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

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

February 27, 2019

Abstract

Learning in neural networks has developed in many directions, from image recognition and speech processing to data analysis in general. Most theories relying on gradient descents tune the weights to constrain the output activity in a given range of activation levels, thereby focusing on the first-order statistics of the network activity. Here we propose a learning formalism akin to the delta rule to tune both afferent and recurrent weights in a network, which shapes the input-output mapping for covariances, i.e. the second-order statistics. Covariance patterns define a natural metric for time series that capture their propagating nature. We develop the theory for classification of time series with hidden dynamics, as determined by their spatio-temporal covariances. This brings a conceptual change of perspective by employing variability in the time series to represent the information to be learned, rather than merely being the noise that corrupts the mean signal. 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 sending neurons. Our theory also complements recent studies of the relationship between noise correlation and (de)coding, showing that activity variability can be the basis for information to be learned by neural networks.

1 Introduction

The present paper considers the problem of transmission of information conveyed by time series in a neural network, as illustrated in Fig. 1A. A classical assumption is that the information is embedded in the mean of each input (see Fig. 1B), which can be evaluated up to sampling noise over an observation window (of duration d). By tuning the network weights (green links and arrow in Fig. 1A), groups of patterns in the mean input activity can be mapped to specific output patterns of mean activity. For supervised learning, the delta rule iteratively evaluates the error in predicting the category for each stimulus presentation and modifies the weights accordingly to improve the classification accuracy [5]. Classification for vector patterns corresponds to the classical perceptron, which does not involve recurrent connectivity (in purple) and has a long history [29, 23]. In its original version the weights determine the linear summation of the input activity to obtain the output activity of a single node. Stereotypical output patterns define categories, as represented with the two distinct mean vectors in red and blue in Fig. 1B. Many refinements have been developed to stabilize learning, in particular non-linear transformations to rectify the output, using step or sigmoidal functions [23, 5]. The capacity of the perceptron, namely how many input patterns it can discriminate, has

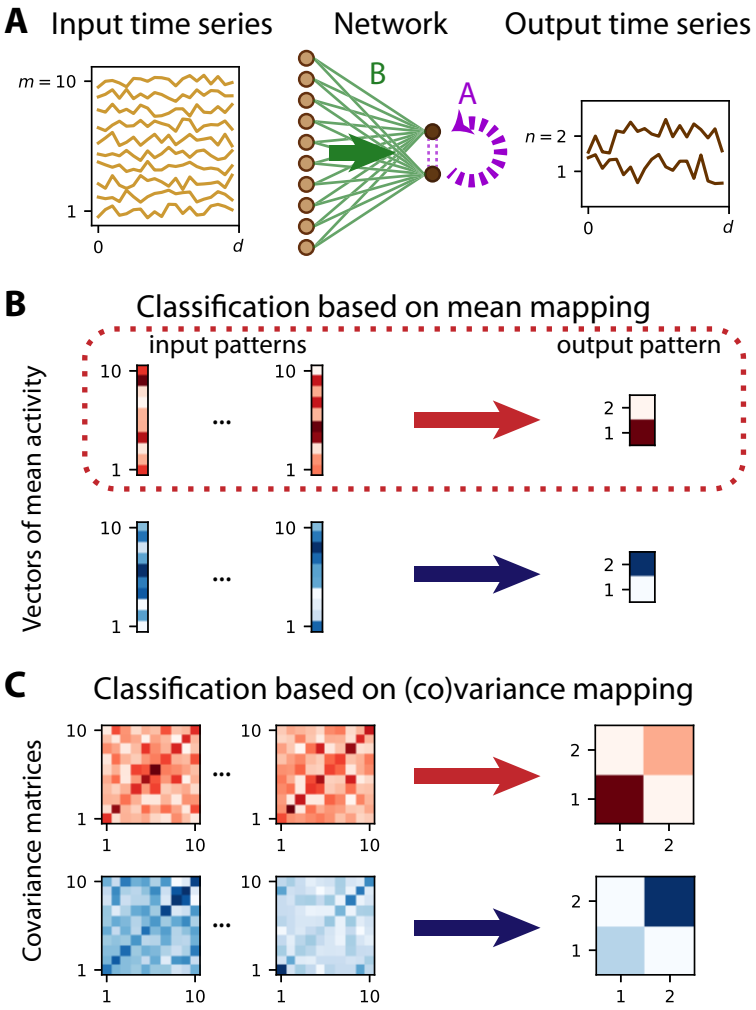

Figure 1: **From mean-based to covariance-based classification with time series. A:** Example network where $n = 2$ output nodes generate time series (in dark brown on the right) from the noisy time series of $m = 10$ input nodes (in light brown on the left). The afferent connections B (green links summarized by the 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:** From the observed time series in panel A, one can calculate the mean activity over the observation window, which gives a vector (darker pixels indicate stronger values). Tuning the connectivity weights such that several input patterns of mean activity (m-dimensional vectors on the left-hand side) are mapped to a same output pattern (n-dimensional vector of the right-hand side) implements a classification scheme, here represented for two categories in red (dotted rectangle) and blue. The mapping between input and output (mean) vectors corresponds to the classical perceptron, with two output nodes here. **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), for example larger variance for one of the two nodes here. The core of this study is a learning mechanism that tunes the connectivity to obtain good classification performance.

been the subject of many studies [9, 12, 31, 3]. To take time into account, the perceptron has been extended from static patterns to spatio-temporal patterns with the recurrent back-propagation [26] and back-propagation through time [25]. It led to modern classification machines like deep learning [21, 30], based on recent improvements in the network architecture like convolutional networks and thanks to computers with increasing power. Importantly, all these algorithms are based on the view that the time series is composed of a succession of snapshots of individual patterns (or trajectories) which are possibly corrupted by noise: they focus on the mean signals at each time (or window) and do not explicitly take into account their temporal correlations, arising from possible cofluctuations in the example of Fig. 1A.

Instead, the present study proposes a new framework for supervised learning that naturally captures the propagating nature of multivariate time series. The goal is the optimization of the mapping between input and output covariances in the network in Fig. 1A, moving from the first-order to the second-order statistics of the network activity. Building up on the classical 'mean perceptron' (Fig. 1B), we use the classification paradigm as an example to illustrate our theory and classify time series based on their covariance patterns (Fig. 1C). Meanwhile, we question the use for recurrent connectivity (in purple in Fig. 1A), for which the theory is much less developed than afferent connectivity (in green). The ansatz is that co-fluctuation patterns —here determined by covariances without or with time lags— convey information that can be used to train the network and to classify input time series into categories. In particular, we aim to use the covariance perceptron to discriminate time series that are characterized by hidden dynamics. This is a radically different viewpoint on signal variability compared to 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 richer representations for the same number of nodes that can be used for training and detection, which we assess in this study.

To fix ideas we use discrete network dynamics similar to the multivariate autoregressive (MAR) process [22]; extensions to continuous-time processes and other dynamical systems are discussed later. The output activity y_i^t with $1 \leq i \leq n$ in discrete time $t \in \mathcal{Z}$ depends on its own immediate past activity (i.e. with a unit time shift) and on the inputs x_k^t with $1 \leq k \leq m$:

$$y_i^t = \sum_{1 \leq j \leq n} A_{ij} y_j^{t-1} + \sum_{1 \leq k \leq m} B_{ik} x_k^t \ , \tag{1}$$

where the recurrent and afferent connections are embodied by the matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$, see Fig. 1A. As we deal with time series, we define the means

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

where the angular brackets indicate the average over realizations and over a period of duration d. The covariances with $\tau \in \mathcal{Z}$ being the time lag (positive or negative integers) are defined as

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

In this study we only consider centered variables with zero mean for covariance-based classification, so the second terms on the right-hand sides vanish in Eq. (3); considerations about

a mixed scenario based on both means and covariances will be discussed at the end of the paper. In addition, a threshold is applied on the output to perform classification, thereby implementing a non-linearity. However, the consistency and learning equations are derived for the linear system while assuming stationarity for the inputs, see Annexes A and B. The observation of limited data via the window of duration d in Fig. 1A results in "noise" compared to the theoretical estimates.

Section 2 considers the shaping of the input-output mapping for covariances by performing online training on the afferent connectivity. The theory is formulated in a similar manner to the delta rule, but adapted for covariances (and in a linear system). Firstly examined for an analytical abstraction of the network dynamics, the learning procedure is then verified on simulated time series while varying the duration d to see the effect of the observation noise. Section 3 explores the information capacity of the covariance perceptron with a non-linearity applied on the observed outputs, or readouts. It compares it with the classical mean perceptron for the same network architecture with only afferent connectivity. This corresponds to the case of offline learning (for the abstracted dynamics) with patterns that are presented to the perceptron without the above-mentioned noise. Section 4 extends the online training of Section 2 to a network with both afferent and recurrent connectivities in order to deal with the temporal structure of covariances, either in inputs or outputs. In that case a non-linearity arises from the network feedback related to the recurrent connectivity.

2 Online learning input-output covariance mappings in feed-forward networks

This section presents the concepts underlying the covariance perceptron 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). The example of Fig. 1B corresponds to the following input-output mapping for the mean vectors:

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

The derivation of this consistency equation with afferent connectivity B only —i.e. $A = 0$ in Eq. (1)— assumes stationarity for the inputs in the absence of temporal correlations ($P^{\tau}_{kl} = 0$ for $\tau \neq 0$). Under the same conditions, our proposed scheme relies on the mapping between the input and output covariance matrices, P^0 and Q^0 in Eq. (3):

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

where T denotes the matrix transpose. Details can be found in Annex A. 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.

2.1 Theory for learning of spatial correlation 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). It corresponds to the consistency equation Eq. (23), which is derived in Annex A assuming stationarity. In this section we also assume vanishing temporal correlations ($P^1 = P^{-1\mathrm{T}} = 0$) and absent recurrent connectivity ($A = 0$). As depicted in Fig. 2A, each training step consists in presenting an input pattern P^0 to the network and the

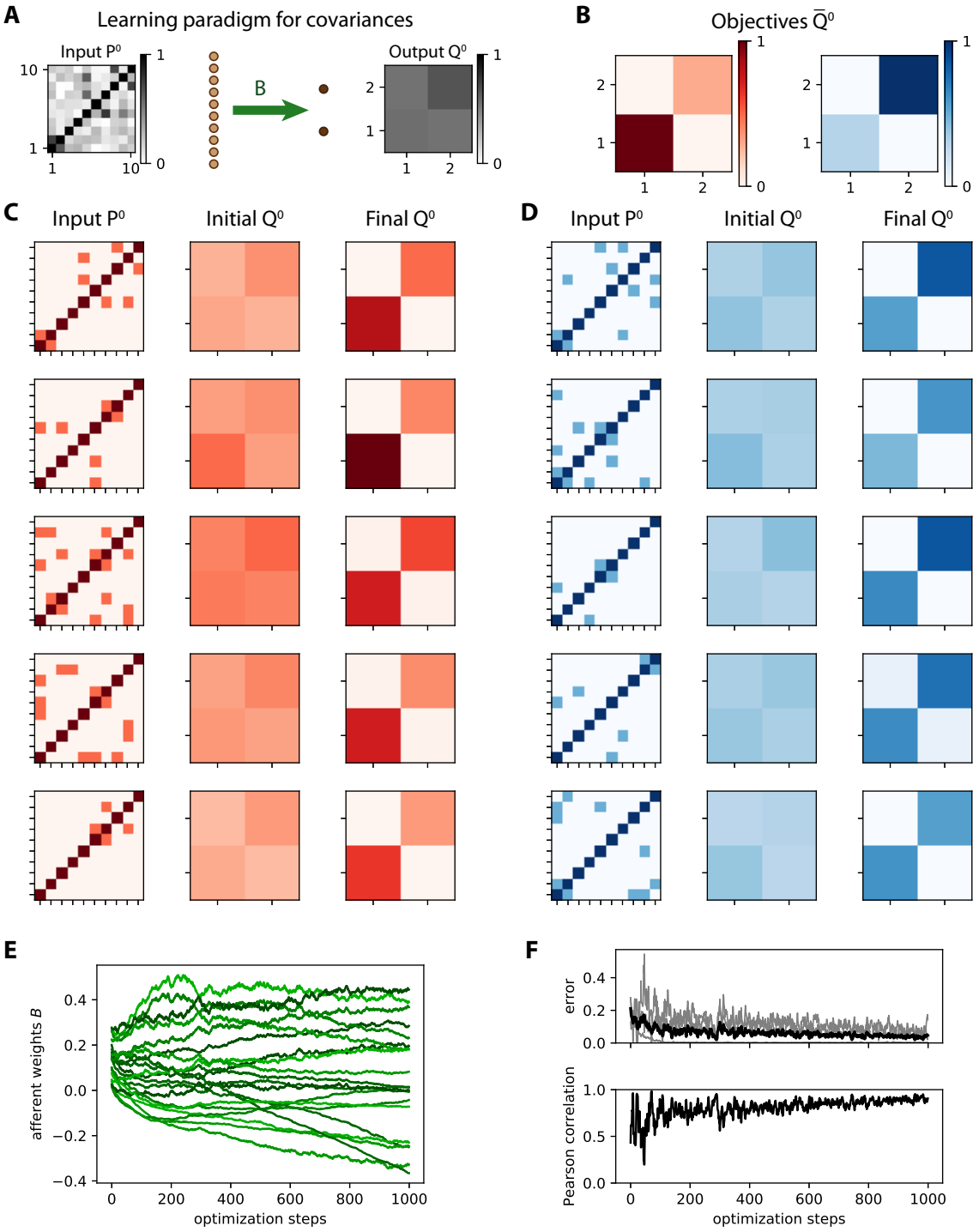

Figure 2: **Learning variances in a feed-forward network. A:** Schematic representation of the input-output mapping for covariances defined by the afferent weight matrix B, linking $m = 10$ input nodes to $n = 2$ output nodes. **B:** Objective output covariance matrices $\bar{Q}^0$ for two categories of inputs. **C:** Matrix for the 5 input covariance patterns P^0 (left column), with their image by 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:** Evolution of the error between Q^0 and $\bar{Q}^0$ at each step (top panel). The individual matrix elements correspond to the gray traces, and the total error taken as the matrix distance E in Eq. (25) to the thick black curve. The Pearson correlation coefficient between the vectorized Q^0 and $\bar{Q}^0$ (bottom panel) describes how they are "aligned", 1 corresponding to a perfect linear match.

resulting output pattern Q^0 is compared to the objective $\bar{Q}^0$ in Fig. 2B. For illustration purpose, we use 2 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, see the left matrix in Fig. 2A to be compared with 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 desired update for each afferent weight B_{ik} is described in Annex B and is based on the chain rule in Eq. (26), which gives, after combining Eq. (27) and Eq. (30) —again using $P^{-1} = 0$ and $A = 0$:

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

where U^{ik} is a $m \times m$ matrix with 0s everywhere except for element (i, k) that is equal to 1. 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 element involves U^{ik} that selects a single non-zero row for $U^{ik} P^0 B^{\mathrm{T}}$ and a single non-zero column for $B P^0 U^{ik\mathrm{T}}$. Therefore, the expression in Eq. (6) is different from $(\bar{Q}^0 - Q^0) P^0 B^{\mathrm{T}} + B P^0 (\bar{Q}^0 - Q^0)^{\mathrm{T}}$, 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. The afferent weights B_{ik} in Fig. 2E become specialized and tend to stabilize at the end of the optimization. Meanwhile, Fig. 2F shows the diminution 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 a threshold at zero for the difference between the output variances in order to make a binary classification, whose 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); note that variances are identical for the two categories here. 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 be efficiently trained to learn categories based on input covariances, just as with input vectors of mean activity.

2.2 Online learning for simulated time series relying on observation 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

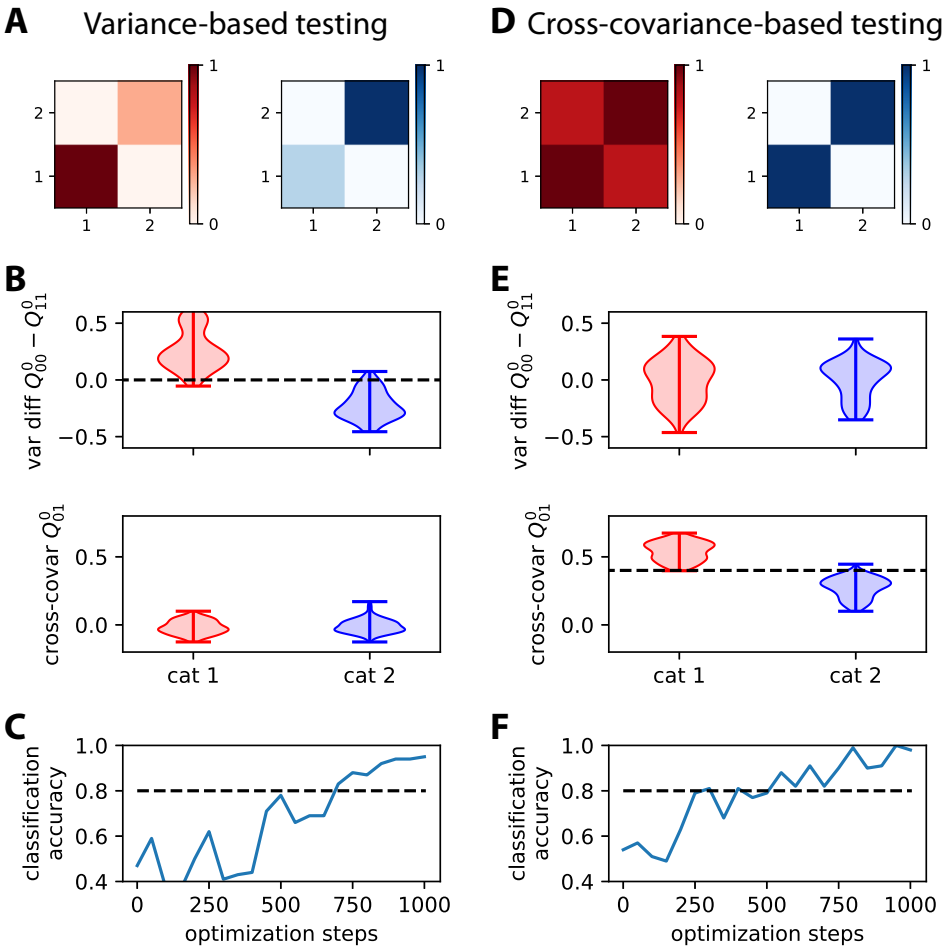

Figure 3: **Comparison between learning output patterns for variance and cross-covariance. A:** The top matrices represent the two objective covariance patterns of Fig. 2B, which differ by the variances for the two nodes. **B:** The plots display two measures based on the output covariance: the difference between the variances of the two nodes (top) and the level of covariance (bottom). Each violin plot corresponds to the distributions for the output covariance in response to 100 noisy versions of the 5 input patterns in the corresponding category. Note that the distributions are partly spread due to the artificial noise applied to the input covariances (see the main text about Fig. 2 for details). The separability between the red and blue distributions indicates a good classification and the dashed line indicates the threshold at zero used for binary classification based on the variance difference. **C:** Evolution of the classification accuracy based on the variance difference between the output nodes during the optimization. The binary classification corresponds to the threshold in panel B and predicts 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 A for two objective covariance patterns that differ by the cross-covariance level, strong for red and zero for blue. The classification in panel F corresponds to a threshold for the cross-covariance at 0.4 set at the middle of the target cross-covariance values (0.8 and 0).

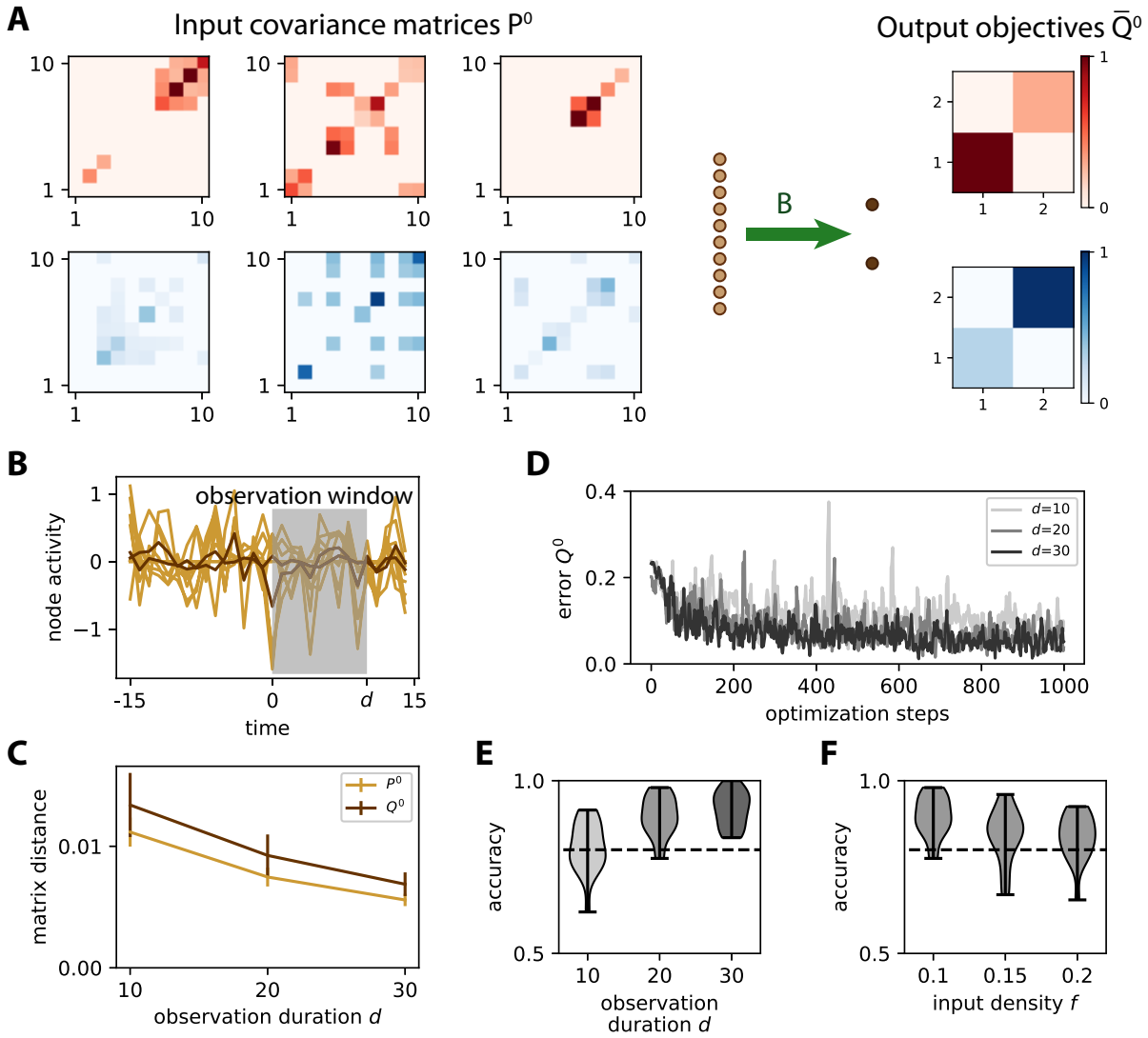

Figure 4: **Online learning input covariances by tuning afferent connectivity. A:** The same network as in Fig. 2A is trained to learn the input spatial covariance structure P^0 determined by the graphical model in Eq. (7). Only 3 matrices $P^0 = WW^{\mathrm{T}}$ out of the 5 for each category are displayed here. Each matrix element in each 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 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 for random connectivities when varying the duration d of the observation window. The error bars indicate the standard error of the mean over 100 noisy versions of the 5 input patterns in each category. **D:** Evolution of the error for 3 example optimizations with various observation duration d as indicated in the legend. **E:** Classification accuracy after training as a function of d pooled for 20 network and input configurations similar to Fig. S3D. 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 $d = 20$.

samples of the process itself are presented, rather than their statistics, the matrices P and Q. 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. Note that the weight update is applied for each pattern presentation.

To do so, we represent the correlation structure among the units by 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:

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

We use 10 patterns for $P^0 = WW^{\mathrm{T}}$ with W having $f = 10\%$ density on average to generate noisy input time series, which the network has to classify based on the variance of the output nodes, see Fig. 4A where only 3 input patterns per category are represented. The input time series thus differ by the spatial correlation structure of their activity.

The covariances from the time series are computed using an observation window of duration d, after discarding a stabilization simulation 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 learning to occur in a proper manner.

As expected, a longer observation duration d helps with stabilizing 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. To assess the quality of the training, we repeat the simulations for 20 network and input configurations, then calculate the variance difference between the two output nodes for the red and blue input patterns. Values for $d \geq 20$ achieve very good classification accuracy in Fig. 4E. These results mean that the covariance estimate can be evaluated with sufficient precision even for a few tens of time points only. 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.

3 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 means). There are two important differences in the present section compared to Section 2. 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 in 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).

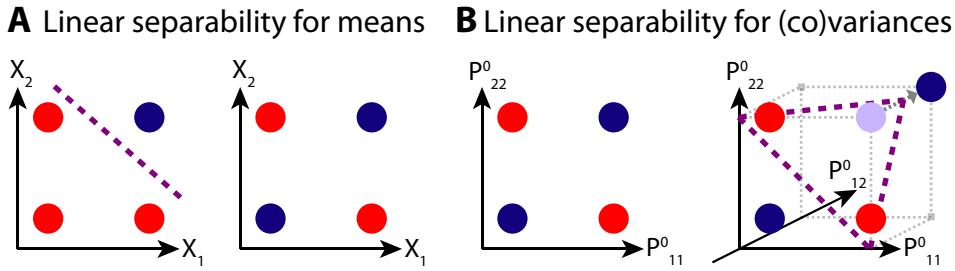

Figure 5: **Comparison between input patterns based on mean and covariance. A:** Two examples of pattern classification for mean-based decoding. In the left diagram, the two categories can be linearly separated, but not in the right diagram. **B:** The left diagram is the equivalent of the right diagram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

3.1 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 means: $m(m+1)/2$ for P^0 including variances compared to m for X. Covariances thus offer a potentially richer 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 dimensionality of the free parameters in the source population is the same in 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 [23].

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}, both small or both large for the blue category, one small and the other large for the red category. Now taking P^0_{11} and P^0_{22} equal to 0 or 1 for small or large values to fix ideas, we have $Q^0_{11} \in \{B^2_{11}, B^2_{12}\}$ for the blue patterns and $Q^0_{11} \in \{0, B^2_{11} + B^2_{12} + 2B_{11}B_{12}P^0_{12}\}$ for red patterns. Provided the red values are smaller than the blue 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 thus gives an additional "degree of freedom" related to P^0_{12} 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 suggests that separability for input covariances may have more flexibility than for input means, due to increased dimensionality.

3.2 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 [12] on the capacity of the mean perceptron.

So far, the weight optimization and classification have been performed in two subsequent steps. After training the connectivity to implement a mapping from given input covariance patterns to two objective covariance patterns, classification is performed by a simple thresholding on the observed matrix element in the output, variance or cross-covariance. We now combine these two procedures into one. The reason is simple: Consider a single entry of the readout covariance matrix Q^0_{ij}. 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 $\hat{Q}^r = Q^0(\hat{P}^r) = B\hat{P}^r B$ following Eq. (5). We are thus demanding less to the individual matrix element in Q^0 than in the previous paradigm: It may live on the entire half-axis, instead of being fixed to a particular value. Note that the numbers of -1 and 1 in ζ^r_{ij} may not be exactly balanced 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 set to 0. It plays an important role for the robustness of the classification [8], 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, not to implement a particular mapping between patterns. This optimization increases the gap and thus the separability between red and green dots in Fig. 6A. In practice, it is simpler to perform this training for a soft-minimum $\hat{\kappa}$, which covaries with the true margin, as shown in Fig. 6B.

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 the random matrix χ^r where diagonal elements are 0 and

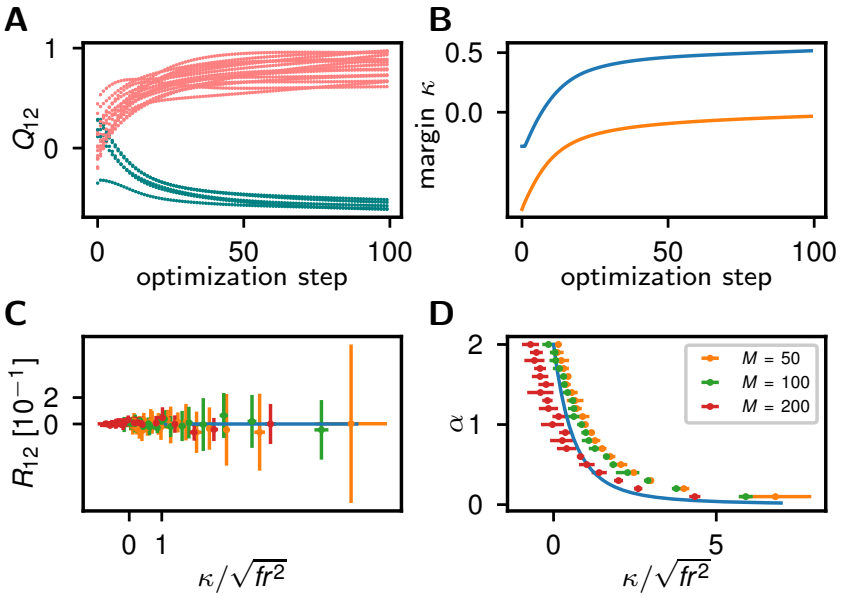

Figure 6: **Memory capacity of the covariance perceptron with a single readout ($n = 2$).** **A:** Evolution of the readout $Q^0_{12} = \hat{Q}^r_{12} = (B\,\hat{P}^p\,B^{\mathrm{T}})_{12}$ over training period, maximizing the soft minimum margin $\hat{\kappa} = -\eta^{-1} \ln \sum_r \exp(-\eta\,\zeta^r_{12}\,\hat{Q}^r_{12})$ by a gradient decent 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 green for $\zeta_{12} = -1$. **B:** Minimum margin over training; blue: minimum margin given by (9); red: soft minimum margin $\hat{\kappa}$. **C:** 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). **D:** 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; $f = 0.2$; $r = 0.5$. Numerical results in C and D from maximization of the soft-minimum margin $\hat{\kappa}$.

off-diagonal elements indexed by (k, l) are independently and identically distributed as $\chi^r_{kl} = 0$ with probability $1-f$ and $\chi^r_{kl} = \pm c$, each with probability $f/2$. 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 lead to correctly classification for all p patterns. This reads for a given margin $\kappa > 0$ and all matrix elements $1 \le i < j \le n$ as

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

The random ensemble for the patterns allows us to employ methods from disordered systems [10]. Closely following the computation for the mean perceptron by Gardner [12, 18], 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 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 readout 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 sending 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 $\langle \ln(\mathcal{V}) \rangle = \lim_{q \to 0} \left(\langle \mathcal{V}^q \rangle - 1 \right)/q$ [10]. 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 for the element $\hat{Q}^{r\alpha}_{ij}$ for a given replicon α. The saddle point equations —cf. Eqs. (56) and (57) in Annex C— admit a vanishing solution $R^{\alpha\beta}_{ij} = 0$ for $i \neq j$. This result is intuitively clear: The two readout 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. 6C shows the overlap observed by numerical optimization as a function of the margin. In the finite-size system studied numerically, a small but nonzero overlap is observed.

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

space of solutions vanishes. Our calculations show that

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

At vanishing margin one obtains $\mathcal{C} = n$. For $n = 2$, a single readout, the capacity is hence identical to the mean perceptron [9]. Moreover, it only depends on the margin through the compound parameter $\bar{\kappa} \equiv \kappa/\sqrt{fc^2}$, relative to the standard deviation of the readout. This dependence on κ is identical for the mean perceptron for which we have $fc^2 = 1$.

The capacity is shown in Fig. 6D 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 hints at an intuitive explanation for the equivalence of both perceptrons. For the case $n = 2$ with two readout vectors and a single label, the problem becomes isotropic in neuron space after the pattern average —cf. Eq. (45) in Annex 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 j'— and their corresponding row vectors in B, namely $\check{B}^{i\mathrm{T}}$, $\check{B}^{j\mathrm{T}}$ and $\check{B}^{j'\mathrm{T}}$. The covariance Q^0_{ij} provides a constraint on $\check{B}^i$. Likewise, the entry $Q^0_{ij''}$ provides a second constraint, potentially contradicting if the readouts must be independent in the sense of coding distinct pattern separations. 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 [18]

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

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

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

bits. The latter expression hence grows $\propto m^3$, while the former only with m^2. If one employs sparse patterns, where $f \propto m^{-1}$, both perceptrons have comparable information content. The dependence on the number of readout neurons n is another linear factor in both cases.

It is worth recalling that the calculations here neglect the constraints that the covariance matrices P must be positive semidefinite. Nonetheless, considering the limit of weak and not too dense entries such that $fc \ll 1$, this requirement is assured by 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.

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

The analysis in the previous section exposed that the capacity of the covariance perceptron is comparable to that of the mean perceptron. To compare and complement the previous results, we use the same optimization as in Figs. 2 and 3, but without additional noise on the presented patterns. We consider (cross-)covariance-based decoding for the network N2 with two output nodes in Fig. 7A. We binarize the output with a threshold function $\theta(Q^0_{12}-0.5)=1$ for $Q^0_{12}>0.5$ and 0 for $Q^0_{12}<0.5$. To incorporate this non-linearity in the gradient descent, we choose as objectives $\bar{Q}^0_{12}\in\{0,1\}$ and redefine the error E in Eq. (25) in Annex B: $E^0=\bar{Q}^0_{12}-\theta(Q^0_{12})$. It follows that $\frac{\partial E}{\partial Q_{12}}$ becomes a matrix full of zeros when the prediction is correct and full of ±1 in the case of erroneous prediction, depending on the category. The error averaged over 50 configurations is displayed in Fig. 7C when increasing the number of covariance patterns similar to the right matrix in Fig. 7B where off-diagonal elements are 0 or $c=1$ (here the matrices are non-negative and required to be positive semidefinite). 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 and to X_1 for mean-based decoding (which is the classical perceptron with binary output) for the network N1 with a single output node in Fig. 7A. 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, which slightly increases for denser covariance matrices in Fig. 7E. Last, Fig. 7F shows that tuning the mapping is robust when increasing the number m of inputs.

4 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 2 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 ($P^0\neq0$ and $P^1=0$) to output spatio-temporal covariances ($Q^0\neq0$ and $Q^1\neq0$), or vice-versa ($P^0\neq0$, $P^1\neq0$, $Q^0\neq0$ and $Q^1=0$). These two examples in Annexes D.2 and D.3 explore the theory for these two cases, relying on the consistency equations Eqs. (23) and (24) derived in Annex A assuming stationarity in place of the real dynamics. Here we consider the configuration in Fig. 1C and verify that the learning procedure tested using theoretical consistency equations also works for simulated time series that differ by their hidden dynamics. This corresponds to the second example where both P^0 and P^1 convey information. To our knowledge the first result for supervised learning that relies on the dynamics in a neural network.

The spatio-temporal structure in our model for inputs is determined by a coupling matrix

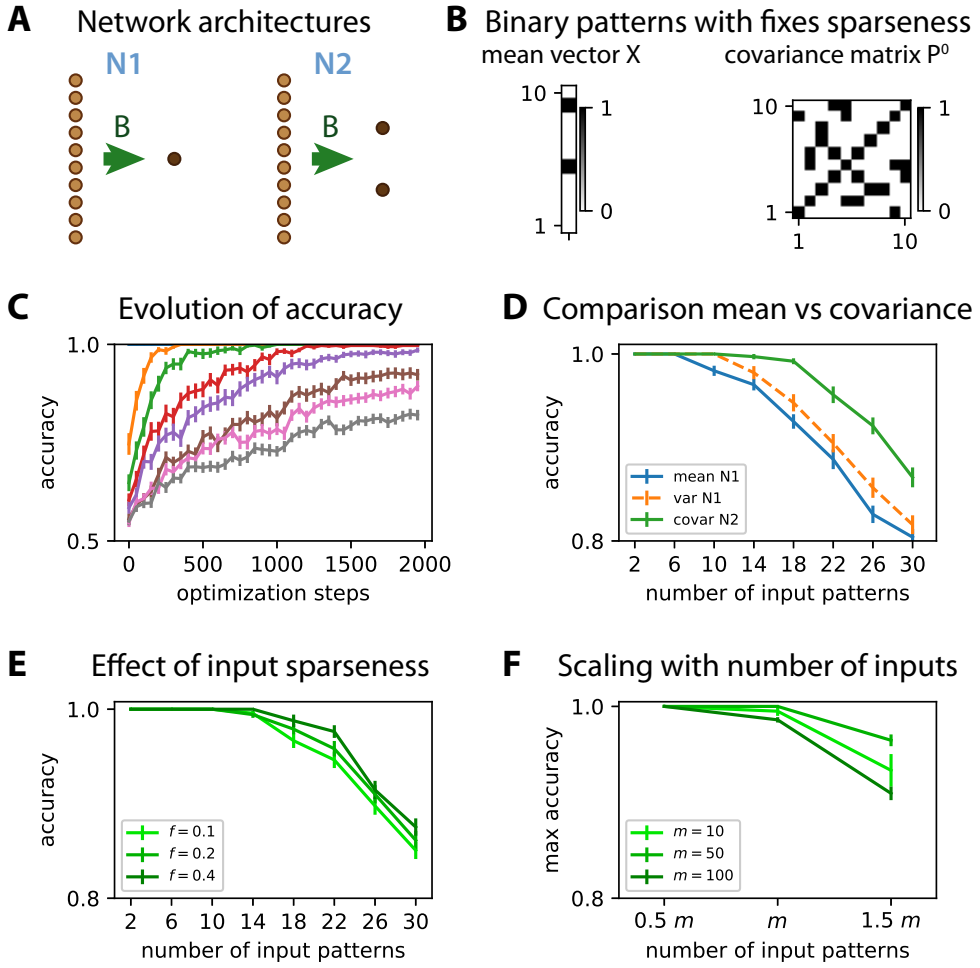

Figure 7: **Numerical evaluation of capacity using offline learning with non-linearity applied to readout. A:** Feedforward networks with afferent connectivity as in Fig. 2A with $n = 1$ and $n = 2$ output nodes, referred to as N1 and N2. **B:** Example mean vector X (left) and covariance matrix P^0 (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.

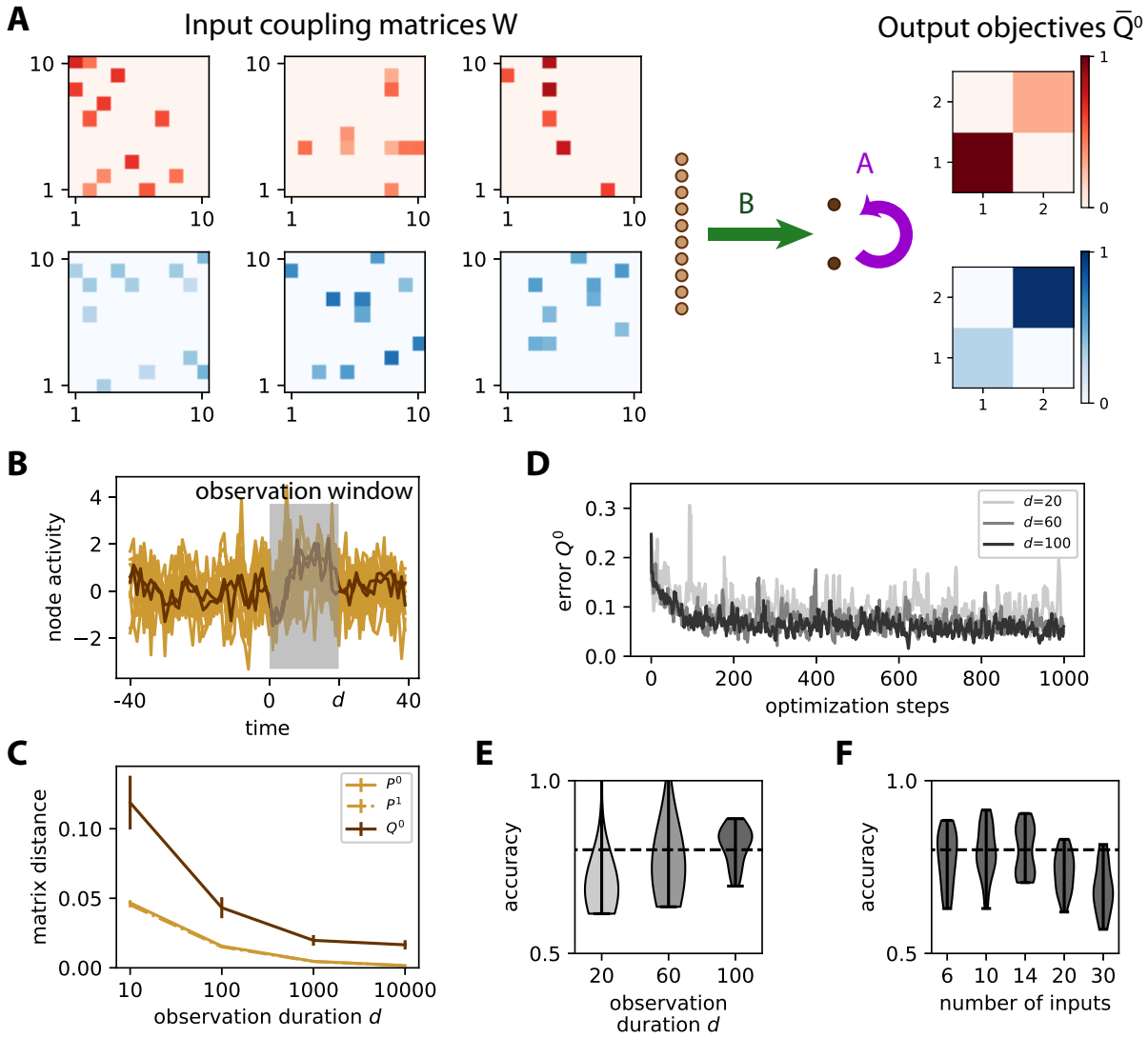

Figure 8: **Online learning for input spatio-temporal covariances with both afferent and recurrent connectivities. A:** The same network as in Fig. S3A to learn the input spatio-temporal covariance structure, which is determined here by a coupling matrix W between the inputs as in Eq. (16). Only 3 examples (left matrices) out of the 5 patterns are displayed for each category. The objective matrices (right) correspond to a specific variance 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 for random connectivities when varying the duration d of the observation window. The error bars indicate the standard error of the mean over 100 noisy versions of the $p = 10$ input patterns, 50 for each category. **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, see the dashed line. The color contrast corresponds to the 3 values for d as in panel D. **F:** Accuracy similar to E for $d = 100$ for 20 configurations when varying the number m of inputs (x-axis). The number of patterns to learn is $p = 10$ for all configurations.

W as a MAR process:

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

with z_k^t being a Gaussian random variable (or white noise with unit variance). This is the equivalent to the graphical model in Eq. (7) for generating temporally correlated input signals. As in Fig. 4, we use 10 patterns for W with 10% density to generate the input time series that the network has to classify based on the output variances, see Fig. 8A where only 3 out of 5 matrices W in each category are displayed. The difference between the input patterns is thus in their hidden dynamics, which the learning has to capture for categorization. Here P^0 satisfies the discrete Lyapunov equation $P^0 = WP^0W^{\mathrm{T}} + \mathbf{1}_m$ and $P^1 = WP^0$, as well as $P^2 = W^2P^0$. The derivation of the consistency equations in Annex A assumes $P^2 = 0$ and is thus an approximation. As the inputs matrix W must have eigenvalues smaller than 1 in modulus for stability purpose, 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. This time simplifying Eq. (70), 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 Annex B:

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

Both formulas correspond to the 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. Note also that Eq. (18) involves the approximation that ignores P^2 and 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. The window duration d affects the precision of the empirical covariances compared to their theoretical counterpart, but the output covariances are much noisier here even for very large d (Fig. 8C), partly due to the approximation mentioned above. This implies that larger d is required to ensure robustness for the learning procedure when training A in addition to B rather than training B alone.

This can be seen in Fig. 8D, where the evolution of the error for the darkest curves with $d \geq 50$ remain lower on average than the the lighter curve with $d = 20$. To assess the quality of the training, we repeat the simulations for 20 network and input configurations, then calculate the variance difference between the two output nodes for the red and blue input patterns. The accuracy gradually improves from $d = 20$ to 100 in Fig. 8F. When varying the number m of inputs while keeping $p = 10$ patterns to learn, the optimization procedure seems to lose robustness for $m \geq 20$. These results contrasts with the robustness and efficiency observed for the feedforward networks trained to detect only spatial covariances and require further study.

5 Discussion

The aim of this paper is the development of a new learning theory for the categorization of time series. Training the weights in the MAR network in a supervised manner, our proposed method extracts the regularities among several input covariance patterns in order to map each category of inputs to a specific output covariance pattern. 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). For the same number of nodes and connections, its capacity per input is theoretically the same as the classical perceptron (Fig. 6). However, the information capacity in bits is one order of magnitude larger because the number of covariance readouts grows quadratically with the number of output neurons. In simulations akin to offline learning, the resulting accuracy of the covariance perceptron compares favorably with the mean perceptron (Fig. 7). Another important result is the demonstration that the covariance perceptron can classify time series with respect to their hidden dynamics (Fig. 8).

The conceptual change of perspective compared to previous studies is that variability in the time series is the basis for the information to be learned, namely the second-order statistics of the co-fluctuating inputs here. Importantly, covariance patterns can involve time lags and are a genuine metric for time series, describing the transitions of activity between nodes. Our theory based on dynamic features contrasts with classical and more "static" approaches based on the mean activity of time series or their binarization, for which the variability is akin to noise. We stress the importance of the results for online learning where cross-validation is performed taking the variability inherent to time series, unlike the test for capacity that rely on noiseless patterns (Fig. 7).

5.1 Covariance-based decoding and representations

The mechanism underlying classification is the linear separability of the input covariance patterns performed by the threshold on the output activity, in the same manner as the classical perceptron for vectors of values. The perceptron is a central concept for classification based on artificial neural networks, from logistic regression [5] to deep learning [21, 30]. The whole output covariance matrix Q^0 can be used as a target, cross-covariances as well as variances In Section 3 the non-linearity on the readout used for classification has been included in the gradient descent, which is straightforward by adapting Eqs. (25) and (27) in Annex B. It remains to be explored which types of non-linearities improve the classification performance —as is well known for the perceptron [23]— or lead to interesting input-output covariance mappings. Nonetheless, our results lay the foundation for covariance perceptron with multiple layers, including with linear feedback in each layer with recurrent connectivity, as its design is consistent with covariance in inputs and 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 / back-propagation algorithm, such as recurrent back-propagation [26] and back-propagation through time [25]. 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 (which implies correlations via the trajectories), the algorithms applied consider the inputs as statistically independent variables. Moreover, unfolding time is 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. A comparison with those approaches for real

data such as movies or sounds is left for future work.

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 involves higher-dimensional spaces in 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 to the vectorized covariances matrices would correspond to $nm(m-1)/2$ weights to tune, to be compared with only nm weights in our study. The present theoretical calculations focus on the capacity of the covariance perceptron for perfect classification (Fig. 6). It uses Gardner's replica method [12] in the thermodynamic limit, toward infinitely many inputs ($m \to \infty$). We have shown that our framework indeed presents an analytically solvable model in this limit and compute the pattern capacity $\mathcal{C} = p/m$ by replica symmetric mean-field theory, analog to the mean perceptron [12]. It turns out that the pattern capacity (per input and output) is exactly identical to that of the mean perceptron once the classification margin fixed. Its information capacity in bits, however, growth with m^3, whereas it only has a dependence as m^2 for the mean perceptron. Both, the pattern capacity and the information capacity, depend linear on the size of the target population n. The latter result is trivial in the case of the mean perceptron —we just have n independent perceptrons here. 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.

The theory to evaluate the capacity for noiseless patterns (Fig. 6) should thus be extended to take into account the observation noise that is inherent to many time series (or models thereof, like the MOU and MAR used here). For such noisy patterns, it appears relevant to evaluate the capacity in the "error regime" [6], which also corresponds to our results using numerical simulation (Fig. 7).

5.2 Learning and functional role for recurrent connectivity

In our theory, recurrent connections do not have the same role as afferent connections that are efficient in separating Q^0 patterns (Figs. 3 and 4). Since even the time series in Fig. 8 differ by their corresponding Q^0, one could argue that afferent connections are sufficient. Indeed, supplementary simulations with tuning only B (not shown) did not show a decrease in the accuracy. A possible role for A is capturing the temporal structure of input covariances or shaping it for output covariances, as studied in Figs. S2 and S3. Other supplementary simulations of Fig. S2 with a specialization for $\bar{Q}^1$ instead of $\bar{Q}^0$ were not successful either. Further study is thus required to better understand the benefit of tuning A. For objectives involving both $\bar{Q}^0$ 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 bring an extra flexibility that broadens the domain of solutions or the stability of learning, although it was not clearly observed so far in our simulations.

On a more technical ground, a positive feature of our learning scheme is its surprising stability (see Annex D.1), even though it should be explored in more depth. The learning equations for A in Annex B are an elaboration of the optimization for recurrent connectivity only in the multivariate Ornstein-Uhlenbeck (MOU) process that we recently proposed [15]. 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 to tune. A natural

gradient descent was used before to train afferent and recurrent connectivity to decorrelate signals and perform blind-source separation [7]. This suggests another possible role for A about the global organization of output nodes, like forming communities of output nodes that are independent of each other (irrespective of the patterns).

5.3 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 requires the derivation of consistency equations for the lagged covariances of MOU process driven by inputs with MOU-like covariances [4]. This is doable, but yields more complicated expressions than for the MAR process.

One can also envisage the following generalization to take into account several types of non-linearity that can arise in recurrently connected networks:

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

with local dynamics determined by ϕ and interactions rectified using 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 Annex A in a range of parameters. The present formalism may thus be extended beyond the non-linearity applied to the readout (Section 3). Note that applying a non-linearity to the dynamics or the output is in fact the same for the mean perceptron, but not for the covariance perceptron.

Another point is that non-linearities involve a cross-talk between statistical orders, meaning that input means may strongly affect output covariances and, conversely, input covariances with output means. This opens the way to mixed decoding paradigms where the relevant information may be distributed in both means and covariances. As the linear system studied here does not exhibit such cross-talk, mixed schemes have been left for future work.

5.4 Learning and (de)coding in biological neuronal networks

An interesting application for the present theory is its adaptation to spiking neuronal networks. In fact, the bio-inspired model of spike-timing-dependent plasticity (STDP) can be expressed on covariances between spike trains [19, 13], which was an inspiration of the present study. STDP-like learning rules were used for object recognition [20] and related to the expectation-maximization algorithm [24]. Although genuine STDP relates to unsupervised learning, extensions were developed to implement supervised learning for spike patterns [17, 27, 16, 11, 32]. 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 [14]. 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 our method can be implemented as an online learning rule. Another important question concerns the locality of the learning rule, which requires pairwise information about neuronal activity.

Studies of noise correlation, which is akin to the variability of spike counts (i.e. mean firing activity), showed that variability is not always an hindrance for decoding [2]. Our study shows another possible role for activity variability and is in line with recent results about stimulus-dependent correlations observed in data [28].

Annex:

A Network dynamics describing activity propagation

Here we recapitulate well-known calculations [22] 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) for stability purpose. 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:

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

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

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

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:

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

and

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

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:

$$\hat{P}^r = 1_m + \chi^r , \quad (35)$$

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

$$\begin{aligned}\chi^r_{kk} &= 0 \\ \chi^r_{k<l} &\equiv \begin{cases} c & \text{with prob. } \frac{1}{2}f \\ -c & \text{with prob. } \frac{1}{2}f \\ 0 & \text{with prob. } 1-f \end{cases} .\end{aligned} \quad (36)$$

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

$$\hat{Q}^r = B\hat{P}^r B^{\mathrm{T}} \ . \tag{37}$$

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 $1 \le i < j \le n$:

$$\begin{aligned} \zeta^r_{i<j}\, \hat{Q}^r_{i<j} &> \kappa \ , \\ \zeta^r_{i<j} &= \begin{cases} 1 & \text{with prob. } \frac{1}{2} \\ -1 & \text{with prob. } \frac{1}{2} \end{cases} , \end{aligned} \tag{38}$$

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. (37) implies

$$\hat{Q}^r_{ii} = \sum_k B^2_{ik} + \sum_{k\neq l} B_{ik}B_{il}\chi^r_{kl} \qquad \forall\, i, r \ . \tag{39}$$

The distribution of the patterns defined by Eq. (36) 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:

$$1 \overset{!}{=} \langle Q^r_{ii} \rangle_\chi = \sum_k B^2_{ik} \qquad \forall\, i \ . \tag{40}$$

This gives another constraint for B, in addition to Eq. (38).

C.1 Gardner's approach to memory capacity

We now define the volume of solutions B whose p mappings in Eq. (37) satisfy the inequalities and the statistics of the output covariance in Eq. (38):

$$\mathcal{V} = \int \mathrm{d}B \prod_i \delta\Big(\sum_k B^2_{ik} - 1\Big) \prod_{r=1}^{p} \prod_{i<j} \theta\Big(\zeta^r_{ij} \left(B\hat{P}^r B^{\mathrm{T}}\right)_{ij} - \kappa\Big) \ . \tag{41}$$

This equation is the analogue to Gardner's approach of the perceptron; see [18, 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 $q \to 0$:

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

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 Eq. (36) and Eq. (38). We rewrite the Heaviside function as

$$\theta\Big(\zeta_{ij}^r \left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - \kappa\Big) = \int_\kappa^\infty \mathrm{d}x_{ij}^\alpha \int_{-\infty}^\infty \frac{\mathrm{d}\tilde{x}_{ij}^\alpha}{2\pi}\, e^{\imath\, \tilde{x}_{ij}^\alpha \left(\zeta_{ij}^r \left(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\right)_{ij} - x_{ij}^\alpha\right)} \tag{43}$$

such that the pattern average

$$\left\langle \prod_{\alpha=1}^q \prod_{r=1}^p \prod_{i<j} \theta\Big(\zeta_{ij}^r \big(B^\alpha \hat{P}^r B^{\alpha\mathrm{T}}\big)_{ij} - \kappa\Big) \right\rangle_{\zeta,\chi} = \prod_{r=1}^p \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^\infty \mathrm{D}\tilde{x}\, e^{\Phi^r(\imath\tilde{x}) - \imath \sum_{\alpha,i,j} \tilde{x}_{ij}^\alpha x_{ij}^\alpha} \tag{44}$$

can be described by a cumulant generating function Φ^r of the variable $Q_{ij}^{r\alpha} \equiv \zeta_{ij}^r (B^\alpha \hat{P}^r B^{\alpha\mathrm{T}})_{ij}$ with respect to the statistics of ζ and χ. The constant $\imath$ is the imaginary unit. Here we have defined the abbreviations $\int_\kappa^\infty Dx \equiv \prod_\alpha \prod_{i<j} \int_\kappa^\infty \mathrm{d}x_{ij}^\alpha$ and $\int_{-\infty}^\infty \mathrm{D}\tilde{x} \equiv \prod_\alpha \prod_{i<j} \int_{-\infty}^\infty \frac{\mathrm{d}\tilde{x}_{ij}^\alpha}{2\pi}$ 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 [18]. As a result, we obtain

$$\begin{aligned} \langle \mathcal{V}^q \rangle_{\zeta,\chi} \;=\; & \prod_{\alpha=1}^q \int \mathrm{d}B^\alpha \prod_i \delta\Big(\sum_k B_{ik}^{\alpha 2} - 1\Big) \\ & \times \prod_{r=1}^p \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^\infty \mathrm{D}\tilde{x} \prod_{i<j} e^{-\imath \sum_\alpha \tilde{x}_{ij}^\alpha x_{ij}'^\alpha - \frac{1}{2}\sum_{\alpha,\beta} \tilde{x}_{ij}^\alpha \left\langle\left\langle Q_{ij}^{r\alpha} Q_{ij}^{r\beta}\right\rangle\right\rangle \tilde{x}_{ij}^\beta} \end{aligned} \tag{45}$$

with

$$\begin{aligned} \left\langle\left\langle Q_{ij}^{r\alpha} Q_{ij}^{r\beta}\right\rangle\right\rangle \;=\; & \Big(\sum_k B_{ik}^\alpha B_{jk}^\alpha\Big)\Big(\sum_k B_{ik}^\beta B_{jk}^\beta\Big) \\ & + fc^2 \Big(\sum_k B_{ik}^\alpha B_{ik}^\beta\Big)\Big(\sum_k B_{jk}^\alpha B_{jk}^\beta\Big) \\ & + fc^2 \Big(\sum_k B_{ik}^\alpha B_{jk}^\beta\Big)\Big(\sum_k B_{ik}^\beta B_{jk}^\alpha\Big)\,. \end{aligned} \tag{46}$$

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. (45), we now define the auxiliary fields as

$$R_{ij}^{\alpha\beta} \equiv \sum_k B_{ik}^\alpha B_{jk}^\beta\,, \tag{47}$$

for $i < j$ and $\alpha \neq \beta$. For $\alpha = \beta$ and $i = j$ we have $R_{ii}^{\alpha\alpha} = 1$ due to Eq. (40). 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. (45) as

$$\begin{aligned}\langle \mathcal{V}^q \rangle &= \int \mathrm{d}R \int \mathrm{D}\check{B} \prod_{\alpha,\beta} \prod_{i\leq j} \delta\big(\check{B}_i^{\alpha\mathrm{T}} \check{B}_j^\beta - R_{ij}^{\alpha\beta}\big)\, \mathcal{G}_{ij}^p \qquad (48)\\ \mathcal{G}_{ij} &= \int_\kappa^\infty \mathrm{D}x \int_{-\infty}^\infty \mathrm{D}\tilde{x}\, e^{-\frac{1}{2}\sum_{\alpha,\beta}\tilde{x}^\alpha \left(R_{ij}^{\alpha\alpha} R_{ij}^{\beta\beta} + fc^2\, R_{ii}^{\alpha\beta}\, R_{jj}^{\alpha\beta} + fc^2\, R_{ij}^{\alpha\beta}\, R_{ij}^{\beta\alpha}\right)\tilde{x}^\beta - \imath \sum_\alpha \tilde{x}^\alpha x'^\alpha},\end{aligned}$$

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

$$\delta(\check{B}^{i\alpha\mathrm{T}}\check{B}^{i\alpha} - 1) = \frac{1}{2\pi\imath}\int_{-i\infty}^{i\infty} \mathrm{d}\tilde{R}_{ii}^{\alpha\alpha}\, \exp\big(-\tilde{R}_{ii}^{\alpha\alpha}\,(\check{B}^{i\alpha\mathrm{T}}\check{B}^{i\alpha} - 1)\big). \qquad (49)$$

Analogously, we employ this Fourier representation of the Dirac δ to express the constraints defining the auxiliary fields Eq. (47) to obtain

$$\begin{aligned}\langle \mathcal{V}^q \rangle &= \int \mathrm{d}R \int \mathrm{d}\tilde{R}\, \exp(\mathcal{S}) \qquad (50)\\ \mathcal{S} &= m\,\ln(F) + \sum_{i<j} p \ln(\mathcal{G}_{ij}) + \mathcal{H} \qquad (51)\\ \mathcal{F} &= \int \mathrm{d}\check{B}\, \exp\Big(-\sum_{\alpha,\beta,i\leq j} \tilde{R}_{ij}^{\alpha\beta}\check{B}_i^\alpha \check{B}_j^\beta\Big) \qquad (52)\\ \mathcal{H} &= \sum_{\alpha,\beta,i\leq j} \tilde{R}_{ij}^{\alpha\beta} R_{ij}^{\alpha\beta} \qquad (53)\end{aligned}$$

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

$$\begin{aligned} R_{ij}^{\alpha\alpha} &= R_{ij}^{=}, \qquad \tilde{R}_{ij}^{\alpha\alpha} = \tilde{R}_{ij}^{=} \qquad (54)\\ R_{ij}^{\alpha\beta} &= R_{ij}^{\neq}, \qquad \tilde{R}_{ij}^{\alpha\beta} = \tilde{R}_{ij}^{\neq}\end{aligned}$$

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

$$\mathcal{H} = q\sum_{i\leq j} \tilde{R}_{ij}^{=} R_{ij}^{=} - q\sum_{i\leq j}\tilde{R}_{ij}^{\neq} R_{ij}^{\neq} \qquad (55)$$

which gives rise to the following saddle point equations

$$\begin{aligned} R_{ij}^{=} &= -\frac{m}{q}\frac{\partial \ln(\mathcal{F})}{\partial \tilde{R}_{ij}^{=}}\,, \quad R_{ij}^{\neq} = \frac{m}{q}\frac{\partial \ln(\mathcal{F})}{\partial \tilde{R}_{ij}^{\neq}} \qquad (56)\\ \tilde{R}_{ij}^{=} &= -\frac{p}{q}\sum_{k<l}\frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{=}}\,, \quad \tilde{R}_{ij}^{\neq} = \frac{p}{q}\sum_{k<l}\frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{\neq}} \qquad (57)\end{aligned}$$

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

$$\sum_{\alpha,\beta} \tilde{x}^\alpha \Big(R_{ij}^{\alpha\alpha} R_{ij}^{\beta\beta} + fc^2\, R_{ii}^{\alpha\beta}\, R_{jj}^{\alpha\beta} + fc^2\, R_{ij}^{\alpha\beta}\, R_{ij}^{\beta\alpha} \Big) \tilde{x}^\beta = (\lambda_{ij}^{=} - \lambda_{ij}^{\neq}) \sum_\alpha \tilde{x}^\alpha \tilde{x}^\alpha + \lambda_{ij}^{\neq} \left(\sum_\alpha \tilde{x}^\alpha \right)^2 , \tag{58}$$

with $\lambda_{ij}^{=} = fc^2\, R_{ii}^{=} R_{jj}^{=} + (1 + fc^2)\, R_{ij}^{=2}$ and $\lambda_{ij}^{\neq} = fc^2\, R_{ii}^{\neq} R_{jj}^{\neq} + R_{ij}^{=2} + fr^2\, R_{ij}^{\neq 2}$. 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

$$\exp\Big(-\frac{1}{2} \lambda_{ij}^{\neq} \Big(\sum_\alpha \tilde{x}^\alpha \Big)^2 \Big) = \int_{-\infty}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\Big(-t^2/2 + it \sqrt{\lambda_{ij}^{\neq}} \sum_\alpha \tilde{x}^\alpha \Big) , \tag{59}$$

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

$$g_{ij}(t) = \int_\kappa^\infty \mathrm{d}x \int_{-\infty}^{\infty} \frac{\mathrm{d}\tilde{x}}{2\pi} \exp\Big(-\frac{1}{2} (\lambda_{ij}^{=} - \lambda_{ij}^{\neq}) \tilde{x}^2 + it \sqrt{\lambda_{ij}^{\neq}} \tilde{x} - i\tilde{x}x \Big) = \frac{1}{2} \mathrm{erfc}(a_{ij}(t)) , \tag{60}$$

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

$$\begin{aligned} \ln(\mathcal{G}_{ij}) &= \ln \langle g_{ij}(t)^q \rangle = \ln \big\langle \exp\big(q \ln\big(g_{ij}(t) \big) \big) \big\rangle \\ &\to \ln\big(1 + q \big\langle \ln\big(g_{ij}(t) \big) \big\rangle \big) \to q \big\langle \ln\big(g_{ij}(t) \big) \big\rangle \\ &= q \Big\langle \ln\Big(\mathrm{erfc}\big(a_{ij}(t) \big) \Big) \Big\rangle + \ln(1/2) . \end{aligned} \tag{61}$$

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. (48)). By Taylor expansion of Eq. (52) 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. (57) as $\tilde{R}_{ij}^{=} = -2R_{ij}^{=} \frac{p}{n} \sum_{k<l} \frac{\partial \ln(\mathcal{G}_{kl})}{\partial R_{ij}^{=2}}$ and $R_{ij}^{\neq} = -2\tilde{R}_{ij} \frac{m}{n} \frac{\partial \ln(\mathcal{F})}{\partial \tilde{U}_{ij}^2}$, 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

$$\ln(\mathcal{F}) \to -\frac{1}{2} q \sum_i \left(\ln\left(\tilde{R}_{ii}^{=} - \tilde{R}_{ii}^{\neq} \right) + \tilde{R}_{ii}^{\neq} / \left(\tilde{R}_{ii}^{=} - \tilde{R}_{ii}^{\neq} \right) \right) + \text{const} \ . \tag{62}$$

Then Eq. (56) can be easily solved to obtain

$$\tilde{R}_{ii}^{=} = -\frac{m}{2} \frac{1 - 2\epsilon_i}{\epsilon_i^2}, \quad \tilde{R}_{ii}^{\neq} = -\frac{m}{2} \frac{1 - \epsilon_i}{\epsilon_i^2} \ . \tag{63}$$

Inserting the solution Eq. (63) into Eq. (57) and using Eq. (61), we get in the limit $\epsilon_i \to 0$

$$\begin{aligned}
-\frac{m}{2} \frac{1}{\epsilon_i^2} &= p \sum_{k<l} \left(\frac{\partial \left\langle \ln\left(\operatorname{erfc}(a_{kl}(t)) \right) \right\rangle}{\partial R_{kk}^{\neq}} \delta_{ki} + \frac{\partial \left\langle \ln\left(\operatorname{erfc}(a_{kl}(t)) \right) \right\rangle}{\partial R_{ll}^{\neq}} \delta_{li} \right) && (64) \\
&= p \sum_{k<l} \int_{-\infty}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\left(-t^2/2\right) \frac{\frac{\partial}{\partial a_{kl}(t)} \operatorname{erfc}(a_{kl}(t))}{\operatorname{erfc}(a_{kl}(t))} \left(\frac{\partial a_{kl}(t)}{\partial R_{kk}^{\neq}} \delta_{ki} + \frac{\partial a_{kl}(t)}{\partial R_{ll}^{\neq}} \delta_{li} \right) && (65) \\
&= p \sum_{k<l} \int_{-\infty}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\left(-t^2/2\right) \frac{-\frac{2}{\sqrt{\pi}} e^{-a_{kl}(t)^2}}{\operatorname{erfc}(a_{kl}(t))} \left(\frac{\partial a_{kl}(t)}{\partial R_{kk}^{\neq}} \delta_{ki} + \frac{\partial a_{kl}(t)}{\partial R_{ll}^{\neq}} \delta_{li} \right)
\end{aligned}$$

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

$$\frac{\partial a_{kl}(t)^2}{\partial R_{kk}^{\neq}} \to \frac{(\bar{\kappa} - t)^2}{2\epsilon_k^2}, \tag{66}$$

the limiting capacity follows from

$$\frac{m}{\epsilon_i^2} = p \int_{-\infty}^{\bar{\kappa}} \frac{dt}{\sqrt{2\pi}} \exp\left(-t^2/2\right) \frac{(\bar{\kappa} - t)^2}{\epsilon_i^2} \sum_{k<l} (\delta_{ki} + \delta_{li}) \tag{67}$$

as

$$\mathcal{C} \equiv \frac{n(n-1)}{2} \frac{p}{m} = \frac{n}{2} \left(\int_{-\bar{\kappa}}^{\infty} \frac{dt}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2} \right) (t + \bar{\kappa})^2 \right)^{-1} \ . \tag{68}$$

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

$$\mathcal{C} = n \ . \tag{69}$$

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

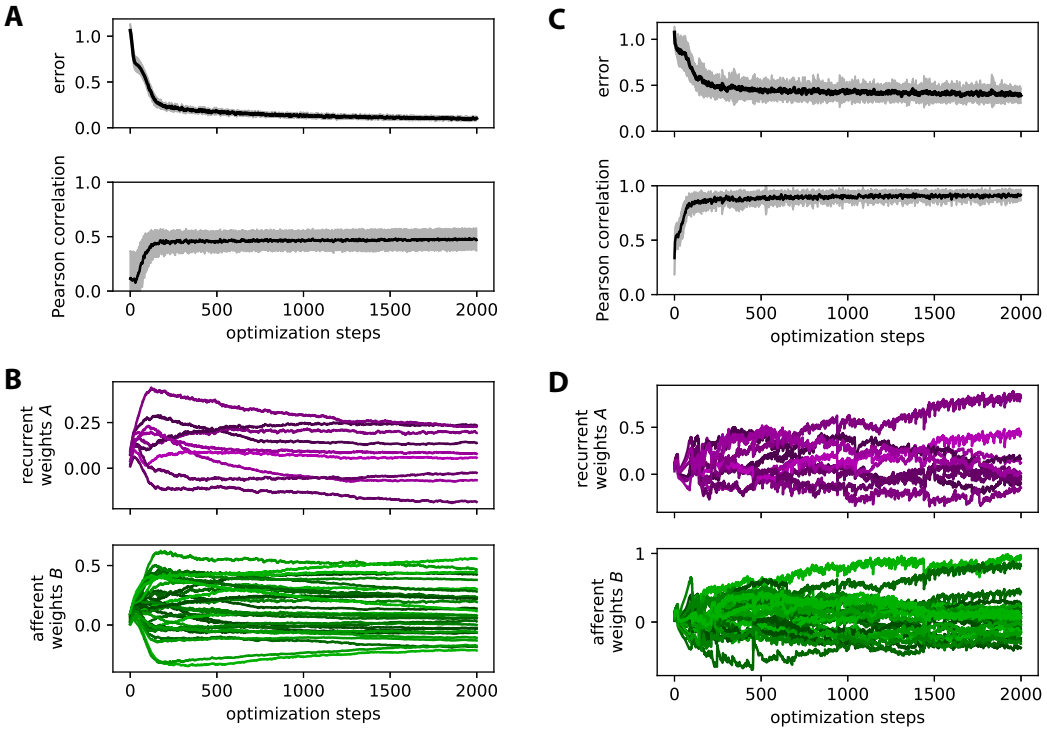

Figure S1: **Stability of ongoing learning. A:** Evolution of the error for 20 optimizations of networks with $m = 10$ inputs to the $n = 3$ outputs. Here the objective output covariance matrix $\bar{Q}^0$ is random and 10 random input patterns P^0 and P^1, but no tuning for Q^1 is performed. The plot 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 according to the MAR procedure in 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.

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:

$$\begin{aligned} \Delta B_{ik} &= \eta_B \left[\left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial B_{ik}} + \left(\bar{Q}^1 - Q^1\right) \odot \frac{\partial Q^1}{\partial B_{ik}} \right] , \\ \Delta A_{ij} &= \eta_A \left[\left(\bar{Q}^0 - Q^0\right) \odot \frac{\partial Q^0}{\partial A_{ij}} + \left(\bar{Q}^1 - Q^1\right) \odot \frac{\partial Q^1}{\partial A_{ij}} \right] . \end{aligned} \tag{70}$$

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:

$$\begin{aligned} \frac{\partial Q^0}{\partial B_{ik}} &= A\frac{\partial Q^0}{\partial B_{ik}}A^{\mathrm{T}} + U^{ik}P^0B^{\mathrm{T}} + BP^0U^{ik\mathrm{T}} , \\ \frac{\partial Q^1}{\partial B_{ik}} &= A\frac{\partial Q^1}{\partial B_{ik}}A^{\mathrm{T}} + AU^{ik}P^0B^{\mathrm{T}} + ABP^0U^{ik\mathrm{T}} , \\ \frac{\partial Q^0}{\partial A_{ij}} &= A\frac{\partial Q^0}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^0A^{\mathrm{T}} + AQ^0V^{ij\mathrm{T}} , \\ \frac{\partial Q^1}{\partial A_{ij}} &= A\frac{\partial Q^1}{\partial A_{ij}}A^{\mathrm{T}} + V^{ij}Q^1A^{\mathrm{T}} + AQ^1V^{ij\mathrm{T}} + V^{ij}BP^0B^{\mathrm{T}} . \end{aligned} \tag{71}$$

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 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.

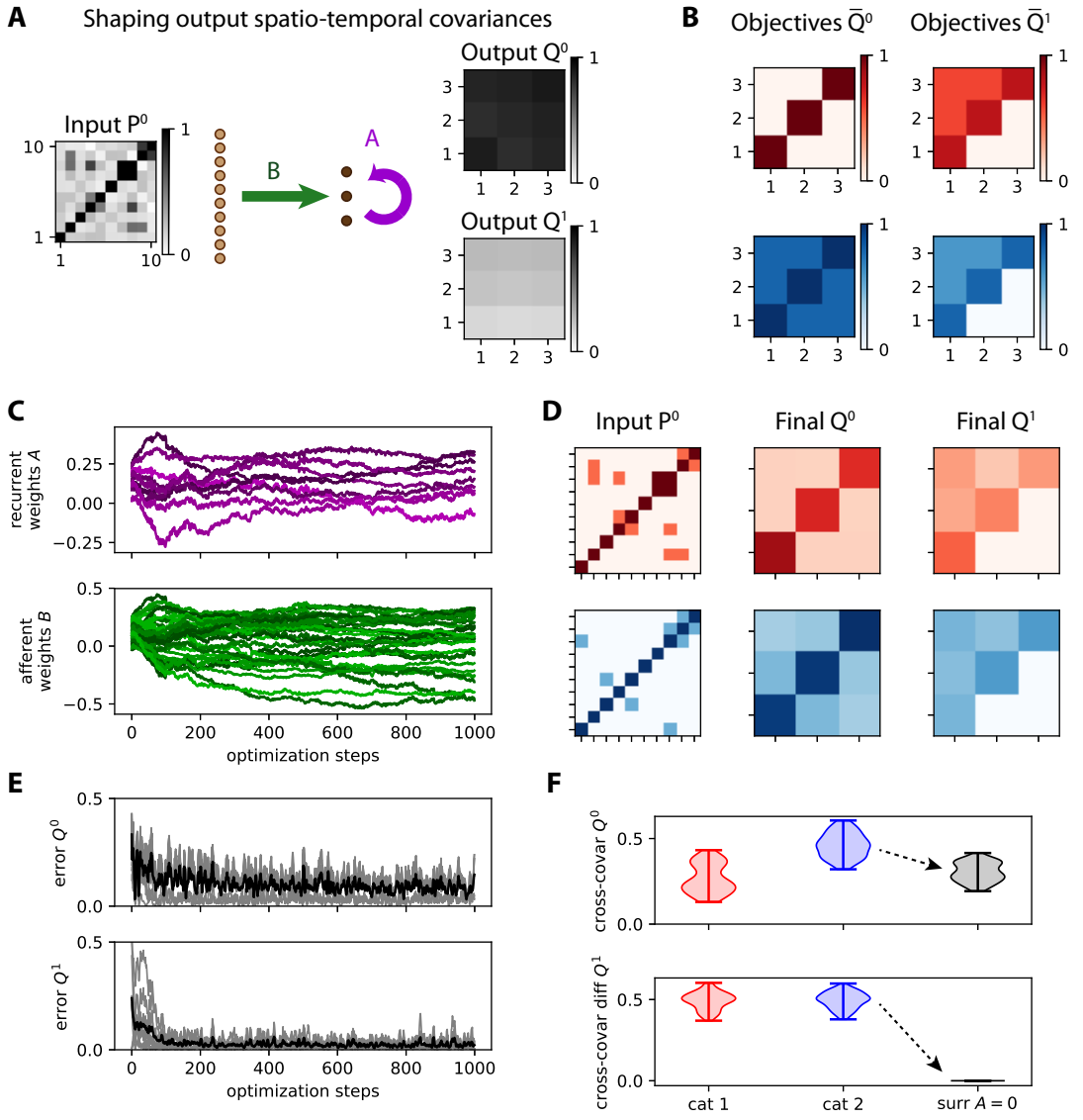

Figure S2: **Shaping output spatio-temporal covariances with both afferent and recurrent connectivities. A:** Network architecture with $m = 10$ input nodes and $n = 3$ output nodes, the latter being connected together by the recurrent weights A (purple arrow). **B:** Objective covariance matrices for 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.

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

[3] O. Barak and M. Rigotti. A simple derivation of a bound on the perceptron margin using singular value decomposition. *Neural Computation*, 23:1935–1943, 2011.

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

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

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

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

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

[9] 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.

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

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

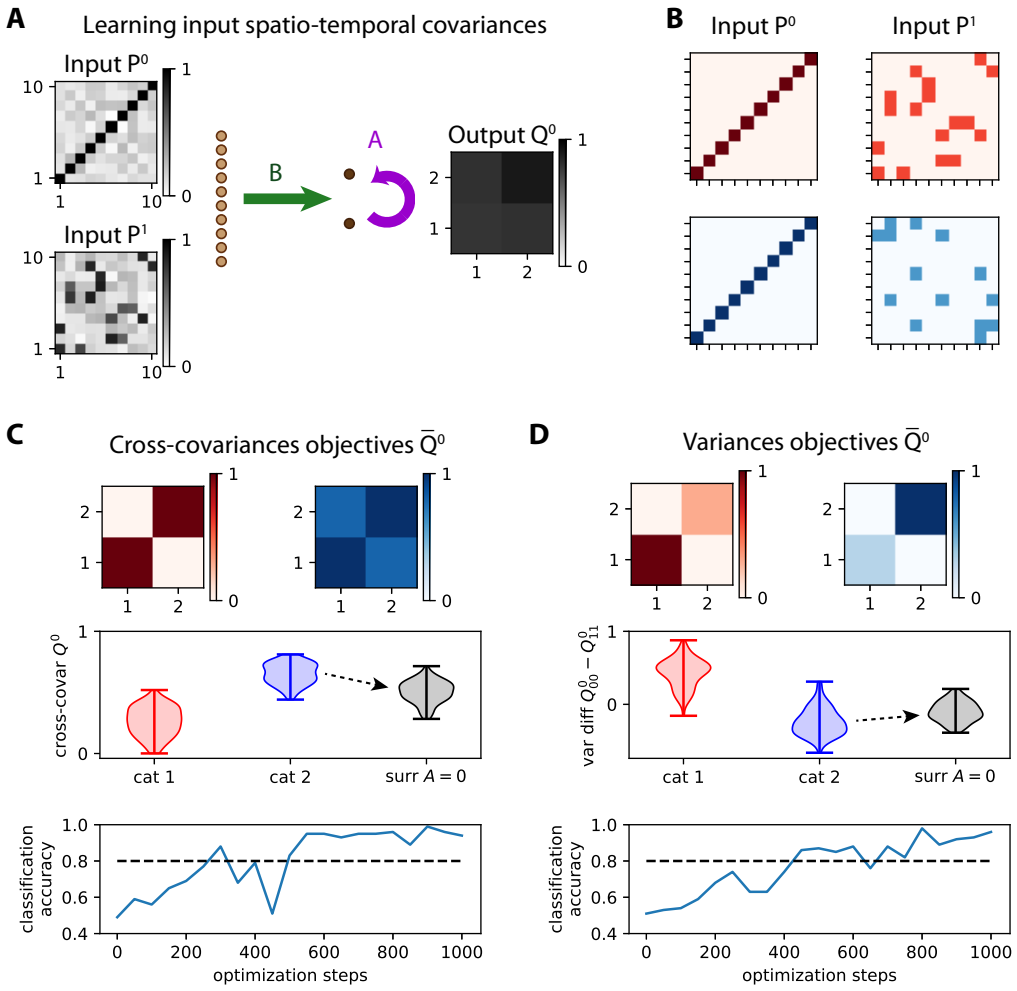

Figure S3: **Learning input spatio-temporal covariances with both afferent and recurrent connectivities. A:** Similar network to Fig. S2A with $m = 10$ input nodes and $n = 2$ output nodes. **B:** Two examples of input patterns corresponding to a pair P^0 and P^1, among the 5 in each category. The P^0 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 threshold 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 threshold as in Fig. 3A-B.

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

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

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

[15] 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.

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

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

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

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

[20] 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.

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

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

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

[24] 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.

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

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

[27] 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.

[28] 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.

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

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

[31] T. Shinzato and Y. Kabashima. Perceptron capacity revisited: classification ability for correlated patterns. *Journal of Physics A: Mathematical and Theoretical*, 41(32):324013, 2008.

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