent connectivity

To theoretically examine covariance-based learning, we start with the abstraction of the MAR dynamics $P^0 \mapsto Q^0$ in Eq. (5). As depicted in Fig. 2A, each training step consists in presenting an input pattern P^0 to the network and the resulting output pattern Q^0 is compared to the objective $\bar{Q}^0$ in Fig. 2B. For illustration, we use two categories (red and blue) of 5 input patterns each, as represented in Fig. 2C-D. To properly test the learning procedure, noise is artificially added to the presented covariance pattern; compare the left matrix in Fig. 2A to the top left matrix in Fig. 2C. The purpose is to mimic the variability of covariances estimated from a (simulated) time series of finite duration (see Fig. 1), without taking into account the details of the sampling noise. The update ΔB_{ik} for each afferent weight B_{ik} is obtained by minimizing the distance in Eq. (25) between the actual and the desired output covariance

$$\begin{aligned} \Delta B_{ik} &= \eta_B \left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial B_{ik}} \\ &= \eta_B \left(\bar{Q}^0 - Q^0\right) \odot \left(U^{ik}P^0B^{\mathrm{T}} + BP^0U^{ik\mathrm{T}}\right) \ , \end{aligned} \tag{6}$$

where U^{ik} is a $m \times m$ matrix with 0s everywhere except for element (i,k) that is equal to 1; this update rule is obtained from the chain rule in Eq. (26), combining Eqs. (27) and (30) with $P^{-1} = 0$ and $A = 0$ (see Appendix B). Here η_B denotes the learning rate and the symbol $\odot$ indicates the element-wise multiplication of matrices followed by the summation of the resulting elements —or alternatively the scalar product of the vectorized matrices. Note that, although this operation is linear, the update for each matrix entry involves U^{ik} that selects a single non-zero row for $U^{ik}P^0B^{\mathrm{T}}$ and a single non-zero column for $BP^0U^{ik\mathrm{T}}$. Therefore, the whole-matrix expression corresponding to Eq. (6) is different from $(\bar{Q}^0 - Q^0)P^0B^{\mathrm{T}} + BP^0(\bar{Q}^0 - Q^0)^{\mathrm{T}}$, as could be naively thought.

Before training, the output covariances are rather homogeneous as in the examples of Fig. 2C-D (initial Q^0) because the weights are initialized with similar random values. During training, the afferent weights B_{ik} in Fig. 2E become specialized and tend to stabilize at the end of the optimization. Accordingly, Fig. 2F shows the decrease of the error E^0 between Q^0 and $\bar{Q}^0$ defined in Eq. (25). After training, the output covariances (final Q^0 in Fig. 2C-D) follow the desired objective patterns with differentiated variances, as well as small covariances.

As a consequence, the network responds to the red input patterns with higher variance in the first output node, and to the blue inputs with higher variance in the second output (top plot in Fig. 3B). We use the difference between the output variances in order to make a binary classification. The classification accuracy corresponds to the percentage of output variances with the desired ordering. The evolution of the accuracy during the optimization is shown in Fig. 3C. Initially around chance level at 50%, the accuracy increases on average due to the gradual shaping of the output by the gradient descent. The jagged evolution is due to the noise artificially added to the input covariance patterns (see the left matrix in Fig. 2A), but it eventually stabilizes around 90%. The network can also be trained by changing the objective matrices to obtain positive cross-covariances for red inputs, but not for blue inputs (Fig. 3D); in that case variances are identical for the two categories. The output cross-covariances have separated distributions for the two input categories after training (bottom plot in Fig. 3E), yielding the good classification accuracy in Fig. 3F. As a sanity check, the variance does not show a significant difference when training for cross-covariances (top plot in Fig. 3E). Conversely, the output cross-covariances are similar and very low for the variance training (bottom plot in Fig. 3B). These results demonstrate that the afferent connections can

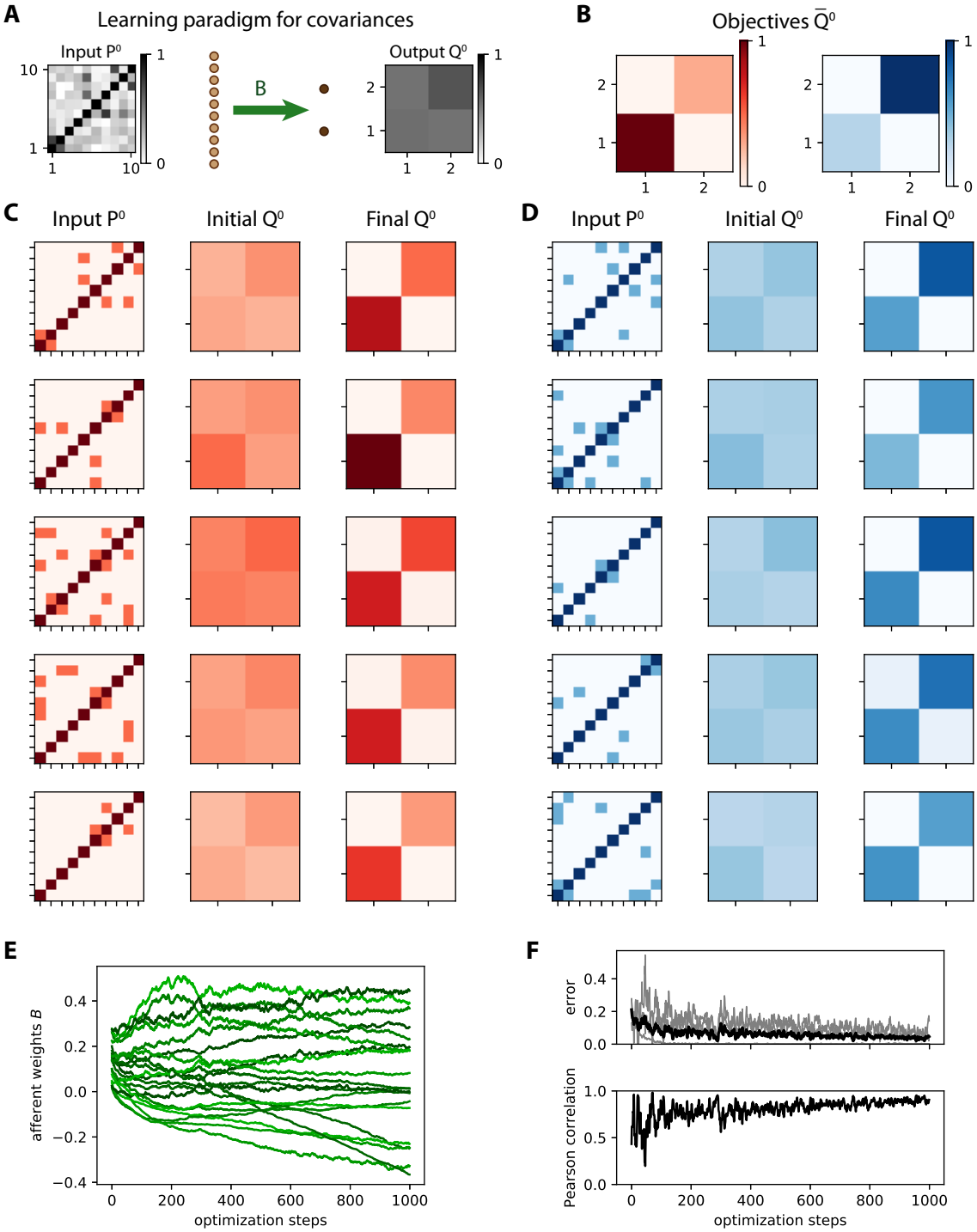

Figure 2: **Learning variances in a feed-forward network.** **A** Schematic representation of the input-output mapping for covariances defined by the afferent weight matrix B, linking $m = 10$ input nodes to $n = 2$ output nodes. **B** Objective output covariance matrices $\bar{Q}^0$ for two categories of inputs. **C** Matrix for the 5 input covariance patterns P^0 (left column), with their image under the original connectivity (middle column) and the final image after learning (right column). **D** Same as C for the second category. **E** Evolution of individual weights of matrix B during ongoing learning. **F** The top panel displays the evolution of the error between Q^0 and $\bar{Q}^0$ at each step. The total error taken as the matrix distance E^0 in Eq. (25) is displayed as a thick black curve, while individual matrix entries are represented by gray traces. In the bottom panel the Pearson correlation coefficient between the vectorized Q^0 and $\bar{Q}^0$ describes how they are "aligned", 1 corresponding to a perfect linear match.

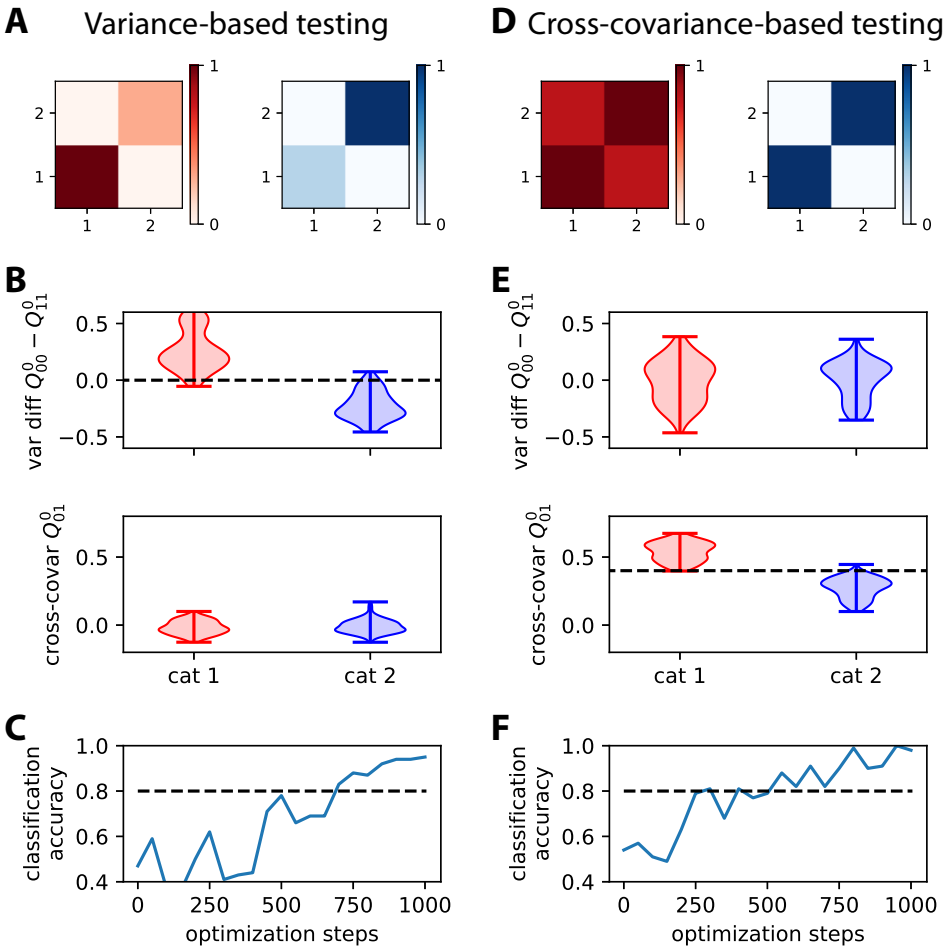

Figure 3: **Comparison between learning output patterns for variance and cross-covariance. A** The top matrices represent the two objective covariance patterns of Fig. 2B, which differ by the variances for the two nodes. **B** The plots display two measures based on the output covariance: the difference between the variances of the two nodes (top) and the cross-covariance (bottom). Each violin plot shows the distributions for the output covariance in response to 100 noisy versions of the 5 input patterns in the corresponding category. Artificial noise applied to the input covariances (see the main text about Fig. 2 for details) contributes to the spread. The separability between the red and blue distributions of the variances indicates a good classification. The dashed line is the tentative implicit boundary enforced by learning using Eq. (30) with the objective patterns in panel A: Its value is the average of the variance differences over the two categories. **C** Evolution of the classification accuracy based on the variance difference between the output nodes during the optimization. Here the binary classifier uses the differece in output variances, predicting red if the variance of the output node 1 is larger than 2, and blue otherwise. The accuracy eventually stabilizes above the dashed line that indicates 80% accuracy. **D**-**F** Same as panels A-C for two objective covariance patterns that differ by the cross-covariance level, strong for red and zero for blue. The classification in panel F results from the implicit boundary enforced by learning for the cross-covariances (dashed line in panel E), here equal to 0.4 that is the midpoint between the target cross-covariance values (0.8 for read and 0 for blue).

be efficiently trained to learn categories based on input (co)variances, just as with input vectors of mean activity in the classical perceptron.

3.2 Online learning for time series observed using a finite time window

Now we turn back to the configuration in Fig. 1C and verify that the learning procedure based on the theoretical consistency equations also works for simulated time series, where the samples of the process itself are presented, rather than their statistics embodied in the matrices P^0 and Q^0. We refer to this as online learning, but note that the covariances are estimated from an observation window, as opposed to a continuous estimation of the covariances. As before, the weight update is applied for each presentation of a pattern.

To generate the input time series, we use a superposition of independent Gaussian random variables z_l^t with unit variance (akin to white noise), which are mixed by a coupling matrix W:

$$x_k^t = \sum_{1 \le l \le m} W_{kl}\, z_l^t \ . \tag{7}$$

We use 10 patterns $P^0 = WW^{\mathrm{T}}$, where W is drawn randomly with $f = 10\%$ density of non-zero entries, so the input time series differ by their spatial covariance structure. The network has to classify these patterns based on the variance of the output nodes. The setting is shown in Fig. 4A, where only three input patterns per category are displayed.

The covariances from the time series are computed using an observation window of duration d, after discarding an initial transient period to remove the influence of initial conditions (corresponding to negative times in Fig. 4B). The window duration d affects the precision of the empirical covariances compared to their theoretical counterpart, as shown in Fig. 4C. This raises the issue of the precision required in practice for effective learning.

As expected, a longer observation duration d helps to stabilize the learning, which can be seen in the evolution of the error in Fig. 4D: the darker curves for $d = 20$ and 30 have fewer upside jumps than the lighter curve for $d = 10$. To assess the quality of the training, we repeat the simulations for 20 network and input configurations, then calculate the difference in variance between the two output nodes as in Fig. 3B-C. Training for windows with $d \ge 20$ achieve very good classification accuracy in Fig. 4E. This indicates that the covariance estimate can be evaluated with sufficient precision from only a few tens of time points. Moreover, the performance only slightly decreases for denser input patterns (Fig. 4F). Similar results can be obtained while training the cross-covariance instead of the variances.

4 Discrimination capacity for perceptron with afferent connections (offline learning)

The efficiency of the binary classification in Fig. 3 relies on tuning the weights to obtain a linear separation between the input covariance patterns. Now we consider the capacity of the covariance perceptron, namely the number of input patterns that can be discriminated in a binary classification, and compare it with the classical linear perceptron (for mean activity). There are two important differences in the present section compared to Section 3. Here we consider noiseless patterns with offline learning, meaning that the weight optimization is performed using a given number p of patterns (or pattern load) and the classification accuracy is evaluated with the same patterns. In addition, the non-linearity applied to the readout (observed output for classification) is incorporated into the weight optimization. We first present geometric considerations about the input-output mappings for the mean and covariance perceptrons. Then we analytically calculate their capacity using methods from statistical physics and compare the prediction to numerical simulation (similar to Fig. 3).

4.1 Input spaces for mean and covariance patterns

Beside the difference between the input-output mappings in terms of the weights B — bilinear for Eq. (5) versus linear for Eq. (4) — the input space has higher dimensionality for covariances than for means: $m(m+1)/2$ for P^0 including variances compared to m for X. Covariances thus offer a potentially richer

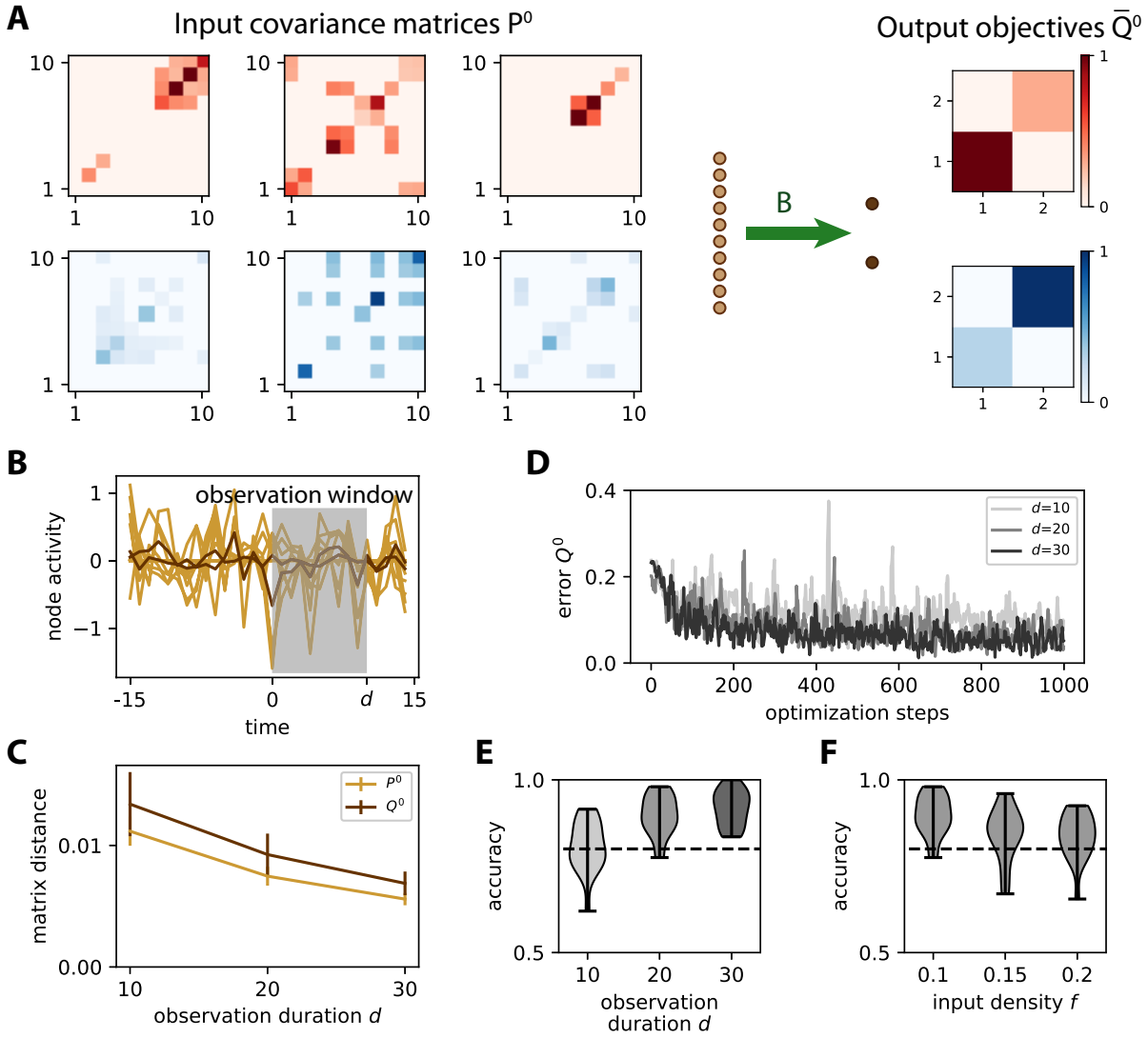

Figure 4: **Online learning input covariances by tuning afferent connectivity. A** The same network as in Fig. 2A is trained to learn the input spatial covariance structure P^0 of time series governed by the dynamics in Eq. (7). Only 3 matrices $P^0 = WW^{\mathrm{T}}$ out of the 5 for each category are displayed. Each entry in each matrix W has a probability $f = 10\%$ of being non-zero, so the actual f is heterogeneous across the W. The objective matrices (right) correspond to a specific variance pattern for the output nodes. **B** Example of simulation of the time series for the inputs (light brown traces) and outputs (dark brown). An observation window (gray area) is used to calculate the covariances from simulated time series. **C** Sampling error as measured by the matrix distance between the covariance estimated from the time series (see panel B) and the corresponding theoretical value when varying the duration d of the observation window. The error bars indicate the standard error of the mean over 100 repetitions of randomly drawn W and afferent connectivity B. **D** Evolution of the error for 3 example optimizations with various observation durations d as indicated in the legend. **E** Classification accuracy at the end of training (cf. Fig. 3C) as a function of d, pooled for 20 network and input configurations. For $d \geq 20$, the accuracy is close to 90% on average, mostly above the dashed line indicating 80%. **F** Similar plot to panel E when varying the input density of W from $f = 10$ to 20%, with $d = 20$.

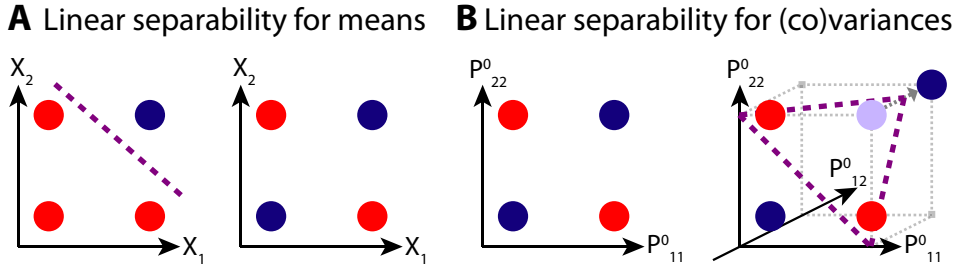

Figure 5: **Comparison between input patterns based on mean and covariance. A** Two examples of pattern classification for mean-based decoding. In the left diagram, the two categories can be linearly separated, but not in the right diagram. **B** The left diagram is the equivalent of the right diagram in panel A for variance-based decoding. The right panel extends the left one by considering the covariance P^0_{12}.

environment, but they also involve constraints related to the fact that a covariance matrix is positive semidefinite:

$$\begin{aligned} P^0_{ij} &= P^0_{ji} \,, \\ P^0_{ii} &\geq 0 \,, \\ |P^0_{ij}| &\leq \sqrt{P^0_{ii}\, P^0_{jj}} \,, \end{aligned} \tag{8}$$

for all indices i and j.

To conceptually compare the mean and the covariance perceptron, we consider an example with $m = 2$ and $n = 1$, so that the number of free parameters for classification (i.e. the afferent weights) and the dimensionality of the output are the same for both perceptrons. In the mean perceptron linear separability for the vector X is implemented by the threshold on $Y_1 = B_{11}X_1 + B_{12}X_2$ and corresponds to a line in the plane (X_1, X_2), as represented by the purple line in the left plot of Fig. 5A that separates the red and blue patterns (colored dots). The right plot of Fig. 5A, however, represents a situation where the two categories of patterns cannot be linearly separated. This corresponds to a well-known limitation of the (linear single-layer) perceptron that cannot implement a logical XOR gate [28].

The same scheme with variance is represented in the left diagram of Fig. 5B. In this example we have $Q^0_{11} = B^2_{11}P^0_{11} + B^2_{12}P^0_{22} + 2B_{11}B_{12}P^0_{12}$. In the absence of the cross-covariance P^0_{12}, the situation is similar to the equation for the mean vector, albeit being in the positive quadrant. This means that the output variance Q^0_{11} cannot implement a linear separation for the XOR configuration of input variances P^0_{11} and P^0_{22}, when they are both small or both large for the blue category, one small and the other large for the red category. Now considering P^0_{12}, we take, as an example, P^0_{11} and P^0_{22} equal to 0 or 1 for small or large values, so we obtain $Q^0_{11} \in \{B^2_{11}, B^2_{12}\}$ for the red patterns and $Q^0_{11} \in \{0, B^2_{11} + B^2_{12} + 2B_{11}B_{12}P^0_{12}\}$ for the blue patterns. Provided the blue values of Q^0_{11} are smaller than the red values, linear separation is achieved. This leads to the sufficient condition $-2B_{11}B_{12}P^0_{12} \geq \max(B^2_{11}, B^2_{12})$. Provided the weight product $B_{11}B_{12}$ and P^0_{12} have opposite signs and that $2|P^0_{12}| \geq \max(|B_{11}/B_{12}|, |B_{12}/B_{11}|)$, a pair of satisfactory weights B_{11} and B_{12} can be found. Observing that $\max(u, 1/u) \geq 1$ for all $u > 0$, a sufficient condition for separating red and blue patterns is $1/2 \leq |P^0_{12}| \leq 1$; the right bound simply comes from Eq. (8).

The increased dimensionality for the inputs related to P^0_{12} thus gives an additional "degree of freedom" for the variance-based decoding in this toy example. This is illustrated in the right diagram of Fig. 5B by the purple dashed triangle representing a plane that separates the blue and red dots: The trick is "moving" the upper right blue dot from the original position (light blue with $P^0_{12} = 0$) in front of the plane to a position behind the plane (dark blue with $P^0_{12} > 0$). This toy example suggests that separability for input covariances may have more flexibility than for input means, due to the larger dimensionality.

4.2 Theoretical capacity and information density for decoding based on output cross-covariances

To get a more quantitative handle on the capacity, we now derive a theory that is exact in the limit of large networks $m \to \infty$ and that can be compared to the seminal theory by Gardner [15] on the capacity of the mean perceptron.

So far, the weight optimization and classification have been performed in two subsequent steps, illustrated in Fig. 6A. After training the connectivity to implement a mapping from given input covariance patterns to two objective covariance patterns (left plot), classification is performed by a simple thresholding based on the observed entries of the output matrix (right plot; in practice, it is equivalent to evaluate the difference between the output variances). We now combine these two procedures into one (see the red and blue lines that "push" the dot clouds in Fig. 6B), while focusing on cross-covariances. The reason is simple: Consider a single entry of the readout covariance matrix Q^0_{ij} with $1 \leq i < j \leq n$. For binary classification, it only matters that the covariance Q^0_{ij} be separable, either above or below a given threshold. For each input pattern $P^0 = \hat{P}^r$ indexed by $1 \leq r \leq p$, we assign a label $\zeta^r_{ij} \in \{-1, 1\}$ corresponding to the position of $\hat{Q}^r_{ij}$ with respect to the threshold, where we define $\hat{Q}^r = Q^0(\hat{P}^r) = B\hat{P}^r B^{\mathrm{T}}$ following Eq. (5). We are thus demanding less to the individual matrix entry in Q^0 than in the previous learning for input-output mapping: It may live on the entire half-axis, instead of being fixed to one particular value. Note that the numbers of -1 and 1 in ζ^r_{ij} may not be exactly balanced between the two categories here.

Formalizing the classification problem, we fix an element $\hat{Q}^r_{ij}$ of the readout matrix and draw a random label $\zeta^r_{ij} \in \{-1, 1\}$ independently for each input pattern $\hat{P}^r$. An important measure for the quality of the classification is the margin defined as

$$\kappa = \min_{1 \leq r \leq p} \left(\zeta^r_{ij} \, \hat{Q}^r_{ij} \right) \ . \tag{9}$$

It measures the smallest distance over all $\hat{Q}^r_{ij}$ from the threshold, here set to 0. It plays an important role for the robustness of the classification [11], as a larger margin tolerates more noise in the input pattern before classification is compromised. The margin of the classification is illustrated in Fig. 6A, where each dot represents one of the p patterns and the color indicates the corresponding category ζ^r_{ij}. As mentioned above, we directly train the afferent weights B to maximize κ. This optimization increases the gap and thus the separability between red and blue dots in Fig. 6C. In practice, it is simpler to perform this training for a soft-minimum κ', which covaries with the true margin κ (9), as shown in Fig. 6D.

The limiting capacity is determined by the pattern load p at which the margin κ vanishes. More generally, we evaluate how many patterns we can discriminate while maintaining a given minimal margin. We consider each input covariance pattern to be of the form $\hat{P}^r = 1_m + \chi^r$ with 1_m the diagonal matrix and a random matrix χ^r with vanishing diagonal elements and off-diagonal elements, indexed by (k, l), that are independently and identically distributed as $\chi^r_{kl} = 0$ with probability $1 - f$ and $\chi^r_{kl} = \pm c$, each with probability $f/2$, while enforcing symmetry for each χ^r. Here f controls the sparseness (or density) of the cross-covariances. From Eq. (5), the task of the perceptron is to find a suitable afferent weight matrix B that leads to correct classification for all p patterns. This requirement reads, for a given margin $\kappa > 0$ and a given entry $1 \leq i < j \leq n$, as

$$\zeta^r_{ij} \left(B\hat{P}^r B^{\mathrm{T}} \right)_{ij} > \kappa \ , \quad \forall \ 1 \leq r \leq p \ . \tag{10}$$

The random ensemble for the patterns allows us to employ methods from disordered systems [13]. Closely following the computation for the mean perceptron by Gardner [15, 21], the idea is to consider the replication of several covariance perceptrons. The replicas, indexed by α and β, have the same task defined by Eq. (10). The sets of patterns $\hat{P}^r$ and labels ζ^r are hence the same for all replicas, but each replicon has its own readout matrix B^α. If the task is not too hard, meaning that the pattern load p is small compared to the number of free parameters B^α_{ik}, there are many solutions to the problem Eq. (10). One thus considers the ensemble of all solutions and computes the typical overlap between the solution B^α and B^β in two different replicas. At a certain load p there should only be a single solution left —the overlap between solutions in different replicas becomes unity. This point defines the limiting capacity $\mathcal{C}$.

Technically, the computation proceeds by defining the volume of all solutions for the whole set of

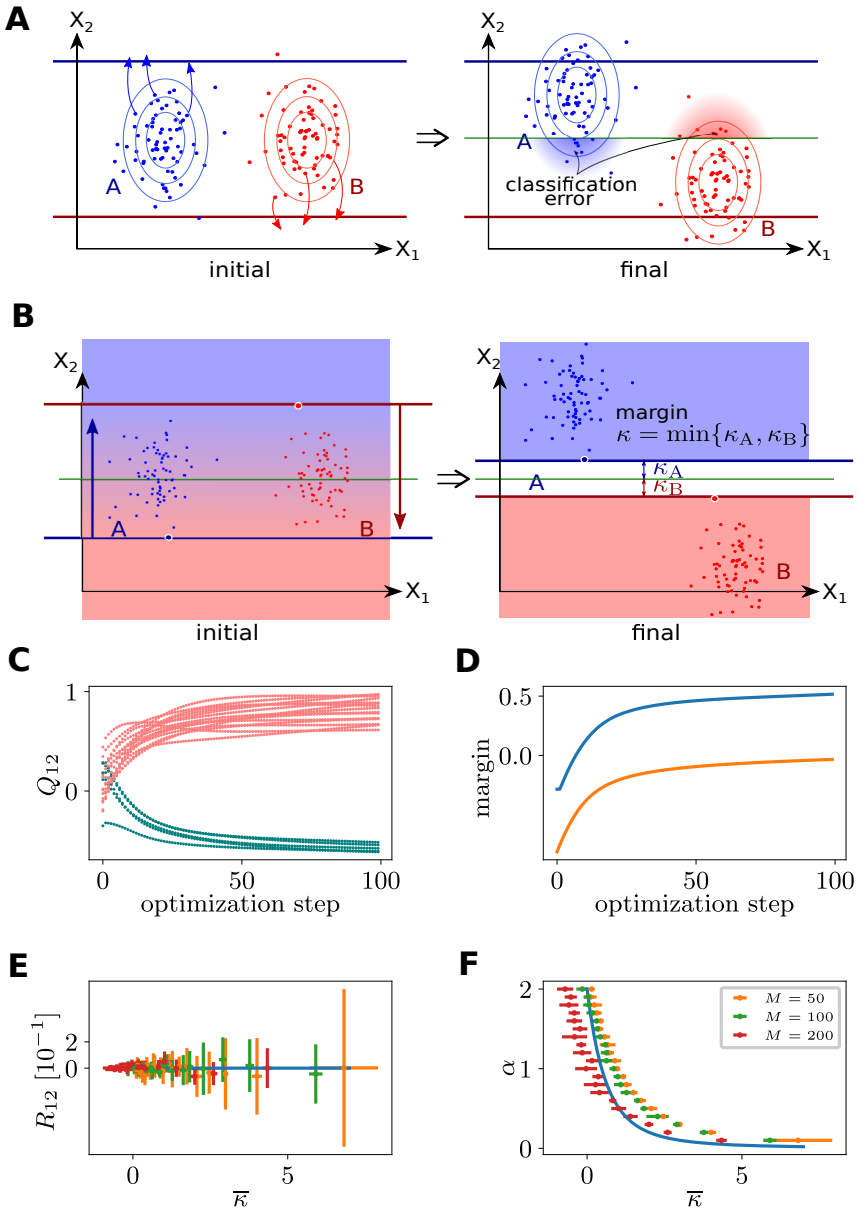

Figure 6: **Memory capacity of the covariance perceptron with a single readout** ($n = 2$). **A** Schematic representation of **B** Evolution of the readouts $\hat{Q}^r_{12} = (B\,\hat{P}^r\,B^{\mathrm{T}})_{12}$ over the optimization that maximizes the soft minimum margin $\kappa' = -\eta^{-1}\,\ln(\sum_r \exp(-\eta\,\zeta^r_{12}\,\hat{Q}^r_{12}))$ by a gradient descent with $\eta = 4$ for $m = 50$ afferent neurons. Each dot corresponds to one of the $p = 20$ patterns: red for $\zeta^r_{12} = 1$ and blue for $\zeta^r_{12} = -1$. **C** a **D** Minimum margin over training; blue: minimum margin given by (9); red: soft minimum margin κ'. **D** Overlap $R_{12} = \check{B}^{1\mathrm{T}}\check{B}^2$ between the pair of row vectors involved in the calculation of the readout Q^0_{12}. Symbols from numerical optimization; error bars show standard error from 5 realizations; solid line from theory in the large m-limit, which predicts $R_{12} \to 0$; see Eq. (C.5). **F** Total number of classifications $\mathcal{C} = p\,\frac{n(n-1)}{2}/m$ relative to the number m of inputs over the effective margin $\bar{\kappa} = \kappa/\sqrt{fc^2}$ relative to the typical variance $\sqrt{fc^2}$ of an element of the readout matrix. Symbols from numerical optimization; solid curve from theory in the large m limit given by Eq. (13). Other parameters: m given in legend; $f = 0.2$; $c = 0.5$. Numerical results in C and D from maximization of the soft-minimum margin κ'.

cross-covariances Q^0_{ij} as

$$\mathcal{V} = \int_S \mathrm{d}B \prod_{i<j} \theta\Big(\zeta^r_{ij} \left(B\,\hat{P}^r\,B^{\mathrm{T}}\right)_{ij} - \kappa\Big) \;, \tag{11}$$

where $\int_S \mathrm{d}B$ integrates over all row vectors that lie on an m-dimensional sphere S —the norm of each row vector of B is set to unity. This constraint leads to a variance of each target neuron which is approximately unity, consistent with the input population. The typical behavior of the system for large m is obtained by first taking the average of $\ln(\mathcal{V})$ over the ensemble of the patterns. It can be computed by the replica trick $\langle \ln(\mathcal{V})\rangle = \lim_{q\to 0} \left(\langle \mathcal{V}^q\rangle - 1\right)/q$ [13]. The assumption is that the system is self-averaging; for large m the capacity should not depend much on the particular realization of patterns. The leading order behavior for $m \to \infty$ follows as a mean-field approximation in the auxiliary variables $R^{\alpha\beta}_{ij} \equiv \sum_{1\le k\le m} B^\alpha_{ik} B^\beta_{jk}$, assuming symmetry over replicas and indices. Here $R^{\alpha\beta}_{ij}$ measures the overlap between the two row vectors of B^α and B^β involved in the calculation of two replica α and β. The saddle point equations —cf. Eqs. (57) and (58) in Appendix C— admit a vanishing solution $R^{\alpha\beta}_{ij} = 0$ for $i \neq j$. This result is intuitively clear: The two row vectors must be close to orthogonal, because otherwise the diagonal of the input covariance pattern $\hat{P}^r_{ii} = 1$ would cause a non-zero bias of the readout Q^α_{ij} irrespective of the label $\zeta^r = \pm 1$. Thus the perceptron would lose flexibility in assigning arbitrary labels to patterns. Fig. 6E indeed shows an overlap $R^{\alpha\alpha}_{ij}$ close to zero, observed for finite-size networks using numerical optimization.

To take into account the total number of independent binary classification labels ζ^r_{ij} relative to the input number m, we define the capacity of the perceptron as

$$\mathcal{C} = \frac{p^* n(n-1)}{2m} \;, \tag{12}$$

where p^* is the maximum load when the overlap $R^{\alpha\neq\beta}_{ii}$ approaches unity —or equivalently the volume of solutions in Eq. (11) vanishes. Our calculation in Appendix C shows that

$$\mathcal{C}(\bar{\kappa}) = \frac{n}{2} \left(\int_{-\bar{\kappa}}^{\infty} \frac{\mathrm{d}u}{\sqrt{2\pi}}\, e^{-\frac{u^2}{2}}\, (u+\bar{\kappa})^2 \right)^{-1} \;. \tag{13}$$

At vanishing margin one obtains $\mathcal{C} = n$. For $n = 2$, a single readout, the capacity is hence identical to the mean perceptron [12]. Moreover, it only depends on the margin through the parameter $\bar{\kappa} \equiv \kappa/\sqrt{fc^2}$, which measures the margin κ relative to the standard deviation of the readout. This dependence on κ is identical for the mean perceptron, which was originally analyzed for $fc^2 = 1$.

The capacity is shown in Fig. 6F in comparison to the direct numerical optimization of the margin. Comparing the curves for different numbers m of inputs, the deviations between the theoretical prediction and numerical results is explained by finite size corrections —at weak loads, the larger network is closer to the analytical result. However, for the larger network the optimization does not converge at high memory loads, explaining the negative margin; pattern separation is incomplete in this regime.

The replica calculation exposes an intuitive explanation for the equivalence of both perceptrons. For the case $n = 2$ with two row vectors of B and a single label, the problem becomes isotropic in neuron space after the pattern average —cf. Eq. (46) in Appendix C. As an example, we assume a readout in an arbitrary direction determined by a row vector of B, say $\check{B}^{1\mathrm{T}} = (1, 0, \ldots, 0)^{\mathrm{T}}$. The readout element is given by $Q^0_{12} = \check{B}^{1\mathrm{T}}\, P^0\, \check{B}^2 = \sum_k \chi_{1k}\, \check{B}^2_k$, which is a simple linear readout of a binary random vector χ_{1k} —the same as with the mean perceptron.

The memory capacity only grows in proportion to n, again similar to n classical mean perceptrons (i.e. n outputs). Intuitively one could expect it to grow as $n(n-1)/2$, the number of classification readouts. It is easy to understand why it is the former: Consider three readout neurons —say i, j, and k— and their corresponding row vectors in B, namely $\check{B}^{i\mathrm{T}}$, $\check{B}^{j\mathrm{T}}$ and $\check{B}^{k\mathrm{T}}$. The covariance Q^0_{ij} provides a constraint on $\check{B}^i$. Likewise, the entry Q^0_{ik} provides a second constraint, potentially contradicting with the first. Stated differently, we have $n(n-1)/2$ independent constraints, but only mn weights in B. Therefore, there is a tradeoff between more readouts and more constraints for the weights.

Even though the pattern load p at a given margin is identical in the two perceptrons, the covariance perceptron has a higher information density. It is sufficient to compare the cases of a single readout in both cases. The mean perceptron stores the information [21]

$$I_{\mathrm{mean}} = m^2\, \mathcal{C}(\kappa) \;, \tag{14}$$

the number of bits required to express the $m\mathcal{C}$ patterns of m binary variables each. The covariance perceptron, on the other hand, stores

$$I_{\text{cov}} = \frac{fm^2(m-1)}{2}\,\mathcal{C}(\kappa/\sqrt{fc^2}) \tag{15}$$

bits. Although the calculations in Appendix C ignore the constraint that the covariance matrices $\hat{P}^r$ must be positive semidefinite, this constraint is ensured when using not too dense and strong entries such that $fc \ll 1$, thanks to the unit diagonal. Since $\sqrt{fc^2}$ only determines the scale on which the margin κ is measured, the optimal capacity can always be achieved if one allows for a sufficiently small margin. In a practical application, where covariances must be estimated from the data, this of course implies a longer observation time d to cope with the estimation error. Under this assumption, the expression for the information density of the covariance perceptron grows $\propto m^3$, while the former for the mean perceptron only grows with m^2. If one employs very sparse patterns such that $f \propto m^{-1}$ (an extreme condition), both perceptrons have comparable information content. The dependence on the number of readout neurons n is another linear factor in both cases.

4.3 Comparison of capacity via training accuracy for mean and covariance perceptrons

The analysis in the previous subsection exposed that the capacity of the covariance perceptron is comparable to that of the mean perceptron. To compare and complement the results in Section 3, we use the same optimization as in Figs. 2 and 3, but without additional noise on the presented patterns. We consider mean-based decoding and variance-based decoding for the network N1 with a single output node in Fig. 7A, as well as cross-covariance-based decoding for the network N2 with two output nodes.

Here we consider binarized outputs obtained using a threshold function θ, for example $\theta(Q^0_{12}) = 1$ for $Q^0_{12} > 0.5$ and 0 for $Q^0_{12} < 0.5$ for the cross-covariance in the network N2, as in the analytical calculation of the capacity. To incorporate this non-linearity in the gradient descent, we choose objectives $\bar{Q}^0_{12} \in \{0, 1\}$ and redefine the error E in Eq. (25) in Appendix B as $E^0 = \bar{Q}^0_{12} - \theta(Q^0_{12})$. It follows that $\frac{\partial E}{\partial Q_{12}}$ becomes a matrix full of zeros when the prediction is correct, whereas erroneous prediction corresponds to ± 1 for the output entries that determine the decision, with the sign depending on the category. We consider the same kind of patterns as with the analytical calculation, similar to the right matrix in Fig. 7B where off-diagonal elements are either 0 or $c = 1$ (we further check that the matrices are non-negative and required to be positive semidefinite). The evolution of the classification accuracy averaged over 50 configurations is displayed in Fig. 7C, where each color corresponds to a given number of input covariance patterns (lower accuracy for more patterns) For each configuration, the maximum accuracy is retained, in line with the offline learning procedure.

The same θ is applied to Q^0_{11} for variance-based decoding with the network N1. For mean-based decoding, we apply θ to X_1 and use binary input patterns (left vector in Fig. 7B), which corresponds to the classical perceptron. The comparison between the respective accuracies when increasing the total number p of patterns to learn ($p/2$ in each category) in Fig. 7D shows that the variance perceptron with the N1 network is on par with the mean perceptron. It also shows a clear advantage for the covariance perceptron, which is partly explained by the fact that the N2 network has twice as many afferent weights as the N1 network. The sparseness of the input patterns also affects the capacity that slightly increases for denser covariance matrices in Fig. 7E, as suggested by the theoretical results on information density. Last, Fig. 7F shows that tuning the mapping is robust when increasing the number m of inputs.

5 Online learning of simulated time series with hidden dynamics for both afferent and recurrent connectivities

We now come back to online learning with noisy time series and extend the results of Section 3 to the tuning of both afferent and recurrent connectivities in Eq. (1) with the same application to classification. From the dynamics described in Eq. (1), a natural use for A is the transformation of input spatial covariances ($P^0 \neq 0$ and $P^1 = 0$) to output spatio-temporal covariances ($Q^0 \neq 0$ and $Q^1 \neq 0$), or vice-versa ($P^0 \neq 0$, $P^1 \neq 0$, $Q^0 \neq 0$ and $Q^1 = 0$). The Appendices D.2 and D.3 provide examples for these two cases. As in

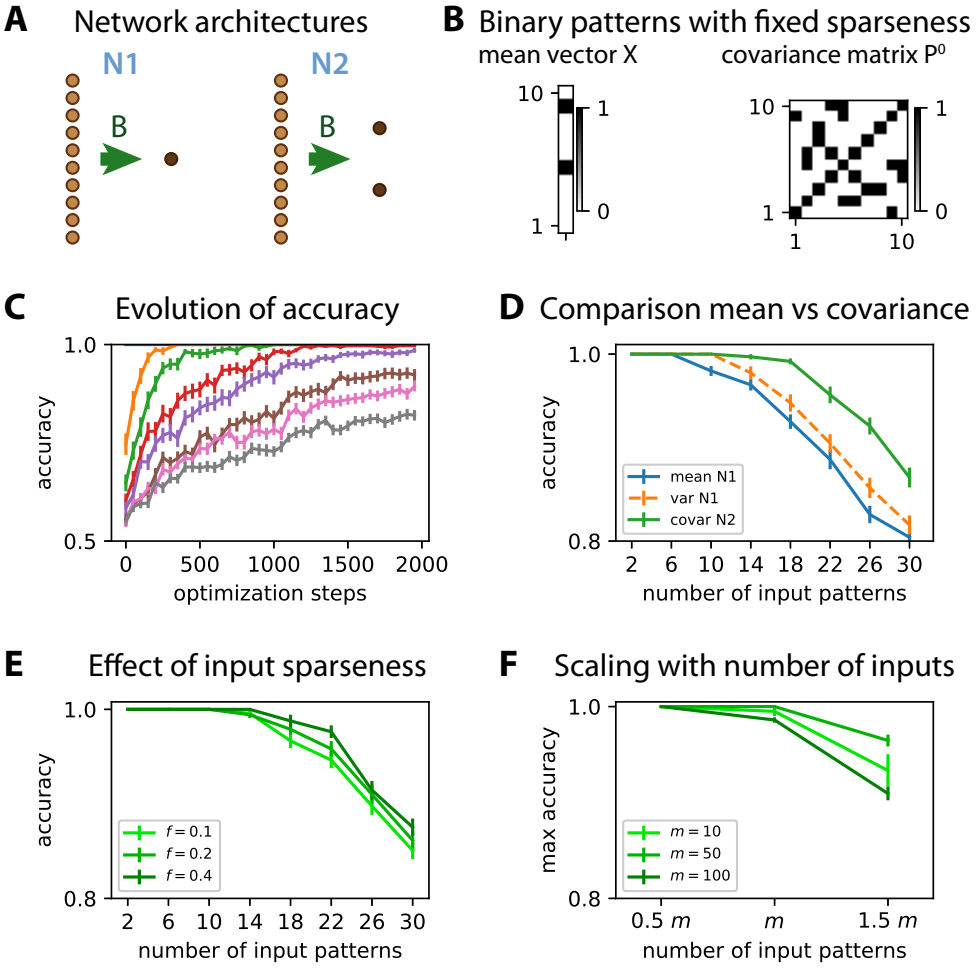

Figure 7: **Numerical evaluation of capacity using offline learning with non-linearity applied to readout. A** Feedforward networks with afferent connectivity as in Fig. 2A with $n = 1$ and $n = 2$ output nodes, referred to as N1 and N2. **B** Example mean vector X (left) and covariance matrix P^0 (right) whose density of non-zero elements is exactly $f = 20\%$. Note that the variances are all fixed to 2 while non-zero off-diagonal elements are set to $c = 1$. **C** Evolution of the classification accuracy over the noiseless patterns (each color correspond to a number of input patterns to learn). The variability corresponds to the standard error of the mean accuracy over 50 input and network configurations. **D** Comparison of the classification accuracies as a function of the number of patterns (x-axis). Covariance-based learning is tested in the N2 architecture using the cross-covariance (see panel B), while variance-based learning and mean-based learning are performed with N1. The plotted values are the maximum accuracy for each configuration, whose means are represented in panel C. **E** Similar plot to panel D for covariance-based learning when varying the sparseness of the input covariance matrices, as indicated by the density f in the legend. The error bars indicate the standard error of the mean accuracy over 50 repetitions. **F** Similar plot to panel E when varying the number m of inputs in the network (see legend). The number of patterns to learned are given as a fraction of m. The error bars correspond to 20 repetitions.

Fig. 2, we here do not simulate the time series, but instead rely on the consistency equations Eqs. (23) and (24), which are obtained in Appendix A under the assumption of stationary statistics. They demonstrate the ability to tune the recurrent connectivity together with the afferent connectivity, which we further examine now. To do so, we consider simulated time series that differ by their hidden dynamics. By "hidden dynamics" we simply mean that each time series obeys a dynamical equation, which determines its spatio-temporal structure that can be used for classification. Concretely, we use

$$x_k^t = \sum_l W_{kl} x_l^{t-1} + z_k^t \ , \tag{16}$$

with z_k^t being independent Gaussian random variables unit variance. This dynamical equation replaces the superposition of Gaussians in Eq. (7) for generating temporally correlated input signals. A class consists of a set of such processes, each with a different choice for the matrix W in Eq. (16), as shown in Fig. 8A. The matrix W itself is not known to the classifier, only the resulting statistics of x that obeys Eq. (16); thus we call this setting "classification of hidden dynamics". The key here is that P^1 conveys information, but not P^0. Our theory predicts that recurrent connectivity is necessary to extract the relevant information to separate the input patterns. To our knowledge this is the first study that tunes the recurrent connectivity in a supervised manner to specifically extract temporal information when spatial information is "absent".

Concretely, we here use 6 patterns for W (3 for each category) to generate the input time series that the network has to classify based on the output variances, illustrated in Fig. 8A. Importantly, we choose $W = \exp(\mu \mathbf{1}_m + V)$ with exp being the matrix exponential and V an antisymmetric matrix and $\mu < 0$ for stability. As a result, the zero-lag covariance of the input signals $P^0 = \frac{1}{1-e^{2\mu}} \mathbf{1}_m$ is the same for all patterns of either category, proportional to the identity matrix as illustrated in Fig. 8B. This can be seen using the discrete Lyapunov equation $P^0 = W P^0 W^{\mathrm{T}} + \mathbf{1}_m$, which is statisfied because $WW^T = \exp(2\mu \mathbf{1}_m + V + V^T) = e^{2\mu} \mathbf{1}_m$. As mentioned earlier, the time-lagged covariances $P^1 = W P^0$ differ across patterns, which is the basis for distinguishing the two categories. The derivation of the consistency equations in Appendix A assumes $P^2 = 0$ and is thus an approximation because we have $P^2 = W^2 P^0$ here. As the input matrix W must have eigenvalues smaller than 1 in modulus to ensure stable dynamics, our approximation corresponds to $||P^2|| = ||WP^1|| < ||P^1||$.

The output is trained only using Q^0, meaning that the input spatio-temporal structure is mapped to an output spatial structure. Simplifying Eq. (71) for the current configuration, the weight updates are given by

$$\begin{aligned} \Delta A_{ij} &= \eta_A \left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial A_{ij}} \ , \\ \Delta B_{ik} &= \eta_B \left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial B_{ik}} \ , \end{aligned} \tag{17}$$

where the derivatives are given by the matrix versions of Eqs. (30) and (32) in Appendix B:

$$\begin{aligned} \frac{\partial Q^0}{\partial A_{ij}} &= A \frac{\partial Q^0}{\partial A_{ij}} A^{\mathrm{T}} + V^{ij} Q^0 A^{\mathrm{T}} + A Q^0 V^{ij\mathrm{T}} + V^{ij} B P^{-1} B^{\mathrm{T}} + B P^{-1\mathrm{T}} B^{\mathrm{T}} V^{ij\mathrm{T}} \ , \\ \frac{\partial Q^0}{\partial B_{ik}} &= A \frac{\partial Q^0}{\partial B_{ik}} A^{\mathrm{T}} + U^{ik} P^0 B^{\mathrm{T}} + B P^0 U^{ik\mathrm{T}} + A U^{ik} P^{-1} B^{\mathrm{T}} + A B P^{-1} U^{ik\mathrm{T}} \\ &\quad + U^{ik} P^{-1\mathrm{T}} B^{\mathrm{T}} A^{\mathrm{T}} + B P^{-1\mathrm{T}} U^{ik\mathrm{T}} A^{\mathrm{T}} \ . \end{aligned} \tag{18}$$

Both formulas have the form of a discrete Lyapunov equation that can be solved at each optimization step to evaluate the weight updates for A and B. The non-linearity due to the recurrent connectivity A thus plays an important role in determining the weight updates. As Eq. (18) involves the approximation of ignoring P^2, the purpose of the following is to test the robustness of the proposed learning in a practical use.

The covariances from the time series are computed using an observation window of duration d represented in Fig. 8B, in the same manner as before. We use a larger window duration d compared to Fig. 4 because the output covariances are much noisier here due to the approximation mentioned above. The influence of d can also be seen in Fig. 8D, where the evolution of the error for the darkest curves with $d \geq 60$ remain lower on average than the lighter curve with $d = 20$. To assess the quality of the

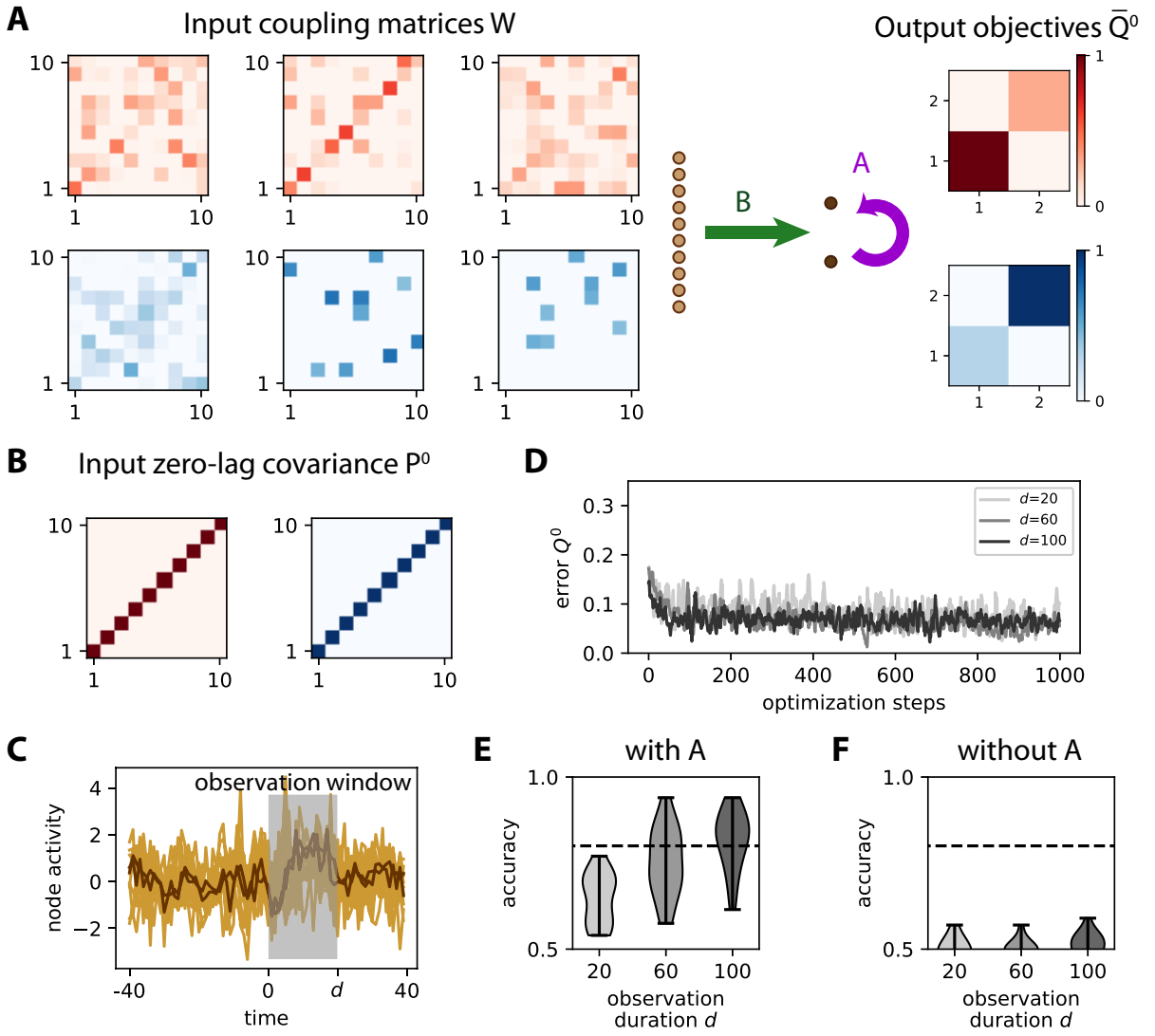

Figure 8: **Online learning for input spatio-temporal covariances with both afferent and recurrent connectivities. A** Same network as in Fig. S3A to learn the input spatio-temporal covariance structure, which is determined here by a coupling matrix W between the inputs as in Eq. (16). Here we have 3 input patterns per category. The objective matrices (right) correspond to a specific variance for the output nodes. **B** The matrices W are constructed such that they all obey the constraint $P^0 = \mathbf{1}_m$. **C** Example of simulation of the time series for the inputs (light brown traces) and outputs (dark brown). An observation window (gray area) is used to calculate the covariances from simulated time series. **D** Evolution of the error for 3 example optimizations with various observation durations d as indicated in the legend. **E** Classification accuracy after training similar to Fig. S3D averaged over 20 network and input configurations. For the largest $d = 100$, the accuracy is above 80% on average (dashed line). The color contrast corresponds to the three values for d as in panel D. **F** Accuracy similar to E when switching off the learning for the recurrent connectivity A.

training, we repeat the simulations for 20 network and input configurations, then calculate the difference in variance between the two output nodes for the red and blue input patterns. The accuracy gradually improves from $d = 20$ to 100 in Fig. 8E. When switching off the learning of A in Fig. 8F, classification stays at chance level. This is expected and confirms our theory, because the learning for B only captures differences in P^0, which is the same for all patterns here. These results demonstrate the importance of recurrent connections in transforming input spatiotemporal covariances into output spatial covariances.

6 Discussion

This paper presents a new learning theory for the categorization of time series. We derive learning rules to train both afferent and recurrent connections of a linear network model in a supervised manner. The proposed method extracts regularities in the spatio-temporal fluctuations of input time series, as quantified by their covariances. Networks can be trained to map several input time series to a stereotypical output time series that represents the respective class, thus implementing a 'covariance perceptron' as shown here for two categories of output covariance patterns.

A main result is that the covariance perceptron can be trained in an online manner to robustly classify time series with various covariance patterns, while observing a few time points only (Fig. 4). Intuitively, this robustness results from the representation of the information by the covariance within a higher-dimensional space compared to the mean, which is employed by classical architectures. The new architecture therefore can make more efficient use of the resources, neurons and synapses, as formally shown by assessing its capacity; the information density is orders of magnitude larger than that of the mean perceptron: It exceeds the mean perceptron by a factor equal to the number of input neurons even though the number of classifiable patterns is theoretically the same as for the classical perceptron (Fig. 6). In simulations akin to offline learning, the resulting accuracy of the covariance perceptron compares favorably with the mean perceptron (Fig. 7). The other main result is the demonstration that the covariance perceptron can classify time series with respect to their hidden dynamics, based on temporal information only (Fig. 8). In other words, the goal here is to distinguish the statistical dependencies in signals that obey different dynamical equations. We stress the importance of the results for online learning: Cross-validation is here performed by taking into account the variability inherent to the time series. This contrasts with the assessment of the capacity that relies on noiseless patterns (Fig. 7).

The conceptual change of perspective compared to previous studies is that variability in the time series is here the basis for the information to be learned, namely the second-order statistics of the co-fluctuating inputs. This view, which is based on dynamical features, thus contrasts with classical and more "static" approaches that consider the variability as noise, potentially compromising the information conveyed in the mean activity of the time series. Importantly, covariance patterns can involve time lags and are a genuine metric for time series, describing the transfer of activity between nodes. This paradigm opens the door to a self-consistent formulation of information processing in recurrent networks: The source of the signal and the classifier both have the same structure of a recurrent network.

6.1 Covariance-based decoding and representations

The mechanism underlying classification is the linear separability of the input covariance patterns performed by a threshold on the output activity, in the same manner as in the classical perceptron for vectors. The perceptron is a central concept for classification based on artificial neuronal networks, from logistic regression [8] to deep learning [24, 39]. The entire output covariance matrix Q^0 can be used as the target quantity to be trained, cross-covariances as well as variances. In Section 4 the non-linearity on the readout used for classification has been included in the gradient descent. It remains to be explored which types of non-linearities improve the classification performance —as is well known for the perceptron [28]— or lead to interesting input-output covariance mappings. Nonetheless, our results lay the foundation for covariance perceptrons with multiple layers, including linear feedback by recurrent connectivity in each layer. The important feature in its design is the consistency of covariance-based information from inputs to outputs.

Although our study is not the first one to train the recurrent connectivity in a supervised manner, our approach differs from previous extensions of the delta rule [28] or the back-propagation algorithm [38], such as recurrent back-propagation [34] and back-propagation through time [33]. Those algorithms focus

on the mean activity (or trajectories over time, based on first-order statistics) and, even though they do take temporal information into account (related to the successive time points in the trajectories), they consider the inputs as statistically independent variables. Moreover, unfolding time corresponds to the adaptation of techniques for feedforward networks to recurrent networks, but it does not take the effect of the recurrent connectivity as in the steady-state dynamics considered here. In the context of unsupervised learning, several rules were proposed to extract information from the spatial correlations of inputs [31] or their temporal variance [4]. Because our training scheme is based on the same input properties, we expect that the strengths exhibited by those learning rules also partly apply to our setting, for example the robustness for the detection of time-warped patterns as studied in [4].

The reduction of dimensionality of covariance patterns —from many input nodes to a few output nodes— implements an "information compression". For the same number of input-output nodes in the network, the use of covariances instead of means makes a higher-dimensional space accessible to represent input and output, which may help in finding a suitable projection for a classification problem. It is worth noting that applying a classical machine-learning algorithm, like the multinomial linear regressor [8], to the vectorized covariance matrices corresponds to $nm(m-1)/2$ weights to tune, to be compared with only nm weights in our study (with m inputs and n outputs). The presented theoretical calculations focus on the capacity of the covariance perceptron for perfect classification (Fig. 6). It uses Gardner's replica method [15] in the thermodynamic limit, toward infinitely many inputs ($m \to \infty$). We have shown that our paradigm indeed presents an analytically solvable model in this limit and compute the pattern capacity $\mathcal{C} = p/m$ by replica symmetric mean-field theory, analogous to the mean perceptron [15]. It turns out that the pattern capacity (per input and output) is exactly identical to that of the mean perceptron. Its information capacity in bits, however, grows with m^3, whereas it only has a dependence as m^2 for the mean perceptron. The proposed paradigm in large networks therefore reaches an information density that is orders of magnitude higher than that of the mean perceptron.

Both the pattern capacity and information capacity linearly depend on the size of the target population n. The latter result is trivial in the case of the mean perceptron —one simply has n independent perceptrons in that case. However, it is non-trivial in the case of the covariance perceptron, because different entries Q^0_{ij} here share the same rows of the matrix B. These partly confounding constraints reduce the capacity from the naively expected dependence on $n(n-1)/2$, the number of independent off-diagonal elements of Q^0, to n.

Future work should extend the theory of the capacity for noiseless patterns (Fig. 6) to take into account the observation noise, which is inherent to time series, as well as to the here-considered network models. For such noisy patterns, it appears relevant to evaluate the capacity in the "error regime" [9], in which classification is not perfect; our numerical results correspond to this regime (Fig. 7).

6.2 Learning and functional role for recurrent connectivity

Our theory shows that recurrent connections are crucial to transform information contained in time-lagged covariances into covariances without time lag (Fig. 8). Simulations confirm that recurrent connections can indeed be learned successfully to perform robust binary classification in this setting. The corresponding learning equations clearly expose the necessity of training the recurrent connections. For objectives involving both covariance matrices, $\bar{Q}^0$ and $\bar{Q}^1$, there must exist an accessible mapping $(P^0, P^1) \mapsto (Q^0, Q^1)$ determined by A and B. The use for A may also bring an extra flexibility that broadens the domain of solutions or improve the stability of learning, even though this was not clearly observed so far in our simulations.

On a more technical ground, a positive feature of our learning scheme is the surprising stability of the recurrent weights A for ongoing learning (see Appendix D.1). Many previous studies use regularization terms, in biology known as "homeostasis", to prevent the problematic growth of recurrent weights that often leads to an explosion of the network activity [42, 43]. It remains to be explored in more depth why such regularizations are not needed in the current framework.

The learning equations for A in Appendix B can be seen as an extension of the optimization for recurrent connectivity recently proposed [18] for the multivariate Ornstein-Uhlenbeck (MOU) process, which is the continuous-time version of the MAR studied here. Such learning update rules fall in the group of natural gradient descents [1] as they take into account the non-linearity between the weights and the output covariances. A natural gradient descent was used before to train afferent and recurrent

connectivity to decorrelate signals and perform blind-source separation [10]. This suggests as another possible role for A the global organization of output nodes; for example, forming communities of output nodes that are independent of each other (irrespective of the patterns).

6.3 Extensions to continuous time and non-linear dynamics

The MAR network dynamics in discrete time used here leads to a simple description for the propagation of temporally-correlated activity. Extension of the learning equations to continuous time MOU processes requires the derivation of consistency equations for the time-lagged covariances. The inputs to the process, for consistency, themselves need to have the statistics of a MOU process [5]. This is doable, but yields more complicated expressions than for the MAR process.

To take into account several types of non-linearities that arise in recurrently connected networks, one can also envisage the following generalization of the network dynamics

$$\mathrm{d}x_i^t = \phi(x_i^t) + \psi\left(\sum_j C_{ij} x_j^t\right) + \mathrm{d}W_i^t \ . \tag{19}$$

Here the local dynamics is determined by ϕ and interactions are rectified by the function ψ. Such non-linearities are expected to vastly affect the covariance mapping in general, but special cases, like the rectified linear function, preserve the validity of the derivation for the linear system in Appendix A in a range of parameters. The present formalism may thus be extended beyond the non-linearity applied to the readout (Section 4). Note that for the mean perceptron a non-linearity applied the dynamics is in fact the same as applied to the output; this is, however, not so for the covariance perceptron.

Another point is that non-linearities cause a cross-talk between statistical orders, meaning that the mean of the input may strongly affect output covariances and, conversely, input covariances may affect the mean of the output. This opens the way to mixed decoding paradigms where the relevant information is distributed in both, means and covariances.

6.4 Learning and (de)coding in biological spiking neuronal networks

An interesting application for the present theory is its adaptation to spiking neuronal networks. In fact, the biologically-inspired model of spike-timing-dependent plasticity (STDP) can be expressed in terms of covariances between spike trains [22, 16], which was an inspiration of the present study. STDP-like learning rules were used for object recognition [23] and related to the expectation-maximization algorithm [30]. Although genuine STDP relates to unsupervised learning, extensions were developed to implement supervised learning for spike patterns [20, 35, 19, 14, 41]. A common trait of those approaches is that learning mechanisms are derived for feedforward connectivity only, even though they have been used and tested in recurrently-connected networks. Instead of focusing on the detailed timing in spike trains in output, our supervised approach could be transposed to shape the input-output mapping between spike-time covariances, which are an intermediate description between spike patterns and firing rate. As such, it allows for some flexibility concerning the spike timing (e.g. jittering) and characterization of input-output patterns, as was explored before for STDP [17]. An important property for covariance-based patterns is that they do not require a reference start time, because the coding is embedded in relative time lags. Our theory thus opens a promising perspective to learn temporal structure of spike trains and provides a theoretical ground to genuinely investigate learning in recurrently connected neuronal networks. A key question is whether the covariance estimation in our method can be robustly implemented in an online fashion. Another important question concerns the locality of the learning rule, which requires pairwise information about neuronal activity.

Here we have used arbitrary covariances for the definition of input patterns, but they could be made closer to examples observed in spiking data, as was proposed earlier for probabilistic representation of the environment [6]. It is important noting that the observed activity structure in data (i.e. covariances) can not only be related to neuronal representations, but also to computations that can be learned (here classification). Studies of noise correlation, which is akin to the variability of spike counts (i.e. mean firing activity), showed that variability is not always a hindrance for decoding [3, 29]. Our study instead makes active use of activity variability and is in line with recent results about stimulus-dependent correlations

observed in data [36]. It thus moves variability into a central position in the quest to understand biological neuronal information processing.

7 Acknowledgements

MG acknowledges funding from the Marie Sklodowska-Curie Action (GrantH2020-MSCA-656547) of the European Commission. MG was also supported by the European Unions Horizon 2020 research and innovation programme under Grant Agreement No. 785907 (HBP615SGA2).

This work was partially supported by Helmholtz young investigator's group VH-NG-1028, the European Union's Horizon 2020 research and innovation programme under grant agreement No. 720270 (Human Brain Project SGA2), the Exploratory Research Space (ERS) seed fund neuroIC002 (part of the DFG excellence initiative) of the RWTH university and the JARA Center for Doctoral studies within the graduate School for Simulation and Data Science (SSD).

A Network dynamics describing activity propagation

Here we recapitulate well-known calculations [25] that describe the statistics of the activity in discrete time in a MAR process in Eq. (1), which we recall here:

$$y_i^t = \sum_j A_{ij} y_j^{t-1} + \sum_k B_{ik} x_k^t \ . \tag{20}$$

Our focus are the self-consistency equations when the multivariate outputs y_i^t are driven by the multivariate inputs x_k^t, whose activity is characterized by the 0-lag covariances P^0 and 1-lag covariances $P^1 = (P^{-1})^{\mathrm{T}}$, where T denotes the matrix transpose. We assume stationary dynamics (over the observation period) and require that the recurrent connectivity matrix A has eigenvalues in the unit circle (modulus strictly smaller than 1) to ensure stability. To keep the calculations as simple as possible, we make the additional hypothesis that $P^{\pm n} = 0$ for $n \geq 2$, meaning that the memory of x_k^t only concerns one time lag. Therefore, the following calculations are only an approximations of the general case for x_k^t, which is discussed in the main text about Fig. 8. Note that this approximation is reasonable when the lagged covariances P^n decrease exponentially with the time lag n, as is the case when inputs are a MAR process.

Under those conditions, we define $R_{ik}^\tau = \langle y_i^{t+\tau} x_k^t \rangle$ and express these matrices in terms of the inputs as a preliminary step. They obey

$$R^\tau = AR^{\tau-1} + BP^\tau \ . \tag{21}$$

Because we assume $P^{\pm n} = 0$ for $n \geq 2$, we have the following expressions

$$\begin{aligned} R^{-n} &= 0 \quad \text{for } n \geq 2 \ , \\ R^{-1} &= BP^{-1} \ , \\ R^0 &= ABP^{-1} + BP^0 \ . \end{aligned} \tag{22}$$

Using the expression for R, we see that the general expression for the zero-lagged covariance of y_i^t depends on both zero-lagged and lagged covariances of x_k^t:

$$\begin{aligned} Q^0 &= AQ^0A^{\mathrm{T}} + BP^0B^{\mathrm{T}} + AR^{-1}B^{\mathrm{T}} + BR^{-1\mathrm{T}}A^{\mathrm{T}} \\ &= AQ^0A^{\mathrm{T}} + BP^0B^{\mathrm{T}} + ABP^{-1}B^{\mathrm{T}} + BP^{-1\mathrm{T}}B^{\mathrm{T}}A^{\mathrm{T}} \ . \end{aligned} \tag{23}$$

The usual (or simplest) Lyapunov equation in discrete time corresponds to $P^1 = 0$ and the afferent connectivity matrix B being the identity with $n = m$ independent inputs that are each sent to a single output. Likewise, we obtain the lagged covariance for y_i^t:

$$\begin{aligned} Q^1 &= AQ^1A^{\mathrm{T}} + BP^1B^{\mathrm{T}} + AR^0B^{\mathrm{T}} + BR^{-2\mathrm{T}}A^{\mathrm{T}} \\ &= AQ^1A^{\mathrm{T}} + BP^1B^{\mathrm{T}} + ABP^0B^{\mathrm{T}} + AABP^{-1}B^{\mathrm{T}} \ . \end{aligned} \tag{24}$$

Note that the latter equation is not symmetric because of our assumption of ignoring $P^{\pm n} = 0$ for $n \geq 2$.

B Theory for weight updates

We now look into the gradient descent to reduce the error E^τ, defined for $\tau \in \{0, 1\}$, between the network covariance Q^τ and the desired covariance $\bar{Q}^\tau$, which we take here as the matrix distance:

$$E^\tau = \frac{1}{2}||Q^\tau - \bar{Q}^\tau||_2 \equiv \frac{1}{2}\sum_{i_1,i_2}(Q^\tau_{i_1 i_2} - \bar{Q}^\tau_{i_1 i_2})^2 \ . \tag{25}$$

The following calculations assume the tuning of B or A, or both.

Starting with afferent weights, the derivation of their updates ΔB_{ik} to reduce the error E^τ at each optimization step is based on the usual chain rule, here adapted to the case of covariances:

$$\Delta B_{ik} = -\eta_B \frac{\partial E^\tau}{\partial B_{ik}} = -\eta_B \sum_{i_1,i_2} \frac{\partial E^\tau}{\partial Q^\tau_{i_1 i_2}} \frac{\partial Q^\tau_{i_1 i_2}}{\partial B_{ik}} = -\eta_B \frac{\partial E^\tau}{\partial Q^\tau} \odot \frac{\partial Q^\tau}{\partial B_{ik}} \ , \tag{26}$$

where η_B is the learning rate for the afferent connectivity and the symbol $\odot$ defined in Eq. (6) corresponds to the sum after the element-wise product of the two matrices. Note that we use distinct indices for B and Q^τ. Once again, this expression implies the sum over all indices (i', j') of the covariance matrix Q^τ. The first terms $\frac{\partial E^\tau}{\partial Q^\tau_{i_1 i_2}}$ can be seen as an $n \times n$ matrix with indices (i_1, i_2):

$$\frac{\partial E^\tau}{\partial Q^\tau} = Q^\tau - \bar{Q}^\tau \ . \tag{27}$$

The second terms in Eq. (26) correspond to a tensor with 4 indices, but we now show that it can be obtained from the above consistency equations in a compact manner. Fixing j and k and using Eq. (23), the "derivative" of Q^0 with respect to B can be expressed as

$$\begin{aligned} \frac{\partial Q^0}{\partial B_{ik}} \ = \ & A\frac{\partial Q^0}{\partial B_{ik}}A + \frac{\partial B}{\partial B_{ik}}P^0 B^{\mathrm{T}} + BP^0\frac{\partial B}{\partial B_{ik}}^{\mathrm{T}} + A\frac{\partial B}{\partial B_{ik}}P^{-1}B^{\mathrm{T}} + ABP^{-1}\frac{\partial B}{\partial B_{ik}}^{\mathrm{T}} \\ & + \frac{\partial B}{\partial B_{ik}}P^{-1\mathrm{T}}B^{\mathrm{T}}A^{\mathrm{T}} + BP^{-1\mathrm{T}}\frac{\partial B}{\partial B_{ik}}^{\mathrm{T}}A^{\mathrm{T}} \ . \end{aligned} \tag{28}$$

Note that the first term on the right-hand side of Eq. (23) does not involve B, so it vanishes. Each of the other terms in Eq. (23) involves B twice, so they each give two terms in the above expression —as when deriving a product. The trick lies in seeing that

$$\frac{\partial B_{i'k'}}{\partial B_{ik}} = \delta_{i'i}\delta_{k'k} \tag{29}$$

where δ denotes the Kronecker delta. In this way we can rewrite the above expression using the basis $n \times m$ matrices U^{ik} that have 0 everywhere except for element (i, k) that is equal to 1. It follows that the n^2 tensor element for each (i, k) can be obtained by solving the following equation:

$$\begin{aligned} \frac{\partial Q^0}{\partial B_{ik}} \ = \ & A\frac{\partial Q^0}{\partial B_{ik}}A + U^{ik}P^0B^{\mathrm{T}} + BP^0U^{ik\mathrm{T}} + AU^{ik}P^{-1}B^{\mathrm{T}} + ABP^{-1}U^{ik\mathrm{T}} \\ & + U^{ik}P^{-1\mathrm{T}}B^{\mathrm{T}}A^{\mathrm{T}} + BP^{-1\mathrm{T}}U^{ik\mathrm{T}}A^{\mathrm{T}} \ , \end{aligned} \tag{30}$$

which has the form of a discrete Lyapunov equation:

$$X = AXA^{\mathrm{T}} + \Sigma \tag{31}$$

with the solution $X = \frac{\partial Q^0}{\partial A_{ij}}$ and Σ being the sum of 6 terms involving matrix multiplications. The last step to obtain the desired update for ΔB_{ik} in Eq. (26) is to multiply the two $n \times n$ matrices in Eqs. (30) and (27) *element-by-element* and sum over all pairs (i_1, i_2) —or alternatively vectorize the two matrices and calculate the scalar product of the two resulting vectors.

Now turning to the case of the recurrent weights, we use the same general procedure as above: We simply substitute each occurrence of A in the consistency equations by a basis matrix, once at a time in the case of matrix products. The "derivation" of Q^0 in Eq. (23) with respect to A gives

$$\frac{\partial Q^0}{\partial A_{ij}} = A\frac{\partial Q^0}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^0A^{\mathrm{T}} + AQ^0V^{ij\mathrm{T}} + V^{ij}BP^{-1}B^{\mathrm{T}} + BP^{-1\mathrm{T}}B^{\mathrm{T}}V^{ij\mathrm{T}} , \tag{32}$$

where V^{ij} is the basis $n \times n$ matrix with 0 everywhere except for (i, j) that is equal to 1. This has the same form as Eq. (31) and, once the solution for the discrete Lyapunov equation is calculated for each pair (i, j), the same element-wise matrix multiplication can be made with Eq. (27) to obtain the weight update ΔA_{ij}.

Likewise, we compute from Eq. (24) the following expressions to reduce the error related to Q^1:

$$\begin{aligned}\frac{\partial Q^1}{\partial B_{ik}} = {} & A\frac{\partial Q^1}{\partial B_{ik}}A + U^{ik}P^1B^{\mathrm{T}} + BP^1U^{ik\mathrm{T}} + AU^{ik}P^0_{k_1k_2}B^{\mathrm{T}} + ABP^0U^{ik\mathrm{T}} \\ & + AAU^{ik}P^{-1}B^{\mathrm{T}} + AABP^{-1}U^{ik\mathrm{T}} ,\end{aligned} \tag{33}$$

and

$$\frac{\partial Q^1}{\partial A_{ij}} = A\frac{\partial Q^1}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^1A^{\mathrm{T}} + AQ^1V^{ij\mathrm{T}} + V^{ij}BP^0B^{\mathrm{T}} + V^{ij}ABP^{-1}B^{\mathrm{T}} + AV^{ij}BP^{-1}B^{\mathrm{T}} . \tag{34}$$

$$\tag{35}$$

These expressions are also discrete Lyapunov equations and can be solved as explained before.

C Theory for the memory capacity of the covariance perceptron with feedforward connectivity

We consider the mapping $P^0 \mapsto Q^0$ in Eq. (5) with $P^0 \in \mathbb{R}^{m\times m}$ and $y \in \mathbb{R}^{n\times n}$, ignoring the recurrent connectivity in Eq. (20). We want to discriminate p input covariance patterns P^0, which we denote by $\hat{P}^r$ with $1 \le r \le p$ and assume to be of the form:

$$\hat{P}^r = 1_m + \chi^r , \tag{36}$$

where 1_m is the $m \times m$ identity matrix and each symmetric matrix $\chi^r = \chi^{\mathrm{T}}$ is drawn randomly with

$$\begin{aligned}\chi^r_{kk} &= 0 \\ \chi^r_{k<l} &\equiv \begin{cases} c & \text{with prob. } \frac{1}{2}f \\ -c & \text{with prob. } \frac{1}{2}f \\ 0 & \text{with prob. } 1-f \end{cases} .\end{aligned} \tag{37}$$

Here we ignore further requirements about $\hat{P}^r$ being positive semidefinite, which is discussed in the main text. For each input covariance $P^0 = \hat{P}^r$, the corresponding output covariance Q^0 for the mapping in Eq. (5) is

$$\hat{Q}^r = B\hat{P}^rB^{\mathrm{T}} . \tag{38}$$

We want to investigate the maximum number p of patterns $\hat{P}^r$ that can be discriminated using $\hat{Q}^r$ in a given network defined by the afferent connectivity B. For the readout covariance matrix $\hat{Q}^r$, we demand the following $n(n-1)/2$ constraints for all $1 \le i < j \le n$:

$$\begin{aligned}\zeta^r_{i<j}\,\hat{Q}^r_{i<j} &> \kappa , \\ \zeta^r_{i<j} &= \begin{cases} 1 & \text{with prob. } \frac{1}{2} \\ -1 & \text{with prob. } \frac{1}{2} \end{cases} ,\end{aligned} \tag{39}$$

with each matrix ζ^r being symmetric as is $\hat{Q}^r$. The parameter κ plays the role of a classification margin. These conditions require each element of the output covariance matrix to be away from zero by κ, on the side determined by the sign of ζ^r_{ij}.

First considering the diagonal elements of $\hat{Q}^r$, we see that Eq. (38) implies

$$\hat{Q}^r_{ii} = \sum_k B^2_{ik} + \sum_{k\neq l} B_{ik}B_{il}\chi^r_{kl} \qquad \forall\, i, r\ . \tag{40}$$

The distribution of the patterns defined by Eq. (37) implies $\langle \chi^r_{kl}\rangle = 0$, so the expected value of output variances $\hat{Q}^r_{ii}$ implies a normalization for each row vector of B, which we assume to be equal to 1:

$$1 \stackrel{!}{=} \langle Q^r_{ii}\rangle_\chi = \sum_k B^2_{ik} \qquad \forall\, i\ . \tag{41}$$

This gives another constraint for B, in addition to Eq. (39).

C.1 Gardner's approach to memory capacity

We now define the volume of solutions B whose p mappings in Eq. (38) satisfy the inequalities and the statistics of the output covariance in Eq. (39):

$$\mathcal{V} = \int \mathrm{d}B \prod_i \delta\Big(\sum_k B^2_{ik} - 1\Big) \prod_{r=1}^{p} \prod_{i<j} \theta\Big(\zeta^r_{ij}\left(B\hat{P}^r B^{\mathrm{T}}\right)_{ij} - \kappa\Big)\ . \tag{42}$$

This equation is the analogue to Gardner's approach of the perceptron; see [21, Section 10.2, eq. 10.83].

We are interested in the average behavior of $\mathcal{V}$ in the limit of large m and therefore consider $\langle \ln(\mathcal{V})\rangle$ using the replica trick $\ln(\mathcal{V}) = \lim_{q\to 0} \frac{Z^q - 1}{q}$. It leads to the study of the pattern average of the following expression in the limit $q \to 0$:

$$\mathcal{V}^q = \left\langle \prod_{\alpha=1}^{q} \int \mathrm{d}B^\alpha \prod_i \delta\Big(\sum_k B^{\alpha 2}_{ik} - 1\Big) \prod_{r=1}^{p} \prod_{i<j} \theta\Big(\zeta^r_{ij}\left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - \kappa\Big)\right\rangle_{\zeta,\chi}\ . \tag{43}$$

Therefore, we consider q such systems that have identical realizations of patterns. If there are many solutions to the set of equations, the average overlap between different systems will be small. In case there is only a single solution, the overlap will be unity.

C.2 Pattern average

We then perform the average over the distribution of patterns that obey Eqs. (37) and (39). We rewrite the Heaviside function as

$$\theta\Big(\zeta^r_{ij}\left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - \kappa\Big) = \int_\kappa^\infty \mathrm{d}x^\alpha_{ij} \int_{-\infty}^{\infty} \frac{\mathrm{d}\tilde{x}^\alpha_{ij}}{2\pi}\, e^{\imath\, \tilde{x}^\alpha_{ij}\left(\zeta^r_{ij}\left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - x^\alpha_{ij}\right)} \tag{44}$$

such that the pattern average

$$\left\langle \prod_{\alpha=1}^{q} \prod_{r=1}^{p} \prod_{i<j} \theta\Big(\zeta^r_{ij}\left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - \kappa\Big)\right\rangle_{\zeta,\chi} = \prod_{r=1}^{p} \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^{\infty} \mathrm{D}\tilde{x}\, e^{\Phi^r(\imath\tilde{x}) - \imath \sum_{\alpha,i,j} \tilde{x}^\alpha_{ij} x^\alpha_{ij}} \tag{45}$$

can be described by a cumulant generating function Φ^r of the variable $Q^{r\alpha}_{ij} \equiv \zeta^r_{ij}(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}})_{ij}$ with respect to the statistics of ζ and χ. The constant $\imath$ is the imaginary unit. Here we have defined the abbreviations $\int_\kappa^\infty Dx \equiv \prod_\alpha \prod_{i<j} \int_\kappa^\infty \mathrm{d}x^\alpha_{ij}$ and $\int_{-\infty}^{\infty} \mathrm{D}\tilde{x} \equiv \prod_\alpha \prod_{i<j} \int_{-\infty}^{\infty} \frac{\mathrm{d}\tilde{x}^\alpha_{ij}}{2\pi}$ and used that the p patterns are statistically independent. The function Φ^r can be expanded in cumulants and, in the large-m limit,

this expansion can be truncated at the second order in a similar fashion to the mean perceptron [21]. As a result, we obtain

$$\begin{aligned}\langle \mathcal{V}^q \rangle_{\zeta,\chi} &= \prod_{\alpha=1}^{q} \int \mathrm{d}B^\alpha \prod_i \delta\Big(\sum_k B_{ik}^{\alpha 2} - 1\Big) \\ &\times \prod_{r=1}^{p} \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^{\infty} \mathrm{D}\tilde{x} \prod_{i<j} e^{-\imath \sum_\alpha \tilde{x}_{ij}^\alpha x_{ij}'^\alpha - \frac{1}{2}\sum_{\alpha,\beta} \tilde{x}_{ij}^\alpha \left\langle\left\langle Q_{ij}^{r\alpha} Q_{ij}^{r\beta}\right\rangle\right\rangle \tilde{x}_{ij}^\beta}\end{aligned} \tag{46}$$

with

$$\begin{aligned}\left\langle\left\langle Q_{ij}^{r\alpha} Q_{ij}^{r\beta}\right\rangle\right\rangle &= \Big(\sum_k B_{ik}^\alpha B_{jk}^\alpha\Big)\Big(\sum_k B_{ik}^\beta B_{jk}^\beta\Big) \\ &+ fc^2 \Big(\sum_k B_{ik}^\alpha B_{ik}^\beta\Big)\Big(\sum_k B_{jk}^\alpha B_{jk}^\beta\Big) \\ &+ fc^2 \Big(\sum_k B_{ik}^\alpha B_{jk}^\beta\Big)\Big(\sum_k B_{ik}^\beta B_{jk}^\alpha\Big) \,.\end{aligned} \tag{47}$$

In the second and third lines we added a single term $k = l$ which is negligible in the large-m limit. We see that the only dependence on the sparseness f and the magnitude c of input covariances is in the form fc^2 —it does not depend on these two parameters separately. The problem is, moreover, now symmetric in all $i < j$ index pairs. We also observe that the bracket that is multiplied p times does not depend on the pattern index r, so that we only get the bracket to the p-th power.

C.3 Auxiliary field formulation

Starting from Eq. (46), we now define the auxiliary fields as

$$R_{ij}^{\alpha\beta} \equiv \sum_k B_{ik}^\alpha B_{jk}^\beta \,, \tag{48}$$

for $i < j$ and $\alpha \neq \beta$. For $\alpha = \beta$ and $i = j$ we have $R_{ii}^{\alpha\alpha} = 1$ due to Eq. (41). The field $R_{ij}^{\alpha\alpha}$ for $i \neq j$ measures the overlap between input vectors to different units. It contributes to the average value of $Q_{ij}^{r\alpha}$ because the unit diagonal (common to all $\hat{P}^r$) is weighted by $R_{ij}^{\alpha\alpha}$. Hence the output $Q_{ij}^{r\alpha}$ will be displaced by $R_{ij}^{\alpha\alpha}$ irrespective of the realization of $\hat{P}^r$. $R_{ij}^{\alpha\beta}$ for $\alpha \neq \beta$ measures the overlap of input vectors in different systems. We denote by $\check{B}^{i\alpha}$ the row vectors of matrix B^α defined as $(\check{B}^{i\alpha})_k \equiv B_{ik}^\alpha$ to rewrite Eq. (46) as

$$\begin{aligned}\langle \mathcal{V}^q \rangle &= \int \mathrm{d}R \int \mathrm{D}\check{B} \prod_{\alpha,\beta} \prod_{i\leq j} \delta(\check{B}_i^{\alpha\mathrm{T}} \check{B}_j^\beta - R_{ij}^{\alpha\beta})\, \mathcal{G}_{ij}^p \\ \mathcal{G}_{ij} &= \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^{\infty} \mathrm{D}\tilde{x}\, e^{-\frac{1}{2}\sum_{\alpha,\beta} \tilde{x}^\alpha \left(R_{ij}^{\alpha\alpha} R_{ij}^{\beta\beta} + fc^2 R_{ii}^{\alpha\beta} R_{jj}^{\alpha\beta} + fc^2 R_{ij}^{\alpha\beta} R_{ij}^{\beta\alpha}\right)\tilde{x}^\beta - \imath \sum_\alpha \tilde{x}^\alpha x'^\alpha} \,,\end{aligned} \tag{49}$$

with $\int \mathrm{d}R \equiv \prod_{\alpha,\beta}\prod_{i<j} \int \mathrm{d}R_{ij}^{\alpha\beta} \prod_{\alpha\neq\beta} \int \mathrm{d}R_{ii}^{\alpha\beta}$ and $\int \mathrm{D}\check{B} \equiv \prod_\alpha \prod_i \prod_{k=1}^m \int \mathrm{d}\check{B}_k^{i\alpha}$. We used that $\mathcal{G}$ is identical for different patterns $\hat{P}^r$, hence we may perform the product over r by taking the p-th power. We express the normalization constraint as

$$\delta(\check{B}^{i\alpha\mathrm{T}} \check{B}^{i\alpha} - 1) = \frac{1}{2\pi\imath} \int_{-i\infty}^{i\infty} \mathrm{d}\tilde{R}_{ii}^{\alpha\alpha} \exp\big(-\tilde{R}_{ii}^{\alpha\alpha}\,(\check{B}^{i\alpha\mathrm{T}} \check{B}^{i\alpha} - 1)\big). \tag{50}$$

Analogously, we employ this Fourier representation of the Dirac δ to express the constraints defining the auxiliary fields Eq. (48) to obtain

$$\langle \mathcal{V}^q \rangle = \int \mathrm{d}R \int \mathrm{d}\tilde{R} \, \exp(\mathcal{S}) \tag{51}$$

$$\mathcal{S} = m \, \ln(F) + \sum_{i<j} p \ln(\mathcal{G}_{ij}) + \mathcal{H} \tag{52}$$

$$\mathcal{F} = \int \mathrm{d}\check{B} \, \exp\Big(- \sum_{\alpha,\beta,i\leq j} \tilde{R}_{ij}^{\alpha\beta} \check{B}_i^\alpha \check{B}_j^\beta \Big) \tag{53}$$

$$\mathcal{H} = \sum_{\alpha,\beta,i\leq j} \tilde{R}_{ij}^{\alpha\beta} R_{ij}^{\alpha\beta} \tag{54}$$

with $\int \mathrm{d}\check{B} \equiv \prod_\alpha \prod_i \int \mathrm{d}\check{B}_i^\alpha$ and $R_{ii}^{\alpha\alpha} = 1$. In defining $\mathcal{F}$ we used that the integral $\int \mathrm{D}\check{B} \exp\big(- \sum_{\alpha,\beta,i\leq j} \tilde{R}_i^{\alpha\beta} \sum_k \check{B}_k^{i\alpha} \check{B}_k^{i\beta}$ factorizes in the index k so that we get m times the same integral for each component $\check{B}_k^{i\alpha} \, \forall k = 1, \ldots, m$.

We are interested in the saddle points of the integrals $\int \mathrm{d}R \int \mathrm{d}\tilde{R}$ and search for a replica-symmetric solution. We therefore set

$$\begin{aligned} R_{ij}^{\alpha\alpha} &= R_{ij}^{=}, \qquad \tilde{R}_{ij}^{\alpha\alpha} = \tilde{R}_{ij}^{=} \\ R_{ij}^{\alpha\beta} &= R_{ij}^{\neq}, \qquad \tilde{R}_{ij}^{\alpha\beta} = \tilde{R}_{ij}^{\neq} \end{aligned} \tag{55}$$

for $\alpha \neq \beta$. Then in the limit $q \to 0$ we get

$$\mathcal{H} = q \sum_{i\leq j} \tilde{R}_{ij}^{=} R_{ij}^{=} - q \sum_{i\leq j} \tilde{R}_{ij}^{\neq} R_{ij}^{\neq} \tag{56}$$

which gives rise to the following saddle point equations

$$R_{ij}^{=} = -\frac{m}{q} \frac{\partial \ln(\mathcal{F})}{\partial \tilde{R}_{ij}^{=}} \, , \quad R_{ij}^{\neq} = \frac{m}{q} \frac{\partial \ln(\mathcal{F})}{\partial \tilde{R}_{ij}^{\neq}} \tag{57}$$

$$\tilde{R}_{ij}^{=} = -\frac{p}{q} \sum_{k<l} \frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{=}} \, , \quad \tilde{R}_{ij}^{\neq} = \frac{p}{q} \sum_{k<l} \frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{\neq}} \tag{58}$$

The above equations show that we need to find the contribution of $\ln(\mathcal{F})$ and $\ln(\mathcal{G})$ proportional to q as this is the only one surviving in the $q \to 0$ limit.

C.4 Limit $q \to 0$

For replica symmetry, the exponent in $\mathcal{G}_{ij}$ simplifies to

$$\sum_{\alpha,\beta} \tilde{x}^\alpha \Big(R_{ij}^{\alpha\alpha} R_{ij}^{\beta\beta} + f c^2 \, R_{ii}^{\alpha\beta} \, R_{jj}^{\alpha\beta} + f c^2 \, R_{ij}^{\alpha\beta} \, R_{ij}^{\beta\alpha} \Big) \tilde{x}^\beta = (\lambda_{ij}^{=} - \lambda_{ij}^{\neq}) \sum_\alpha \tilde{x}^\alpha \tilde{x}^\alpha + \lambda_{ij}^{\neq} \left(\sum_\alpha \tilde{x}^\alpha \right)^2 \, , \tag{59}$$

with $\lambda_{ij}^{=} = f c^2 \, R_{ii}^{=} R_{jj}^{=} + (1 + f c^2) \, R_{ij}^{=2}$ and $\lambda_{ij}^{\neq} = f c^2 \, R_{ii}^{\neq} R_{jj}^{\neq} + R_{ij}^{=2} + f r^2 \, R_{ij}^{\neq 2}$. The replica are coupled by the factor $\lambda_{ij}^{\neq}$, which renders $\int_{-\infty}^{\infty} \mathrm{D}\tilde{x}$ in $\mathcal{G}_{ij}$ an q-dimensional integral. In order to apply the limit $q \to 0$, it is convenient to decouple the replicas by performing the Hubbard-Stratonovich transformation

$$\exp\Big(-\frac{1}{2} \lambda_{ij}^{\neq} \Big(\sum_\alpha \tilde{x}^\alpha \Big)^2 \Big) = \int_{-\infty}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\Big(-t^2/2 + it \sqrt{\lambda_{ij}^{\neq}} \sum_\alpha \tilde{x}^\alpha \Big) \, , \tag{60}$$

which turns the $2q$-dimensional integral $\int_{-\infty}^{\infty} \mathrm{D}x \int_{-\infty}^{\infty} \mathrm{D}\tilde{x}$ into a Gaussian integral over the qth power of a function $g_{ij}(t)$ that is given by a two-dimensional integral

$$g_{ij}(t) = \int_\kappa^\infty \mathrm{d}x \int_{-\infty}^{\infty} \frac{\mathrm{d}\tilde{x}}{2\pi} \exp\Big(-\frac{1}{2} (\lambda_{ij}^{=} - \lambda_{ij}^{\neq}) \tilde{x}^2 + it \sqrt{\lambda_{ij}^{\neq}} \tilde{x} - i\tilde{x}x \Big) = \frac{1}{2} \mathrm{erfc}(a_{ij}(t)) \, , \tag{61}$$

with $a_{ij}(t) = (\kappa - t\sqrt{\lambda_{ij}^{\neq}})/\sqrt{2(\lambda_{ij}^{=} - \lambda_{ij}^{\neq})}$. The resulting form of $\mathcal{G}_{ij}$ allows to take advantage of the $q \to 0$ limit by approximating

$$\begin{aligned} \ln(\mathcal{G}_{ij}) &= \ln\langle g_{ij}(t)^q\rangle = \ln\left\langle \exp\left(q \ln\left(g_{ij}(t)\right)\right)\right\rangle \\ &\to \ln\left(1 + q\left\langle \ln\left(g_{ij}(t)\right)\right\rangle\right) \to q\left\langle \ln\left(g_{ij}(t)\right)\right\rangle \\ &= q\left\langle \ln\left(\operatorname{erfc}\left(a_{ij}(t)\right)\right)\right\rangle + \ln(1/2)\ . \end{aligned} \tag{62}$$

C.5 Limiting capacity

We are interested in the limit $R_{ii}^{\neq} \to R_{ii}^{=} = 1$, which denotes the point where only a single solution is found: the overlap of the readout between replicas is identical to the length of the vector in each individual replicon, so only a single solution is found. So we set $R_{ii}^{\neq} = 1 - \epsilon_i$ and study the limit $\epsilon_i \to 0$ for all $i \in [1, m]$. We need to be careful in taking this limit as $\ln(\mathcal{G}_{ij})$ is singular for $\epsilon = 0$. The saddle-point equations relate derivatives of $\ln(\mathcal{G}_{ij})$ to tilde-fields, which in turn are defined by $\ln(\mathcal{F})$. A singularity in $\ln(\mathcal{G}_{ij})$ at $\epsilon = 0$ therefore implies also a singularity in $\ln(\mathcal{F})$. These singularities will cancel in the following in the calculation of the capacity.

In the following, we first focus on the fields $R_{ij}^{=}$ and $R_{ij}^{\neq}$ for $i < j$: The function $\ln G_{ij}$ depends quadratically on $R_{ij}^{=}$ and $R_{ij}^{\neq}$ (see Eq. (49)). By Taylor expansion of Eq. (53) around $\tilde{R}_{ij}^{=} = \tilde{R}_{ij}^{\neq} = 0$, one can observe that all odd Taylor coefficients vanish since they are determined by odd moments of a Gaussian integral with zero mean. Therefore, also $\ln F$ depends quadratically on $\tilde{R}_{ij}^{=}$ and $\tilde{R}_{ij}^{\neq}$. By rewriting Eq. (58) as $\tilde{R}_{ij}^{=} = -2R_{ij}^{=}\frac{p}{n}\sum_{k<l}\frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{=2}}$ and $R_{ij}^{\neq} = -2\tilde{R}_{ij}\frac{m}{n}\frac{\partial \ln(\mathcal{F})}{\partial \bar{U}_{ij}^2}$, respectively, and analogously for $\tilde{R}_{ij}^{\neq}$ and $R_{ij}^{\neq}$, we see that $R_{ij}^{=} = \tilde{R}_{ij}^{=} = R_{ij}^{\neq} = \tilde{R}_{ij}^{\neq} = 0$ is a solution to the saddle point equations. This solution makes sense as $R_{ij}^{=}$ represents a displacement of the Q_{ij}, therefore a non-vanishing value would hinder the classification. At the point of limiting capacity all replicas find the same solution. Therefore, also the overlap $R_{ij}^{\neq}$ across replica must vanish. Using $\tilde{R}_{ij}^{=} = \tilde{R}_{ij}^{\neq} = 0$, an analogous procedure as in Section C.4 can be performed to calculate the term $\ln(\mathcal{F})$ in the $q \to 0$ limit

$$\ln(\mathcal{F}) \to -\frac{1}{2}q\sum_i\left(\ln\left(\tilde{R}_{ii}^{=} - \tilde{R}_{ii}^{\neq}\right) + \tilde{R}_{ii}^{\neq}/\left(\tilde{R}_{ii}^{=} - \tilde{R}_{ii}^{\neq}\right)\right) + \text{const}\ . \tag{63}$$

Then Eq. (57) can be easily solved to obtain

$$\tilde{R}_{ii}^{=} = -\frac{m}{2}\frac{1-2\epsilon_i}{\epsilon_i^2}, \quad \tilde{R}_{ii}^{\neq} = -\frac{m}{2}\frac{1-\epsilon_i}{\epsilon_i^2}\ . \tag{64}$$

Inserting the solution Eq. (64) into Eq. (58) and using Eq. (62), we get in the limit $\epsilon_i \to 0$

$$-\frac{m}{2}\frac{1}{\epsilon_i^2} = p\sum_{k<l}\left(\frac{\partial\left\langle \ln\left(\operatorname{erfc}(a_{kl}(t))\right)\right\rangle}{\partial R_{kk}^{\neq}}\delta_{ki} + \frac{\partial\left\langle \ln\left(\operatorname{erfc}(a_{kl}(t))\right)\right\rangle}{\partial R_{ll}^{\neq}}\delta_{li}\right) \tag{65}$$

$$= p\sum_{k<l}\int_{-\infty}^{\infty}\frac{dt}{\sqrt{2\pi}}\exp\left(-t^2/2\right)\frac{\frac{\partial}{\partial a_{kl}(t)}\operatorname{erfc}(a_{kl}(t))}{\operatorname{erfc}(a_{kl}(t))}\left(\frac{\partial a_{kl}(t)}{\partial R_{kk}^{\neq}}\delta_{ki} + \frac{\partial a_{kl}(t)}{\partial R_{ll}^{\neq}}\delta_{li}\right) \tag{66}$$

$$= p\sum_{k<l}\int_{-\infty}^{\infty}\frac{dt}{\sqrt{2\pi}}\exp\left(-t^2/2\right)\frac{-\frac{2}{\sqrt{\pi}}e^{-a_{kl}(t)^2}}{\operatorname{erfc}(a_{kl}(t))}\left(\frac{\partial a_{kl}(t)}{\partial R_{kk}^{\neq}}\delta_{ki} + \frac{\partial a_{kl}(t)}{\partial R_{ll}^{\neq}}\delta_{li}\right)$$

For $\epsilon_k, \epsilon_l \to 0$ the function $a_{kl}(t)$ goes to negative infinity for $t > \bar{\kappa} \equiv \kappa/\sqrt{fr^2}$ and $\operatorname{erfc}(a_{kl}(t)) \to 2$. In this case the nominator in the integrand makes the integral vanish. Therefore, we can restrict the integration range to $t \in (-\infty, \bar{\kappa}]$, where $a_{kl}(t) \to \infty$ for $\epsilon \to 0$, such that we can insert the limit behavior of $\operatorname{erfc}(a_{kl}(t)) \to e^{-a_{kl}(t)^2}/(\sqrt{\pi}a_{kl}(t))$. Using

$$\frac{\partial a_{kl}(t)^2}{\partial R_{kk}^{\neq}} \to \frac{(\bar{\kappa} - t)^2}{2\epsilon_k^2}, \tag{67}$$

the limiting capacity follows from

$$\frac{m}{\epsilon_i^2} = p \int_{-\infty}^{\bar{\kappa}} \frac{dt}{\sqrt{2\pi}} \exp\left(-t^2/2\right) \frac{\left(\bar{\kappa}-t\right)^2}{\epsilon_i^2} \sum_{k<l} \left(\delta_{ki}+\delta_{li}\right) \tag{68}$$

as

$$\mathcal{C} \equiv \frac{n(n-1)}{2}\frac{p}{m} = \frac{n}{2}\left(\int_{-\bar{\kappa}}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right)\left(t+\bar{\kappa}\right)^2\right)^{-1} . \tag{69}$$

The capacity is identical to the capacity of the mean perceptron. In particular, for $\kappa = 0$, we get

$$\mathcal{C} = n \ . \tag{70}$$

D Supplementary results

D.1 Stability of ongoing learning

Fig. S1 illustrates the stability of the learning procedure, both to decrease the error to the best possible minimum and avoid the "explosion" of recurrent weights A, i.e. them diverging to $\pm\infty$. Here all 10 input patterns have the same objective matrices, $\bar{Q}^0$ in Fig. S1A-B and a pair $\bar{Q}^0$ and $\bar{Q}^1$ in Fig. S1C-D. In both cases, the error firstly decreases then stabilizes, still slowly decreasing. The evolution of the Pearson correlation indicates that the structure of the output(s) remains stable over the optimization, even though the network may not perfectly converge towards the objective(s) in error or Pearson correlation.

The procedure to generate realistic input and objective output patterns in Fig. S1C-D aims to ensure that a solution for A and B exists. Indeed, for usual time series, P^0 and P^1 are not independent, and the choice of the MAR model for the network dynamics similarly imposes constraints on $\bar{Q}^0$ and $\bar{Q}^1$. Nevertheless, we had $\sim 15\%$ of the optimizations failing due to an explosion for A. For completely random $\bar{Q}^0$ and $\bar{Q}^1$, the conclusion from numerical simulations is that the explosion of A is very likely.

D.2 Shaping output spatio-temporal covariances

As shown in Fig. S2A, we want to tune both B and A to obtain a desired spatio-temporal structure in output. We consider inputs x_k^t with spatial covariances only (since $P^1 = 0$) to be mapped to spatio-temporal covariances for y_i^t. For this purpose, we generalize Eq. (6) to calculate the weight updates for A and B from the errors of both Q^0 and Q^1:

$$\begin{aligned}
\Delta B_{ik} &= \eta_B \left[\left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial B_{ik}} + \left(\bar{Q}^1 - Q^1\right) \odot \frac{\partial Q^1}{\partial B_{ik}}\right] , \\
\Delta A_{ij} &= \eta_A \left[\left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial A_{ij}} + \left(\bar{Q}^1 - Q^1\right) \odot \frac{\partial Q^1}{\partial A_{ij}}\right] .
\end{aligned} \tag{71}$$

The matrix "derivatives" are given by Eq. (30), Eq. (33), Eq. (32) and Eq. (34) in Annex A while setting $P^1 = P^{-1\mathrm{T}} = 0$, which read in matrix form:

$$\begin{aligned}
\frac{\partial Q^0}{\partial B_{ik}} &= A\frac{\partial Q^0}{\partial B_{ik}}A^{\mathrm{T}} + U^{ik}P^0B^{\mathrm{T}} + BP^0U^{ik\mathrm{T}} , \\
\frac{\partial Q^1}{\partial B_{ik}} &= A\frac{\partial Q^1}{\partial B_{ik}}A^{\mathrm{T}} + AU^{ik}P^0B^{\mathrm{T}} + ABP^0U^{ik\mathrm{T}} , \\
\frac{\partial Q^0}{\partial A_{ij}} &= A\frac{\partial Q^0}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^0A^{\mathrm{T}} + AQ^0V^{ij\mathrm{T}} , \\
\frac{\partial Q^1}{\partial A_{ij}} &= A\frac{\partial Q^1}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^1A^{\mathrm{T}} + AQ^1V^{ij\mathrm{T}} + V^{ij}BP^0B^{\mathrm{T}} .
\end{aligned} \tag{72}$$

Similar to U^{ik}, the $n \times m$ matrix V^{ij} has 0 everywhere except for element (i,j). The key to evaluate the weight update for A is seeing that the third and fourth lines correspond to the discrete Lyapunov equation that can be solved at each optimization step. As before, we randomly draw 10 input patterns to

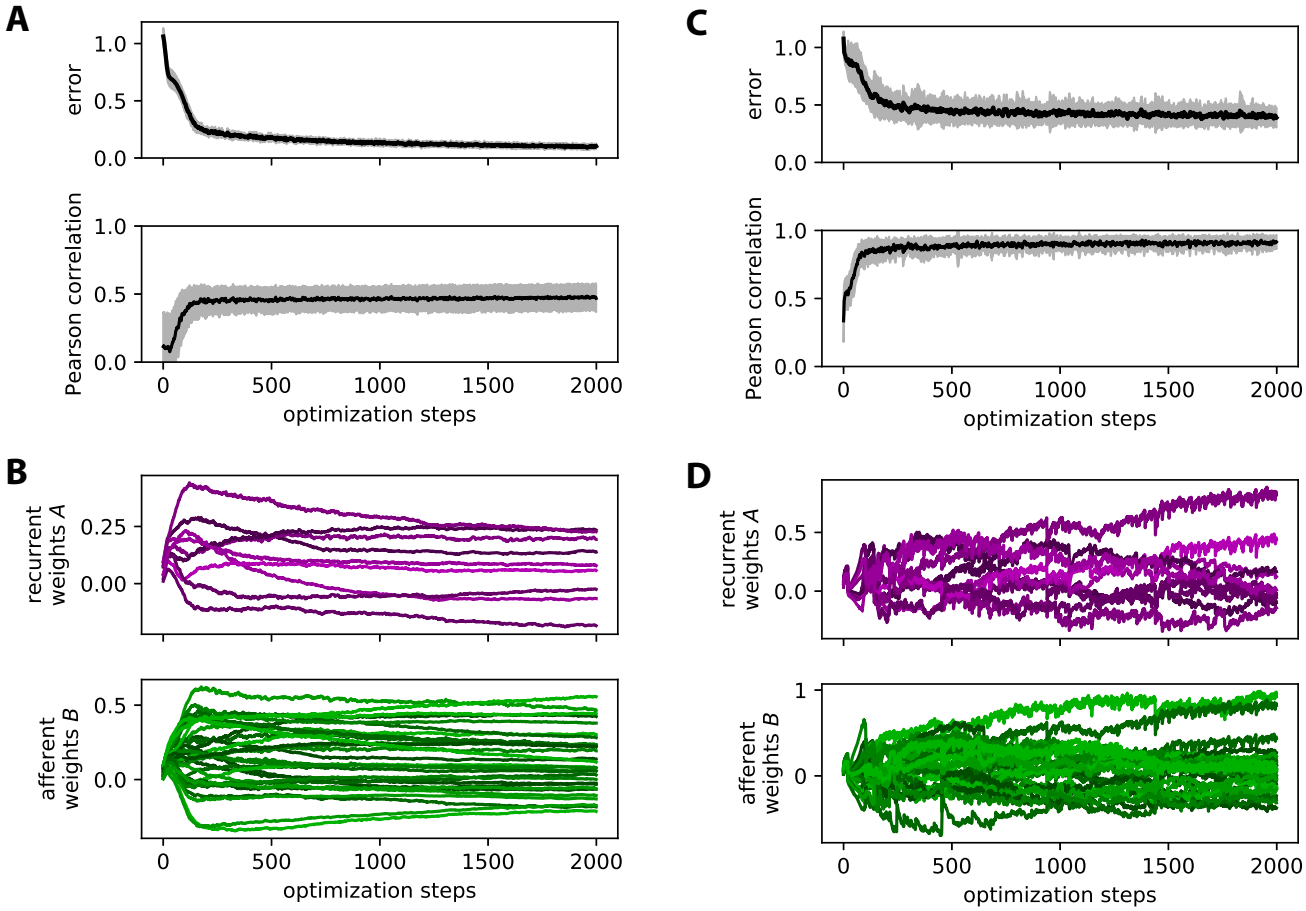

Figure S1: **Stability of ongoing learning.** **A** Evolution of the error for 20 optimizations of networks with $m = 10$ inputs to the $n = 3$ outputs. Here the objective output covariance matrix $\bar{Q}^0$ is random, as are the 10 random input patterns P^0 and P^1, but no tuning for Q^1 is performed. The plot is similar to Fig. 2F with the error (matrix distance) and the Pearson correlation between Q^0 and $\bar{Q}^0$ over all 20 optimizations: The black trace corresponds to the mean over the 20 optimizations and the gray area to the standard deviation. **B** Example evolution of the afferent and recurrent weights (green and purple traces, respectively) for an optimization. **C-D** Same as panels A-B for 20 optimizations of the same type of networks with more "realistic" objective pairs $\bar{Q}^0$ and $\bar{Q}^1$, as well as P^0 and P^1 input patterns. Both pairs are generated by a MAR, Eq. (16), which yields the consistency equations $P^1 = WP^0$ and $P^0 - WP^0W^{\mathrm{T}} = \mathbf{1}$ for a given W. The plotted values correspond to the mean of the two errors or Pearson correlations for Q^0 and Q^1.

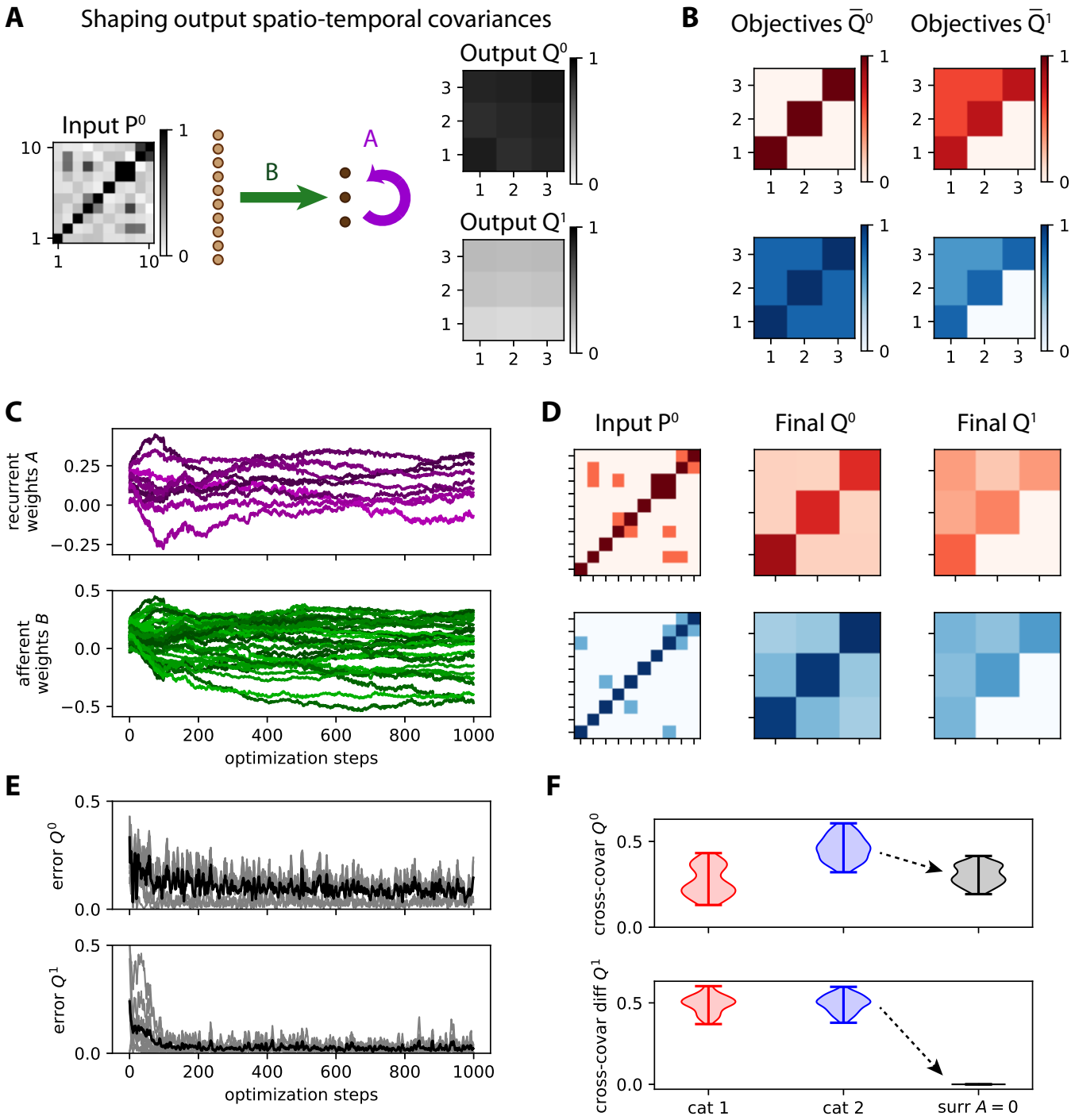

Figure S2: **Shaping output spatio-temporal covariances with both afferent and recurrent connectivities.** **A** Network architecture with $m = 10$ input nodes and $n = 3$ output nodes, the latter being connected together by the recurrent weights A (purple arrow). **B** Objective covariance matrices for two categories (red and blue). **C** Evolution of the afferent and recurrent weights (green and purple traces, respectively). **D** Two examples after training of output patterns Q^0 and Q^1 in response to two input patterns P^0, among the 5 in each category. **E** Evolution of the error for the two output covariance matrices. **F** After training, the covariances in Q^0 allow for the discrimination between the two categories, while the structure of Q^1 is similar for the two categories. The plot is similar to Fig. 3. The black surrogate corresponds to forcing $A = 0$ with the trained B and presenting the blue inputs, demonstrating that the trained A is important in shaping the output structure.

be classified into 2 categories of 5 each, whose objective matrices Q^0 and Q^1 are represented in Fig. S2B. A positive outcome is that the weight updates lead to rather stable learning dynamics, even for the recurrent connectivity in Fig. S2C. The stability of ongoing learning while leaving classification aside is examined in Annex D.1, see Fig. S1. Meanwhile, the errors for both Q^0 and Q^1 decrease and eventually stabilize close to zero in Fig. S2E.

After training, the network maps the input patterns P^0 in the desired manner for Q^0 and Q^1, see the two examples in Fig. S2D and the robustness test in Fig. S2F —in a similar manner to Fig. 3. The surrogates (black distribution in Fig. S2F) correspond to setting $A = 0$ with the trained B, which strongly affects the output covariance (here for blue input patterns). This illustrates the importance of tuning the recurrent connectivity in shaping Q^1, as well as with the discrimination capability for Q^0.

D.3 Learning input spatio-temporal covariances

Now we consider the "converse" configuration of Fig. S2A where each input pattern is formed by a pair of non-zero P^0 and P^1, see Fig. S3A. The output is trained only using Q^0, meaning that the input spatio-temporal structure is mapped to an output spatial structure. This time simplifying Eq. (71), the weight updates are given by Eq. (18), which corresponds to discrete Lyapunov equations that can be solved at each optimization step to evaluate the weight update for A and B.

We first examine the specialization in terms of covariances in Q^0 as defined by the objectives in Fig. S3C. Here we take input patterns P^0 that are all identical (left matrices in Fig. S3B) such that the weight specialization must be based on the discrepancies between P^1 across inputs, even though this configuration may not be realistic for simulated time series. The desired outcome after training is obtained as illustrated in Fig. S3C. The surrogates (in black) indicate the importance of the trained recurrent connectivity A, although it appears less strong here than in Fig. S2F. Despite incidental troughs, the classification accuracy increases and eventually stabilizes around 90%. Second, Fig. S3D uses the same procedure for specializing the variances in Q^0 and shows similar conclusions. Together, these results demonstrate a useful flexibility in tuning the input-output covariance mapping using the MAR network.

References

[1] S.-i. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[2] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen. Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses. 273(5283):1868–1871, 1996.

[3] B. B. Averbeck, P. E. Latham, and A. Pouget. Neural correlations, population coding and computation. *Nat Rev Neurosci*, 7:358–366, 2006.

[4] O. Barak and M. Tsodyks. Recognition by variance: learning rules for spatiotemporal patterns. *Neural Comput*, 18:2343–2358, 2006.

[5] B. Bercu, F. Proïa, and N. Savy. On ornstein–uhlenbeck driven by ornstein–uhlenbeck processes. *Statistics and Probability Letters*, 85:36–44, 2014.

[6] P. Berkes, G. Orbán, M. Lengyel, and J. Fiser. Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331:83–87, 2011.

[7] G. Bi and M. Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. 18:10464–10472, 1998.

[8] C. M. Bishop. *Pattern Recognition and Machine Learning*. Number 978-0-387-31073-2. Springer, 2006.

[9] N. Brunel, J. P. Nadal, and G. Toulouse. Information capacity of a perceptron. *Journal of Physics A: Mathematical and General*, 25:5017–5038, 1992.

[10] S. Choi, A. Cichocki, and S. Amari. Equivariant nonstationary source separation. *Neural Netw*, 15:121–130, 2002.

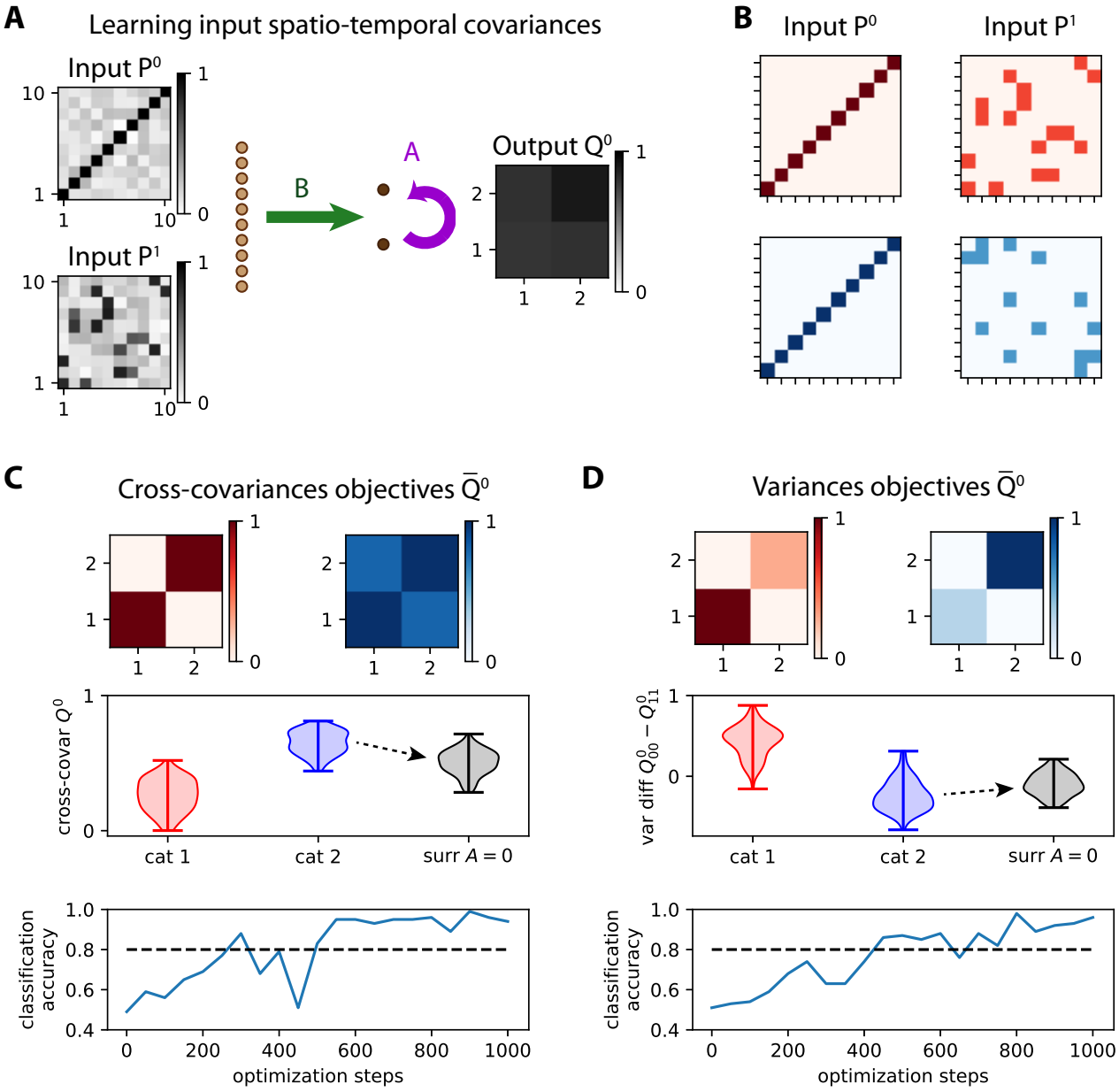

Figure S3: **Learning input spatio-temporal covariances with both afferent and recurrent connectivities.** **A** Similar network to Fig. S2A with $m = 10$ input nodes and $n = 2$ output nodes. **B** Two examples of input patterns corresponding to a pair P^0 and P^1, among the 5 in each category. The P^0 matrices are identical for all patterns. **C** Classification based on specializing covariances for the two categories: absent for red and positive for blue (top matrices, same as Fig. 3D). The middle plot is similar to Fig. 3, where the separability of the red and blue distributions indicates the performance of the classification. The comparison between the black and blue distribution shows the importance of the recurrent connectivity A, which is forced to 0 for the surrogates. The bottom plot indicates the evolution of the classification accuracy during the optimization. The binary classification uses the same boundary as in Fig. 3E. **D** Same as panel C for specializing the variances of the output nodes, with the same objective matrices and classification procedure as in Fig. 3A-B.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[12] T. M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14:326–334, 1965.

[13] K. H. Fischer and J. A. Hertz. *Spin Glasses*. Cambridge University Press, 1991.

[14] B. Gardner and A. Grüning. Supervised learning in spiking neural networks for precise temporal encoding. *PLoS One*, 11:e0161335, 2016.

[15] E. Gardner. The space of interactions in neural network models. *Journal of Physics A: Mathematical and General*, 21:257, 1988.

[16] M. Gilson, A. Burkitt, and L. J. van Hemmen. Stdp in recurrent neuronal networks. *Front Comput Neurosci*, 4:23, 2010.

[17] M. Gilson, T. Masquelier, and E. Hugues. Stdp allows fast rate-modulated coding with poisson-like spike trains. *PLoS Comput Biol*, 7:e1002231, 2011.

[18] M. Gilson, R. Moreno-Bote, A. Ponce-Alvarez, P. Ritter, and G. Deco. Estimation of directed effective connectivity from fmri functional connectivity hints at asymmetries of cortical connectome. *PLoS Comput Biol*, 12:e1004762, 2016.

[19] R. Gütig, T. Gollisch, H. Sompolinsky, and M. Meister. Computing complex visual features with retinal spike times. *PLoS One*, 8:e53063, 2013.

[20] R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9:420–428, 2006.

[21] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Longman, 1991.

[22] R. Kempter, W. Gerstner, and J. Van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498–4514, 1999.

[23] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Netw*, 99:56–67, 2018.

[24] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[25] H. Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[26] Z. F. Mainen and T. J. Sejnowski. Reliability of spike timing in neocortical neurons. 268:1503–1506, 1995.

[27] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. 275:213–215, 10. January 1997.

[28] M. L. Minsky and S. A. Papert. *Perceptrons*. Cambridge MIT Press, 1969.

[29] R. Moreno-Bote, J. Beck, I. Kanitscheider, X. Pitkow, P. Latham, and A. Pouget. Information-limiting correlations. *Nat Neurosci*, 17:1410–1417, 2014.

[30] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol*, 9:e1003037, 2013.

[31] E. Oja. A simplified neuron model as a principal component analyzer. *J Math Biol*, 15:267–273, 1982.

[32] G. Orbán, P. Berkes, J. Fiser, and M. Lengyel. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron*, 92:530–543, 2016.

[33] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Trans Neural Netw*, 6:1212–1228, 1995.

[34] F. J. Pineda. Generalization of back-propagation to recurrent neural networks. *Phys Rev Lett*, 59:2229–2232, Nov 1987.

[35] F. Ponulak and A. Kasiński. Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural Comput*, 22:467–510, 2010.

[36] L. Posani, S. Cocco, K. Ježek, and R. Monasson. Functional connectivity models for decoding of spatial representations from hippocampal ca1 recordings. *J Comput Neurosci*, 43:17–33, 2017.

[37] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*, 65:386–408, 1958.

[38] E. Rumelhart, David, E. Hinton, Geoffrey, and J. Williams, Ronald. Learning representations by back-propagating errors. 323:533–536, October 1986.

[39] J. Schmidhuber. Deep learning in neural networks: an overview. *Neural Netw*, 61:85–117, 2015.

[40] B. Widrow and M. E. Hoff. Adaptive switching circuits. In IRE, editor, *1960 IRE WESCON Convention Record (Part 4)*, pages 96–104, 1960.

[41] F. Zenke and S. Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput*, 30:1514–1541, 2018.

[42] F. Zenke, G. Hennequin, and W. Gerstner. Synaptic plasticity in neural networks needs homeostasis with a fast rate detector. *PLoS Comput Biol*, 9:e1003330, 2013.

[43] P. Zheng, C. Dimitrakakis, and J. Triesch. Network self-organization explains the statistics and dynamics of synaptic connection strengths in cortex. *PLoS Comput Biol*, 9:e1002848, 2013.