scientific reports

OPEN

Probing the structure–function relationship with neural networks constructed by solving a system of linear equations

Camilo J. Mininni[1,2]✉ & B. Silvano Zanutto[1,2]

Neural network models are an invaluable tool to understand brain function since they allow us to connect the cellular and circuit levels with behaviour. Neural networks usually comprise a huge number of parameters, which must be chosen carefully such that networks reproduce anatomical, behavioural, and neurophysiological data. These parameters are usually fitted with off-the-shelf optimization algorithms that iteratively change network parameters and simulate the network to evaluate its performance and improve fitting. Here we propose to invert the fitting process by proceeding from the network dynamics towards network parameters. Firing state transitions are chosen according to the transition graph associated with the solution of a task. Then, a system of linear equations is constructed from the network firing states and membrane potentials, in a way that guarantees the consistency of the system. This allows us to uncouple the dynamical features of the model, like its neurons firing rate and correlation, from the structural features, and the task-solving algorithm implemented by the network. We employed our method to probe the structure–function relationship in a sequence memory task. The networks obtained showed connectivity and firing statistics that recapitulated experimental observations. We argue that the proposed method is a complementary and needed alternative to the way neural networks are constructed to model brain function.

Understanding brain function requires the construction of models that explain experimental data, which encompass behavioural outcome, anatomical features, neurons biophysics, and coding properties, among others[1,2]. Many kinds of models have been proposed along history. Among them, neural network models are well poised to connect all levels of analysis, from the behavioural to the molecular level, being a natural choice as neurons are the functional units of the brain. Yet, constructing suitable neural network models is not an easy task. Their parameters can be defined according to experimental data, or randomly chosen when the data is not available. However, this approach may fall short given the complexity of the nervous systems. To tackle this issue, theorists have employed optimization algorithms that adjust network parameters in a direction that minimizes a loss function. Conversely, the loss function is constructed in such a way that it is minimized when the network satisfies the desired experimental observations that we are seeking to explain, like an animal's performance in one or several tasks[3], connectivity constraints such as Dale's principle[4], or connectivity with a certain degree of sparseness[5]. Optimization methods are widely used in artificial intelligence (AI), and the ongoing deep learning revolution has prompted an explosion of fitting algorithms, and the eagerness to take advantage of them to build models of brain function[6,7]. However, AI needs are different from the theoretical neuroscience needs. Artificial intelligence deals with the construction of systems capable of solving difficult tasks, employing very general optimization algorithms for parameter fitting[8]. On the other hand, models in neuroscience are expected to explain how animals behave in simple tasks, yet with biologically plausible neural networks. Simple tasks are desired because behavioural outcomes are easier to interpret, and mechanistic explanations easier to envisage. Thus, in AI the difficulty resides in the task, while in theoretical neuroscience it lies in the restrictions in network design that are imposed by biology. Therefore, methods for parameter fitting in theoretical neuroscience can take advantage of this point – the simplicity of the task – to solve problems that could be too hard to solve with generic optimization algorithms.

[1]Consejo Nacional de Investigaciones Científicas y Técnicas, Instituto de Biología y Medicina Experimental, Buenos Aires, Argentina. [2]Universidad de Buenos Aires, Facultad de Ingeniería, Instituto de Ingeniería Biomédica, Buenos Aires, Argentina. ✉email: cmininni@fi.uba.ar

One approach that has been overlooked consists in finding the synaptic weights of a network as the solution of a system of equations. For many commonly employed neural network models, neurons perform a weighted sum of their inputs, followed by a non-linear transformation. For this kind of model, the synaptic weights can readily be found by solving a linear system of equations in which the neurons' firings constitute the coefficient matrix, and the added postsynaptic potentials are the dependent variables. Thus, the problem of finding the network parameters translates into finding a suitable *dynamic* that the network should follow when solving a behavioural task of interest. Although this problem might seem as hard as the former, we show in this work that viable network dynamics can easily be found from the graph of transitions associated with the solution of the task. By doing so, we were able to construct networks with millions of parameters extremely fast, without inefficient searches in parameter space. Moreover, optimization algorithms may have biases for a subset of all possible solutions[9]. These biases depend on the algorithms employed, the hyperparameters, and the regularizations, and the relation between biases and their causes might be difficult to understand or control[10]. In contrast, our method takes samples from a distribution of networks with a desired dynamic, while structural constraints can be imposed in a subsequent step. Since the algorithm proceeds from the network firing states to the network parameters, we call it the *Firing to Parameter* (FTP) algorithm.

The results are organized as follows: first, we describe the key aspects of the FTP algorithm. Second, we employ FTP to construct networks of binary neurons that solve a sequence memory task. We then show how to use the algorithm to obtain networks of desired firing rate and pairwise correlation, and how to incorporate structural constraints, such as Dale's principle and sparse connectivity. We finally demonstrate how the FTP can be exploited to construct specific null models to test hypotheses regarding the relationship between network structure and function. The networks obtained displayed firing and connectivity features that resemble those observed in neurophysiological experiments.

Results

Constructing neural networks with predefined dynamics.

We will consider a network of N_{rec} recurrently connected binary neurons (the *integration* neurons), which receive information about the environment from a set of N_{in} input neurons (Fig. 1a). The temporal evolution of the network is dictated by the standard equations of the linear-threshold neuron model:

$$\mathbf{u}(t) = \mathbf{y}(t)\mathbf{W}^{in} + \mathbf{z}(t-1)\mathbf{W}^{rec} \tag{1}$$

$$\mathbf{z}(t) = \mathrm{H}(\mathbf{u}(t) - \boldsymbol{\theta}) \tag{2}$$

where $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$ are synaptic weights matrices of input and integration neurons, vector $\mathbf{y}$ contains firing states of input neurons, which codify the stimuli presented, $\boldsymbol{\theta}$ is a vector of neuron thresholds, and H stands for the Heaviside function. Vector $\mathbf{u}$ is a real-valued vector of length N_{rec} that collects the *network activation states*, akin to membrane potentials, and $\mathbf{z}$ is a vector of zeros and ones that collects the *network firing states*. At any given time t the recurrent network can adopt one out of M network firing states m, defined by vector $\mathbf{z}_m$.

Equation (1) shows how firing states and activation states are linearly related through the synaptic weights. Thus, the weight matrices can be computed exactly by solving a linear system of equations, provided that firing states and the resulting activations are known. To specify, we define $\mathbf{c}_{s,m} = [\mathbf{y}_s \;\; \mathbf{z}_m]$, the concatenation of the input firing state during presentation of stimulus s, and the network firing state when the network is at state m. Next, we construct a matrix $\mathbf{C}$ whose rows are vectors $\mathbf{c}_{s,m}$ for all the combinations of stimuli and network states we want to include in the proposed dynamic. Then, the following system of linear equations follows:

$$\mathbf{CW} = \mathbf{U} \tag{3}$$

where matrix $\mathbf{W} = [\mathbf{W}^{in} \;\; \mathbf{W}^{rec}]$ is the concatenation of the input and recurrent synaptic weights. Here the i^{th} row in $\mathbf{U}$ is the activation state $\mathbf{u}(t)$ when the i^{th} row of $\mathbf{C}$ is $[\mathbf{y}(t) \;\; \mathbf{z}(t-1)]$. Then, $\mathbf{W}$ can be computed as:

$$\mathbf{W} = \mathbf{C}^{+}\mathbf{U} \tag{4}$$

where $\mathbf{C}^{+}$ stands for the pseudoinverse of $\mathbf{C}$. We employed the pseudoinverse because it gives the solution that minimizes the Frobenius norm[11]. In this manner, synaptic weights are going to be small in absolute value, which is desirable since biological synapses are constrained in the number of receptors and vesicles.

Matrices $\mathbf{C}$ and $\mathbf{U}$ should be picked in such a way that they instantiate the state transitions exhibited by a network while solving a target task, e.g., transitions to different network states after presentation of different stimuli in a discrimination task. Simple tasks like the ones employed in behavioural neuroscience exhibit simple state transitions, hence the transition graph is known. Therefore, it only remains to find the actual vectors $\mathbf{u}$ and $\mathbf{c}$. A naïve approximation to this problem would be to pick vectors $\mathbf{u}$ at random, apply the thresholds to obtain the associated vectors $\mathbf{z}$, and construct matrices $\mathbf{C}$ and $\mathbf{U}$ by following the desired transition graph. However, by doing so we likely end up having an inconsistent system of equations, meaning that no network of neurons can follow those state transitions. To understand this important point, we may consider the case of two stimuli s_1 and s_2, codified by input neurons with firing states $\mathbf{y}_1$ and $\mathbf{y}_2$ respectively. Then, each vector $[\mathbf{y}_1 \;\; \mathbf{z}_m]$ can be expressed as a linear combination of $[\mathbf{y}_2 \;\; \mathbf{z}_m]$ and vectors $[\mathbf{y}_1 \;\; \mathbf{z}_P]$ and $[\mathbf{y}_2 \;\; \mathbf{z}_P]$, where vector $\mathbf{z}_P$ can be any vector taken from the set of all firing states the network can adopt:

$$[\mathbf{y}_2 \;\; \mathbf{z}_m] = [\mathbf{y}_1 \;\; \mathbf{z}_m] - [\mathbf{y}_1 \;\; \mathbf{z}_P] + [\mathbf{y}_2 \;\; \mathbf{z}_P] \tag{5}$$

Thus, $\mathrm{rank}(\mathbf{C}) = M + 1$. Following the Rouché-Capelli theorem[12], Eq. (3) has a solution if and only if $\mathrm{rank}(\mathbf{C}) = \mathrm{rank}([\mathbf{C} \;\; \mathbf{U}])$, being $[\mathbf{C} \;\; \mathbf{U}]$ the augmented matrix. Yet, if we choose the rows of $\mathbf{U}$ randomly, when

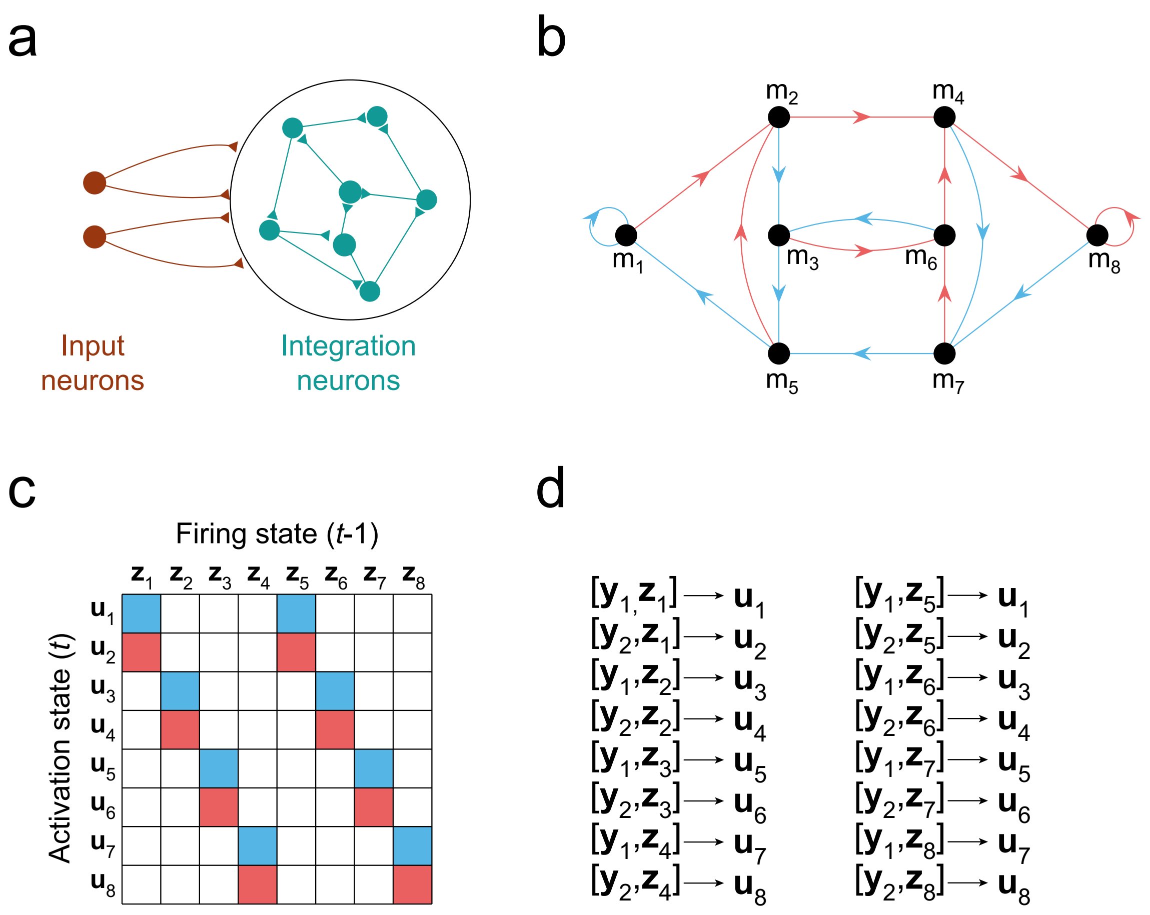


Figure 1. Constructing recurrent networks that follow a predefined transition graph. (**a**) Networks are composed of binary neurons. Input neurons codify stimuli and project to the integration neurons through synaptic weights $\mathbf{W}^{in}$. Integration neurons are recurrently connected through synaptic weights $\mathbf{W}^{rec}$. (**b**) Transition graph showing transitions between network states during execution of the s-task, for $\tau = 3$. Each node in the graph is a network state, and the directed edges depict transitions between states after stimuli presentation (blue for s_1, red for s_2). Each possible sequence of 3 stimuli is codified by exactly one network state. Nodes are numbered such that transitions can be represented in a simple transition matrix. (**c**) Transition matrix associated with the transition graph in panel (**b**). It shows the activation states **u** that are reached when integration neurons are in a population firing state **z**, and s_1 (blue) or s_2 (red) are presented. (**d**) Same transitions depicted in panels (**b**) and (**c**), but explicitly showing vectors $\mathbf{u}_i$ and vectors $\mathbf{c}_i$, which are the concatenation of one **y** and one **z**. The index i is such that $\mathbf{z}_i$ and $\mathbf{u}_i$ are the firing state and activation vectors corresponding to network state m_i.

adjoined to matrix **C** the linear dependencies expressed in Eq. (5) will be broken, and the resulting augmented matrix will have rank above $M+1$. However, if the rows of **U** are linearly combined following the linear combinations present in **C**, the resulting system of equations will be consistent, and its solution will retrieve matrix **W**. This is the key aspect of the FTP algorithm, which can be succinctly stated as:

1. Find the transition graph between network states that solve the target task.
2. Choose activation states **u** from a desired distribution.
3. Use vectors **u** and associated vectors **c** to construct matrices **U** and **C** following the transition graph.
4. Linearly combine rows in **U** to preserve the linear combinations among rows of **C** (Eq. 5)

Testing the FTP algorithm in a sequence memory task. In the following we exemplify the utility of the FTP algorithm by constructing networks that solve a sequence memory task (s-task). In this task two stimuli s_1 and s_2, codified by input neurons firing states $\mathbf{y}_1$ and $\mathbf{y}_2$, are sequentially presented at each time step, chosen randomly with equal probability. To obtain reward at time step t the network has to recall the stimulus presented at time step $t - \Delta t$. By recall we mean that the stimulus at time $t - \Delta t$ is univocally codified by the network firing state at time t. Consequently, successful behaviour requires to have a memory of stimuli sequences of length $\tau = \Delta t + 1$. We chose this task because its complexity grows exponentially with τ, making it a good benchmark to test the computational efficiency of the algorithm. It has also been shown that neural populations are sensitive to stimulus presentation ordering, and the coding of such sequences is incompletely understood[13]. Thus,

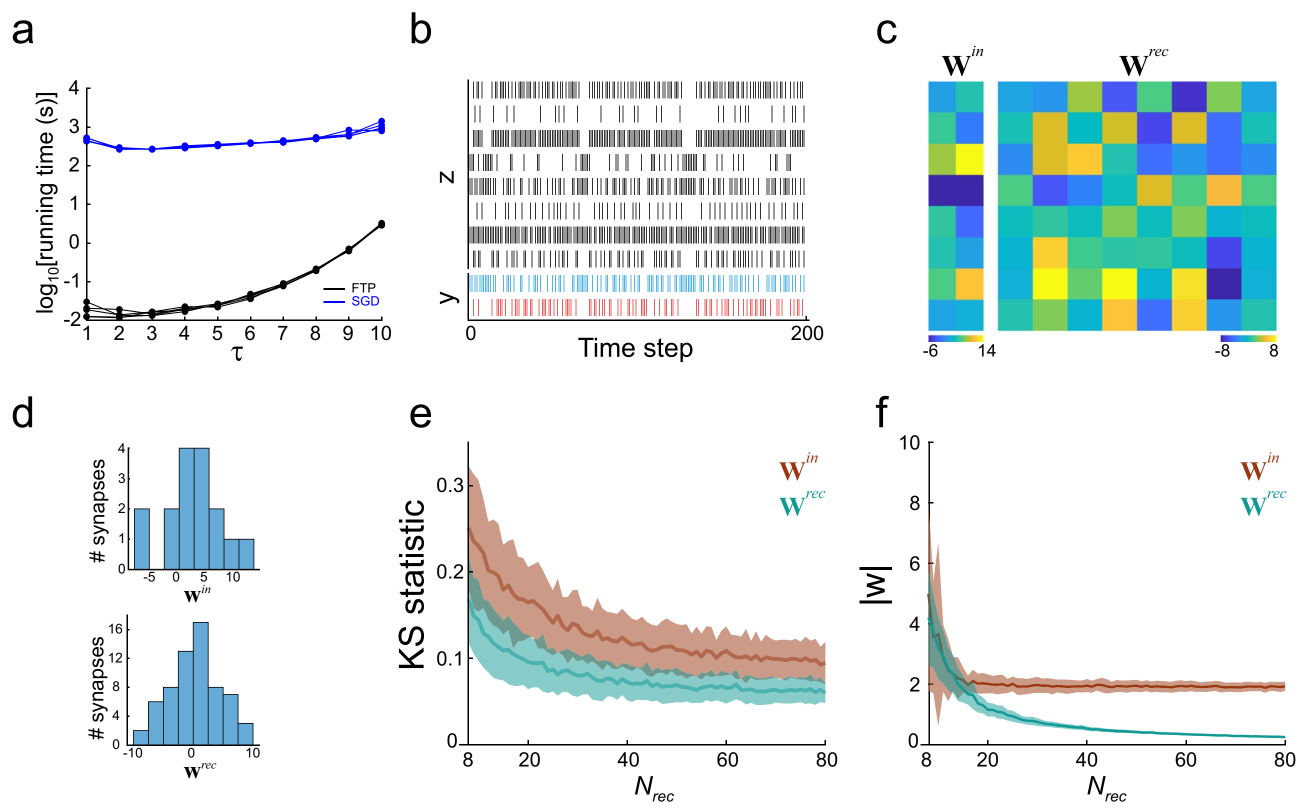


Figure 2. Solving the s-task with the FTP algorithm. (**a**) Efficiency of FTP and SGD, measured as the time expended in finding a solution network for s-tasks of different τ. The time expended by the FTP algorithm is orders of magnitude lower than the time expended by the SGD algorithm (n = 5 networks for each τ value). (**b**) Raster plot showing the neurons firing states in a network constructed to solve the s-task for $\tau = 3$. The network is composed of 8 integration neurons and 2 input neurons. Each possible sequence of 3 stimuli has a unique network firing state that codifies the sequence. Therefore, the network has 8 possible firing states. (**c**) Input and recurrent synaptic weights of the network. (**d**) Distribution of synaptic weights for input (upper panel) and recurrent (lower panel) synaptic weights, for the same network as in (**b,c**). (**e**) Kolmogorov–Smirnov statistic between the distribution of synaptic weights and a normal distribution of the same mean and variance. As the network size increases, the distribution of synaptic weights gets closer to a normal. Integration neurons weights are closer to a normal distribution than sensory neurons weights. Mean ± SD are shown for n = 100 networks that solve the s-task, with $\tau = 3$. (**f**) Absolute synaptic weight values as a function of network size. Absolute values are higher and of larger variability when the neuron count is close to the number of coded stimuli sequences. As the network size increases the absolute mean value and dispersion decrease. Input neurons weights quickly reach a minimum, while integration neurons weights decrease in the entire range of network sizes. Mean ± SD are shown for n = 100 networks that solve the s-task, with $\tau = 3$.

we decided to exploit the FTP capabilities to study the neural dynamics and connectivity in networks of binary neurons that are capable of solving the s-task. Figure 1b shows the network states and state transitions gated by the stimuli when solving the s-task with $\tau = 3$. Despite its complexity, the state transitions required to solve the task have a stereotyped structure, which is evident when network states are numbered properly (Fig. 1b–d). The transition graph in Fig. 1b shows the state transitions any network that solves the s-task should follow. With the transition matrix structure at hand, we can construct matrices **C** and **U** (see details in the "Methods" section).

We assessed the performance of the algorithm by measuring the time it takes to find networks that solve the s-task ($N_{rec} = 1024$, $\tau = 1$ to $\tau = 10$) and comparing those times with the time taken by a stochastic gradient descent (SGD) algorithm (*Adam* optimizer[14], adapted to fit threshold units[15]). The FTP algorithm outperformed SGD by at least two orders of magnitude (Fig. 2a). This is not surprising when considering that a linear system of equations can be solved in polynomial time, while SGD (and any other fitting algorithm) requires huge numbers of network evaluations to obtain a single set of parameters updates. An example network constructed with FTP is shown in Fig. 2b,c, of $N_{rec} = 8$ neurons and capable of solving an s-task with $\tau = 3$. The resulting synaptic weight distribution had zero mean and resembled a normal distribution, at least for the $\mathbf{W}^{rec}$ values (Fig. 2d). In fact, the synaptic weight distributions became progressively closer to a normal distribution as more neurons were employed in network construction (Fig. 2e). We also noted that the absolute weight value decreased, especially for $\mathbf{W}^{rec}$ values (Fig. 2f), which can be explained by thinking that more neurons imply more parameters and hence more degrees of freedom to reach a lower Frobenius norm. This observation will become important later when imposing structural constraints to the network.

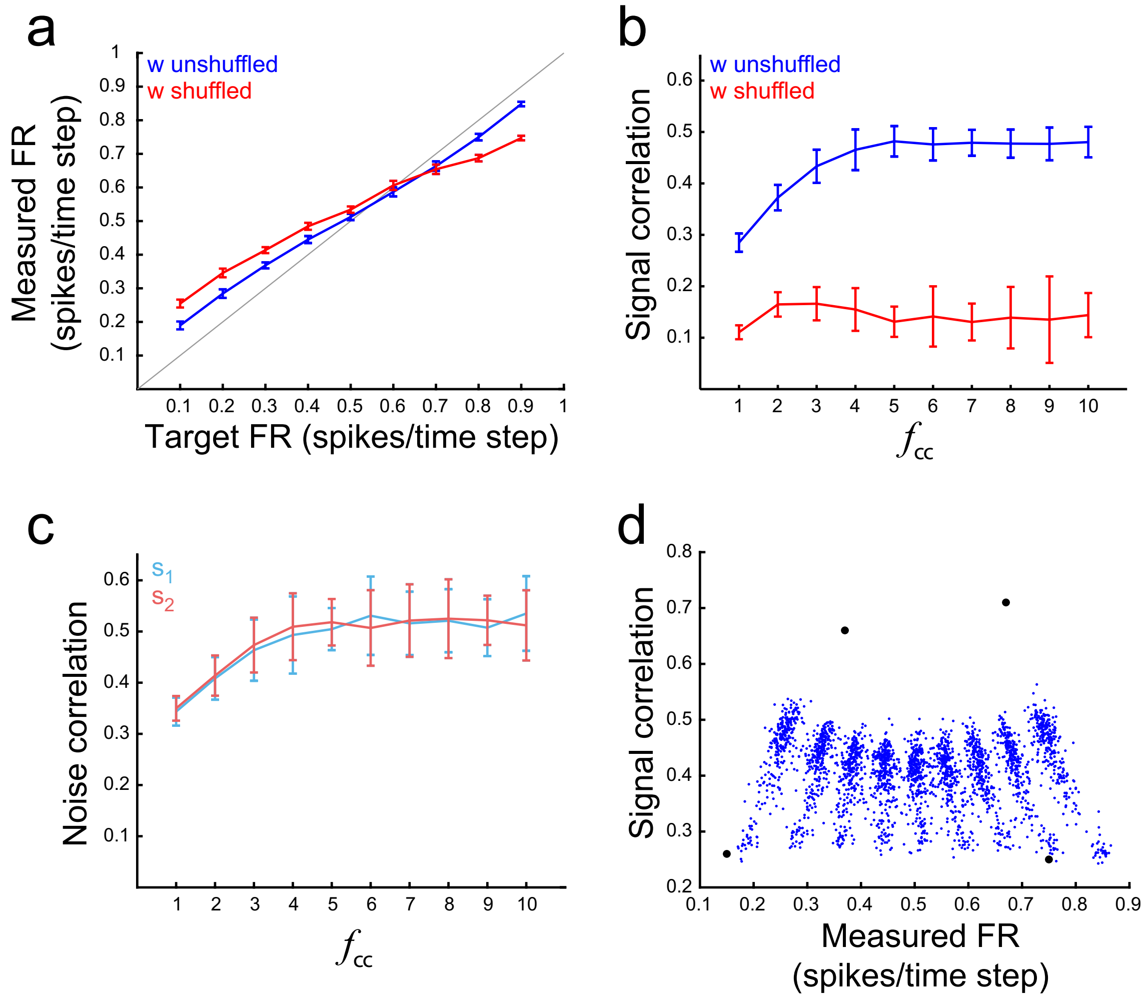
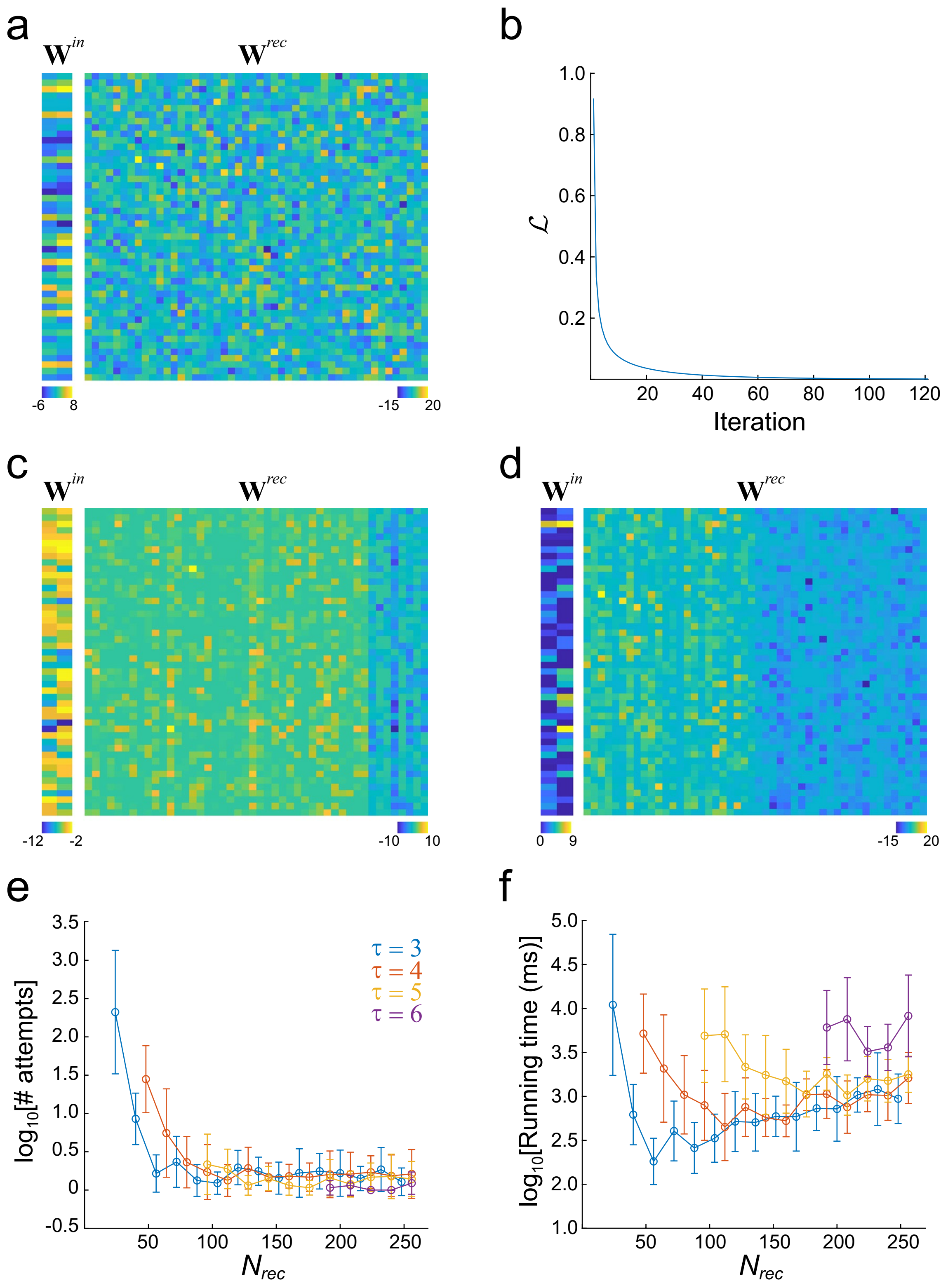


Figure 3. Imposing activity constraints with FTP. (**a**) FR measured in networks constructed to solve the s-task, as a function of the target FR. The FR of networks constructed with the FTP algorithm is close to the target FR (blue line). There is a tendency to obtain lower firing rates for target FR values above 0.5 spikes/time step, and higher firing rates for target FR values below 0.5 spikes/time step (the gray line is the identity function). The same networks with their synaptic weights shuffled (red line) show a similar relationship between target FR and measured FR, albeit with a lower slope. A total of 30 networks were generated for each target FR. Weights in $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$ were shuffled separately. Mean ± SD are shown. (**b**) Correlation between pairs of integration neurons as a function of the scaling factor f_{cc}. Pairwise correlation, computed over all time steps, increases with f_{cc} until it saturates at CC = 0.48 for $f_{cc} \geq 5$ (blue line). Networks with their afferent synaptic weights shuffled (red line) show low correlation, invariant to f_{cc}. A total of 30 networks were constructed for each f_{cc} value, with target FR set to 0.1 spikes/time step. Mean ± SD are shown. (**c**) Pairwise correlation computed separately for s_1 and s_2 (noise correlation). The correlation coefficient increases with f_{cc}, similarly for both stimuli and closely following correlation values in (b). Mean ± SD are shown, n = 30. (**d**) Measured FR as a function of pairwise correlation. Each blue dot shows the FR and CC of one network constructed to solve the s-task with desired FR and CC imposed through $\mathbf{U}_{base}$ initialization. Values for 2700 networks are shown. Points form stripes pointing towards FR = 0.5 spikes/time step, each stripe corresponding to networks with the same target FR. As correlation increases, the measured FR tends to 0.5 spikes/time step. Black dots show FR and CC of 4 networks for which desired FR and CC were imposed by evolution of a population of $\mathbf{U}_{base}$ matrices.

Specifying the firing rate and pairwise correlation of the network. We want to construct neural network models that not only solve relevant tasks but do so under desired firing constraints, as measured in real brains. Some of these constraints are low firing rates (FR)[16,17], or low correlation coefficients (CC)[18]. In regular optimization algorithms, these constraints can be imposed by introducing regularization terms in the loss function[5]. On the other hand, in the FTP algorithm the activation states of the network are the result of linearly combining the rows of an initial matrix $\mathbf{U}_{base}$ (see "Methods"). Hence, we can apply firing constraints by appropriately choosing this initial matrix. For example, to attain networks that solve the s-task with low/high firing, it suffices to choose an initial matrix $\mathbf{U}_{base}$ such that, after thresholding, the resulting matrix $\mathbf{C}$ has few/many ones. Following this procedure, we constructed networks with average FR within a wide range of target FR (Fig. 3a, blue line). Shuffling values within $\mathbf{W}^{in}$ matrices, or within $\mathbf{W}^{rec}$ matrices, produced only small changes to


a
$\mathbf{W}^{in}$
$\mathbf{W}^{rec}$
-6 8
-15 20
b
$\mathcal{L}$
1.0
0.8
0.6
0.4
0.2
20 40 60 80 100 120
Iteration
c
$\mathbf{W}^{in}$
$\mathbf{W}^{rec}$
-12 -2
-10 10
d
$\mathbf{W}^{in}$
$\mathbf{W}^{rec}$
0 9
-15 20
e
$\tau = 3$
$\tau = 4$
$\tau = 5$
$\tau = 6$
$\log_{10}$[# attempts]
3.5 3.0 2.5 2.0 1.5 1.0 0.5 0 -0.5
50 100 150 200 250
N_{rec}
f
$\log_{10}$[Running time (ms)]
5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.0
50 100 150 200 250
N_{rec}

◀**Figure 4.** Imposing structural constraints with FTP. (**a**) Synaptic weight matrices $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$ of the network obtained through FTP before structural constraints were imposed. The network was constructed to solve the s-task for $\tau = 4$ and $f_r = 3$, with a target FR of 0.1 spikes/time step. (**b**) Loss function $\mathcal{L}$ as a function of the number of iterations of the PGD algorithm. The loss function falls below the criterium $e_1 = 10^{-3}$ at iteration 121. (**c,d**) Synaptic weight matrices $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$ for a network with the same stimulus–response mapping but after applying structural constraints: (**c**) no self-connections, Dale's principle, with 40 excitatory and 10 inhibitory neurons, and sparsity $sp = 40\%$; (**d**) no self-connections, Dale's principle, with 26 excitatory and 24 inhibitory neurons, and sparsity $sp = 23\%$. (**e**) Average number of attempts to obtain one network with successful structural fitting, as a function of the number of integration neurons, and for different τ. The number of attempts is high when the neuron number is low, but it decreases fast as the neuron number increases. From 60 neurons onwards, less than five attempts are needed, on average, to obtain one network with the desired structural constraints. Mean ± SD are shown. (**f**), total running time to obtain one network with successful structural fitting, as a function of the number of integration neurons, and for different τ (color code as in (**e**)). Running time decreases and then increases for $\tau = 3$ and $\tau = 4$. The case of $\tau = 6$ is the one with more neurons and equations to solve and present some of the highest running times, even when the number of neurons is high. Nevertheless, all average running times are in the order of tens of seconds.

the average FRs (Fig. 3a, red line). This suggests that it is the distribution of synaptic weights the critical statistic that defines the networks average FR, and not the detailed arrangement of these weights in the synaptic weight matrix.

On the other hand, we can construct networks with desired signal correlation, by multiplying $\mathbf{\Delta}$ (the difference in effects produced by s_1 and s_2) by a factor f_{cc} (Fig. 3b). For networks shown in Fig. 3 ($\tau = 4, f_r = 3$), correlations could be modulated in a range between 0.1 and 0.5. Scaling $\mathbf{\Delta}$ was expected to induce signal correlation. However, it also induced noise correlation as a by-product (Fig. 3c). Correlations in networks solving the s-task were significantly higher than correlations in their synaptic weight-shuffled counterparts (Fig. 3b, blue vs. red), which suggests that signal correlations depend on the precise arrangement of values in the weight matrix, and not only on the weight distribution, as was the case with FR. It also suggests that the set of networks that solve the s-task necessarily exhibit correlations above a minimum. On the other hand, correlations did not exceed a certain value: higher correlations may imply a reduced number of network states, incompatible with the number of sequences required to be codified.

Hand-based manipulation of $\mathbf{U}_{base}$ allows us to generate solution networks in a wide range of FR and CC (Fig. 3d, blue dots). An even better control of firing and correlation can be achieved by fitting $\mathbf{U}_{base}$ thought an optimization algorithm. We employed a genetic algorithm (GA) in which the fitness of $\mathbf{U}_{base}$ is a function of the FR and CC computed from the network firing states generated from that $\mathbf{U}_{base}$. Fitting $\mathbf{U}_{base}$ gave access to more extreme values of FR and CC (Fig. 3d, black dots), and kept computational cost low by computing FR and CC from the set of network firing vectors $\mathbf{c}$ instead of computing the firings by simulating the network. Altogether, both methods ($\mathbf{U}_{base}$ manipulation, or its evolution with a GA) generated networks with desired activity constraints that perfectly solved the task.

Applying structural constraints with projected gradient descend in isofunction weight space.

Networks generated so far share one structural constraint: their synaptic weight matrix is the one that minimizes the Frobenius norm. Other relevant structural constraints, such as the low probability of self-connections[19], Dale's principle[4], or sparse connectivity are not satisfied. Since these structural constraints are key experimentally observed features, we were interested in imposing such constraints onto the $\mathbf{W}$ obtained by the algorithm. To do this we followed a projected gradient descent (PGD) approach[20], taking advantage of the fact that the loss function $\mathcal{L}$, which encloses the structural constraints, is a linear function with respect to the synaptic weights, and that the matrix $\mathbf{W}$ can be changed without changing the stimulus–response mapping (see "Methods"). To exemplify the procedure, we constructed a network that solves the s-task for $\tau = 4$, with $f_r = 3$ (Fig. 4a), and then we employed PGD to remove self-connections, enforce Dale's principle with a 4:1 Ex:In ratio, and set a sparsity $sp = 40\%$ (defined as the percentage of weights equal to zero). The PGD steadily reduced the loss $\mathcal{L}$, reaching a negligible value, provided that the network had enough neurons (Fig. 4b). It is remarkable that so dissimilar synaptic weight matrices, like the ones depicted in Fig. 4a,c,d, gave rise to the same stimulus–response mapping.

We noted that structural constraints could not be imposed on networks with low number of neurons, for example, networks with $N = N_{in} + N_{rec} < 2M$. This is not surprising, since it is expected that imposing more constraints require more parameters. To evaluate the efficiency of the PGD in relation with the number of neurons, we imposed the above structural constraints for networks that solved the s-task with $\tau = 3$ to $\tau = 6$, and N_{rec} between 32 and 256 neurons. Since matrix $\mathbf{U}$ and vector $\boldsymbol{\theta}$ were randomly chosen, it was expected that some of them resulted in matrices $\mathbf{W}$ for which the structural constraints were impossible to apply. Consequently, we measured the efficiency of PGD by computing *# attempts*, the number of networks that were required until obtaining the first successfully constrained network. It can be seen that *# attempts* decreased as the number of neurons increased (Fig. 4e). Concordantly, the computing time required to obtain a successfully constrained network decreased as the number of neurons increased (Fig. 4f), probably due to the decrement in *# attempts* that were required. The fitting time was higher for networks with the highest neuron count, but always within the order of tens of seconds, even for $N_{rec} = 256$.

Exploiting the FTP algorithm for hypothesis testing.

A distinctive aspect of the FTP algorithm is that the desired dynamic of the network is defined first and in great detail. In fact, we can specify the network

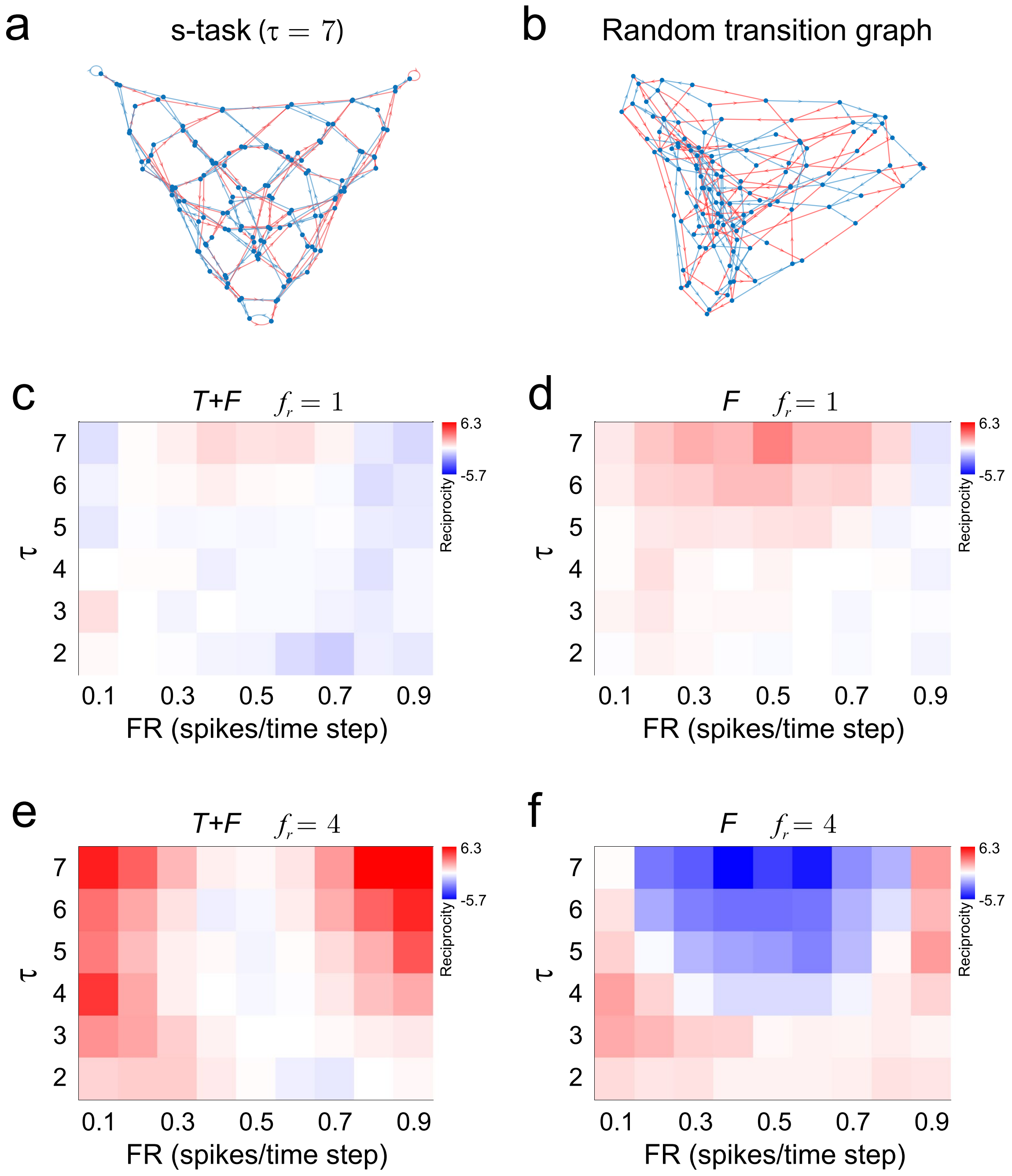


Figure 5. Reciprocity as a function of τ, FR and the type of transition graph. (**a**) Transition graph for solving the s-task with $\tau = 7$. Blue and red lines represent transitions gated by s_1 and s_2, respectively. (**b**) Random transition graph. Nodes (network states) may receive different number of incoming connections. There are 24 nodes that are gated by both stimuli. (**c**) Reciprocity for $T+F$ networks, with $f_r = 1$, as a function of τ and target FR. Reciprocity changes from slightly negative to slightly positive as τ increases. For $\tau = 7$, reciprocity is maximized around target FR = 0.5 spikes/time step and decreases for lower and higher values of target FR. (**d**) F networks with $f_r = 1$ shows increasing positive reciprocity as τ increases, maximized at target FR = 0.5 spikes/time step. (**e**) When the number of neurons is higher ($f_r = 4$), $T+F$ networks show positive reciprocity that is minimal around target FR = 0.5, and increases towards higher and lower target FR, reaching the highest reciprocity values among all networks screened. (**f**) Reciprocity of F networks turns increasingly negative as τ increases, reaching the lowest reciprocity among all networks screened, around target FR = 0.5 spikes/time step. For all panels, 30 networks were constructed for each τ and target FR combination. Normalized means (mean/SD) are shown. Positive and negative reciprocity values were mapped separately to colours red and blue, respectively. Red tones go from 0 reciprocity (white) to maximal positive reciprocity (pure red). Blue tones go from 0 reciprocity (white) to maximal (in absolute value) negative reciprocity (pure blue). All random graphs were constructed with $f_{bc} = 0.5$. Graphs were plotted with the Force-directed layout.

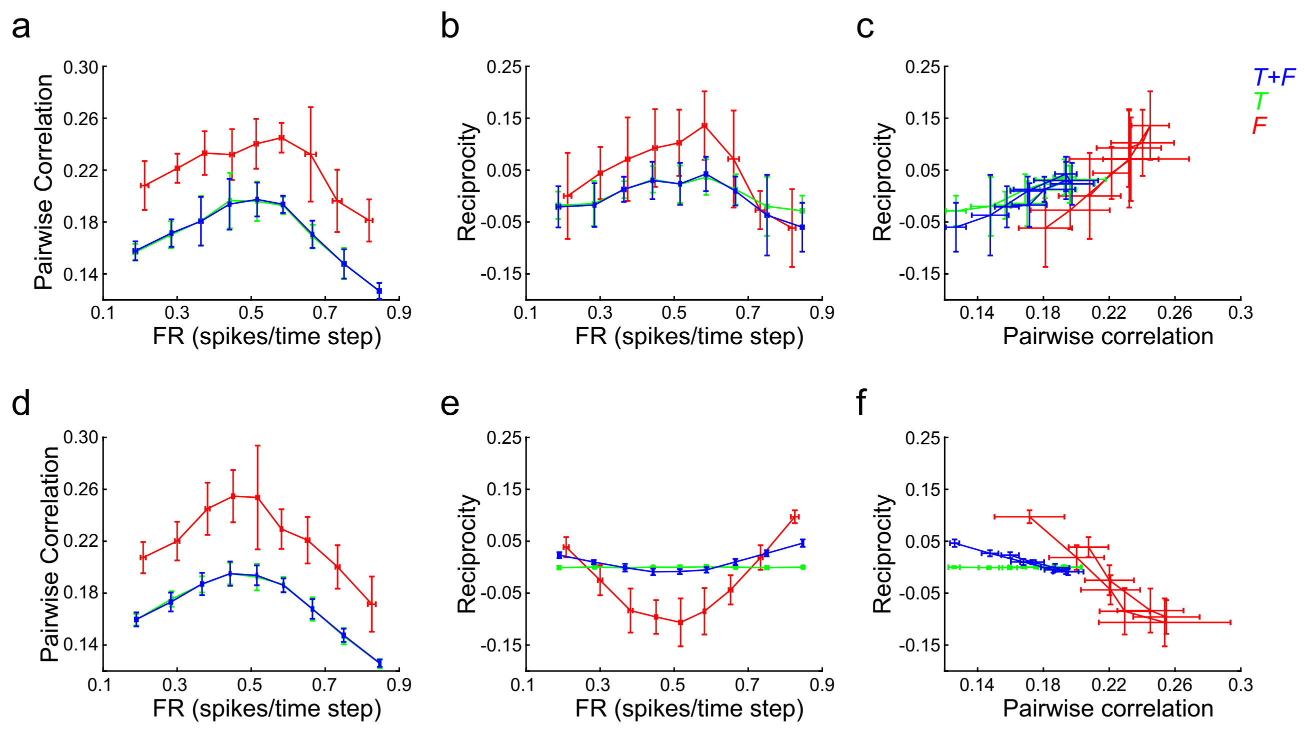


Figure 6. Correlation and reciprocity differentiate networks with sequence memory from random transition networks. (**a–c**) Networks constructed to solve the s-task with $\tau = 7$ and $f_r = 1$ ($T+F$ networks), their isofunction network (T network), and networks with the same number of neurons and network states that follow a random transition graph (F networks). (**a**) Correlation increases as FR approaches 0.5 spikes/time step. F networks show a positive offset in comparison with $T+F$ and T networks. (**b**) The dependency between reciprocity and FR is similar to the dependency between CC and FR. Higher reciprocity values are found in F networks. (**c**) Reciprocity grows linearly with correlation, as expected from panels (**a**) and (**b**). (**d–f**) idem **a–c**, but with $f_r = 4$. (**d**) The CC/FR relationship is similar to the one observed with a lower neuron number (panel (a)). (**e**) The reciprocity/FR relationship inverted as the neuron number was increased. Reciprocity is minimized as FR approaches 0.5 spikes/time steps. F networks show pronounced negative reciprocity. (**f**) Reciprocity decreases linearly with correlation, as expected from panels (**d**) and (**e**). Mean ± SD are shown; n = 10 networks were constructed for each target FR and network type. All random graphs were constructed with $f_{bc} = 0.25$.

dynamic up to the level of single state transitions, and the algorithm finds the set of synaptic weights that makes the desired dynamic possible. This approach is ideal for testing very specific hypotheses about the relationship between network function (neural dynamics and task performance) and the underlying structure (synaptic connectivity).

We wanted to understand how the firing rate and firing dynamic constrained connectivity. To that end, we constructed networks that solved the s-task, over a range of network sizes and mean firing rate, and compared them with networks that had the same number of neurons and network states but for which the graphs of transitions between network states were generated at random (Fig. 5a,b). These "random transition" networks did not have sequence memory, but their dynamic was similar in complexity to that of networks that solved the s-task, turning them in perfect examples to study the connectivity features behind the capacity to codify sequences of stimuli. We focused on measuring the level of reciprocity in the network, which has been observed experimentally[21] and its implications studied theoretically[22]. By screening networks with memory ranging from $\tau = 2$ to $\tau = 7$, and FR from 0.1 spikes/time step to 0.9 spikes/time step, we found that reciprocity varied with τ, FR and neuron number. In particular, we observed that, when $f_r = 1$, reciprocity was positive and of lower mean for networks that were the minimum Frobenius norm solution to the s-task ($T+F$ networks, Fig. 5c), in comparison with networks that were the minimum Frobenius norm solution to a random transition graph (F networks, Fig. 5d). However, for bigger f_r the relationship was inverted, and $T+F$ networks showed positive reciprocity (Fig. 5e) while F networks showed negative reciprocity (Fig. 5f).

To further describe these relationships, we selected networks constructed for $\tau = 7$ and $f_r = 1$ (Fig. 6a–c) and $f_r = 4$ (Fig. 6d–f). Signal correlation varied with FR following an inverted U-shape relationship, with a maximum next to 0.5 spikes/time step (Fig. 6a,d). Note that, up to FR = 0.5 spikes/time step, CC increased with FR, as has been observed experimentally[23,24]. Interestingly, the way CC changed with FR was similar for both $T+F$ and F networks, with the distinction that F networks had overall higher correlations than $T+F$ networks. Shuffling the inter-spike interval of each neuron destroyed the dependence between FR and CC. The CCs after shuffling were also much smaller than the CC values of the non-shuffled firings ($CC_{shuffled} = 0.0224 \pm 1.10^{-4}$, mean ± SD, for n = 90 T + S networks pooled over all FR values). These results ruled out the possibility that correlations were

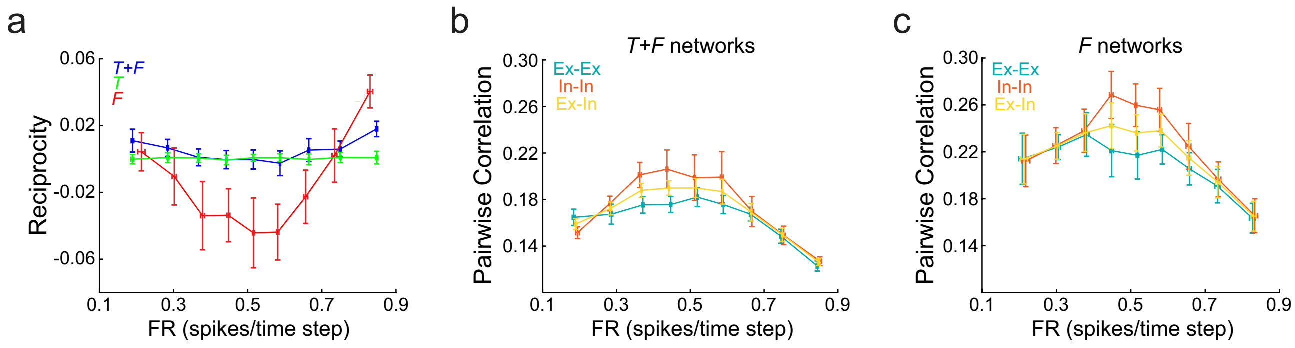


Figure 7. Reciprocity and correlation of structurally constrained networks. (**a**) Reciprocity as a function of FR for networks without self-connections and Dale's principle with 1:1 Ex:In ratio. Reciprocity shows a parabolic relationship with FR, decreasing as FR approaches 0.5 spikes/time step. *F* networks show strong negative reciprocity, while *T* networks reciprocity is close to zero. (**b**) Pairwise correlation for *T* + *F* networks as a function of firing rate. The correlation was computed over pairs of excitatory neurons (Ex-Ex), pairs of inhibitory neurons (In-In), and pairs of one excitatory and one inhibitory neuron (Ex-In). Correlation has a maximum close to 0.5 spikes/time step. The In-In pairs show the highest correlations, followed by the Ex-In pairs. The Ex-Ex pairs show the lowest correlation. (**c**) pairwise correlation for *F* networks as a function of firing rate. The CC/FR relationship is similar to the one observed for *T* + *F* networks, although *F* networks correlation is displaced towards higher values. Mean ± SD are shown; n = 20 networks were constructed for each target FR and network type. Firing rates of excitatory or inhibitory neurons are displayed for Ex-Ex and In-In curves, respectively. For Ex-In curves, the average FR over all neurons is shown.

trivially increasing with FR because of the higher number of spikes. We also found an inverted U-shape relationship between reciprocity and FR, and a linear relationship between reciprocity and CC. With $f_r = 1$ reciprocity tended to be maximized as FR approached 0.5 spikes/time step, with *F* networks showing higher (and positive) reciprocity (Fig. 6b). With $f_r = 4$, reciprocity tended to increase as FR departed from 0.5 spikes/time step towards lower and higher values, i.e., networks with lower correlation (Fig. 6e). Specially, *F* networks showed negative reciprocity for all firing rates, except for the more extreme cases (0.1 and 0.9 spikes/time step). Just like the reciprocity/FR relationship inverted with the number of neurons, so did the reciprocity/CC relationship. Networks with higher reciprocity had higher correlation when the number of neurons was low (Fig. 6c). However, and somewhat counterintuitive, when the number of neurons was higher, more reciprocity implied lower correlation (Fig. 6f). Networks that solved the s-task but did not minimize the Frobenius norm (*T* networks) showed almost zero reciprocity, implying that reciprocity was not a property of all networks that solve the s-task. On the contrary, most networks that solved the s-task did not show significant reciprocity, unless other structural constraints, such as Frobenius norm minimization, was imposed. However, Frobenius norm minimization alone only produced negative reciprocity (in random graphs). For positive reciprocity to occur in networks with high number of neurons, both high sequence memory and Frobenius norm minimization were required.

We asked whether the results depicted in Fig. 6 could be replicated in networks that lacked self-connections and complied with Dale's principle. To that end we imposed these structural constraints to networks constructed with $\tau = 7$ and $f_r = 4$, and found a reciprocity/FR relationship that resembled the one observed in unconstrained networks, with *F* networks showing prominent negative reciprocity and *T* + *F* networks showing increasing reciprocity as FR departed from 0.5 spikes/time step (Fig. 7a). Correlations increased as FR approached 0.5 spikes/time step, with *F* networks showing more correlation than *T* + *F* networks (Fig. 7b,c). Correlation in *T* + *F* and *F* networks were higher for pairs of inhibitory neurons than for pairs of excitatory neurons, as has been observed experimentally[25], while correlations between excitatory and inhibitory neurons lied in the middle.

We went further on by asking which features of network dynamics were responsible for the differences in reciprocity that we observed between *T* + *F* and *F* networks. The FTP algorithm finds the network connectivity from a detailed description of the network dynamics, specified in its transition graph. Therefore, we hypothesized that specific changes in the transition graph could have a precise impact in network reciprocity. In *T* + *F* networks, any given network state could only be reached after the presentation of either s_1 or s_2, but not both. This is clearly visualized in the transition graph of Fig. 1b, in which each node has all blue or all red incoming edges, but never incoming edges of both colours. These network states codify stimuli in an absolute manner since it only suffices to know the network state at time step *t* to know the identity of the presented stimuli at that time step. We termed these nodes *monocoloured* nodes. On the other hand, in an *F* network each network state can be reached from either s_1 or s_2, or from both. Network states which can be reached from both stimuli (termed *bicoloured* nodes) codify stimuli in a relative manner, meaning that the identity of the stimulus presented at time step *t* can be decoded only if *both* the network states at *t* and $t - 1$ are known. Then, we specifically asked whether the proportion of bicolored and monocoloured nodes could explain the strong differences in correlation and reciprocity found between *T* + *F* and *F* networks. To that end, we constructed *F* networks with increasing fraction of bicolored nodes (f_{bc}), and computed reciprocity for fixed τ, FR and f_r (Fig. 8). We found that reciprocity became increasingly more negative as we increased the fraction of bicolored nodes. When all network states were monocoloured ($f_{bc} = 0$), networks showed zero reciprocity, as observed in *T* + *F* networks with the same FR and f_r. This result highlights the power of the FTP algorithm to relate neural dynamics with network connectivity.

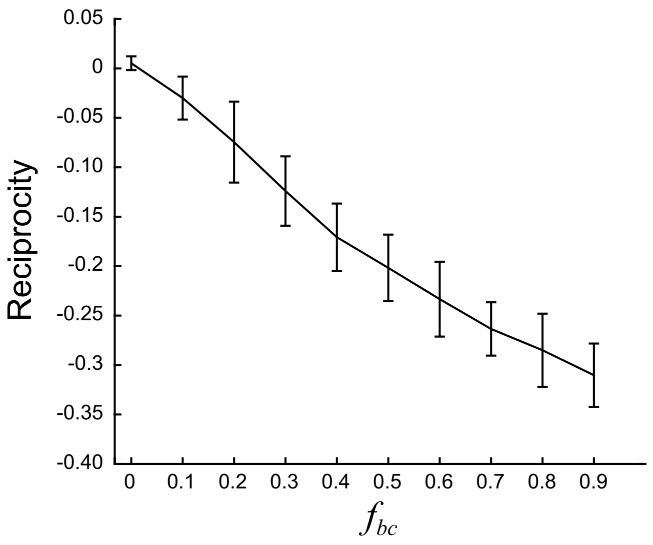


Figure 8. Relative-coding network states cause negative reciprocity. Reciprocity as a function of f_{bc}, the fraction of bicolored nodes, in networks that follow random transition graphs. Reciprocity decreases linearly with f_{bc}, approaching zero as f_{bc} approaches zero. Mean ± SD are shown, n = 30 networks for each f_{bc}. Networks were constructed with target FR = 0.5 spikes/ time step, and with the same number of neurons and network firing states as $T+F$ networks constructed with $\tau = 7$ and $f_r = 4$.

Discussion

We have presented a simple method to generate binary neural network models that accomplish the desired task. Networks composed of binary neurons are computationally inexpensive, and despite their simplicity many neurophysiological and neuroanatomical observations have been recapitulated employing these networks[22,26]. Our key contribution is to note that, for networks in which neuron inputs are linearly added, their synaptic weights can be found by solving a system of linear equations. In turn, this system can be constructed from the transition graph associated with the solution of the target task. System consistency is guaranteed if the dependent variables of the system (the neuron activations) are linearly combined following the linear dependences among the independent variables (the firing states). We have shown how the FTP algorithm works with the simplest of networks. Yet, we think that the same procedure could be implemented in networks composed of more complex neuron models, like the firing rate model or the leaky integrate-and-fire model, provided that a system of linear equations can be constructed.

Current automated methods of network model construction rely on off-the-shelf optimization algorithms typically employed in the artificial intelligence field, like stochastic gradient descent[5], genetic algorithms[27], or evolutionary strategies[28]. These optimization algorithms iteratively change network parameters in a direction that minimizes a loss function and have proved to be very effective in finding networks that solve very complex tasks[29,30]. However, they require a considerable amount of human intervention, and there is no certainty that they will arrive at a solution. Moreover, each optimization iteration requires the evaluation of the network, which is time consuming, especially for a recurrent network performing in a multi-trial task. In contrast, the FTP algorithm reduces the problem of finding a suitable network to a series of linear combinations and the solution of a linear system, with both operations performed in polynomial time. Most importantly, it is guaranteed that the resulting network will solve the task perfectly.

Traditional optimization algorithms require the definition of a loss function that encompasses all the constraints the network should satisfy, whether these are task related, activity related, or structural. Then, the loss function is minimized and hence all constraints are enforced at once. In this scenario, the relationship between parameters and the loss function can be quite complex, and conflict between constraints may emerge. Conversely, one key advantage of our method is that it uncouples the dynamic and coding aspects of the network from the structural aspects, giving us the opportunity to adjust them independently. Moreover, since the method proceeds from the network firing states to its parameters, it allows us to find networks with desired activity profiles, and to study the resulting connectivity. Further structural constraints can be enforced in a second stage, by projected gradient descent, or any other optimization algorithm. The fact that projected gradient descent worked so well suggests that structural constraints are easy to implement once the connectivity required to solve the task is in place. This is probably because many connectivity features of a network, such as its sparseness or its clustering coefficient, are a simple function of its synaptic weights. On the contrary, the relationship between the synaptic weights and the network function, or coding capabilities, is far more complex. In this way, our method gives more control over each class of constraint, making the whole process simpler at the same time.

To show its applicability, we employed the FTP algorithm to construct networks that solve a stimuli sequence memory task (s-task) in which networks have to codify in their network firing states the sequence of the last τ stimuli that were presented. The s-task, akin to the n-back task commonly employed in cognitive neuroscience, is relevant within the scope of working memory function. Working memory is traditionally associated with maintaining information about a single stimulus in the persistent activity of recurrently connected neurons[31,32], although mounting evidence suggests that neuron populations code information in the form of highly heterogenous firing sequences[33,34]. Sustained activity can be a suitable strategy when there is one specific relevant stimulus to attend, whose identity has been already elucidated. However, more complex scenarios require keeping track of sequences of stimuli. An example of this case is the processing of language, in which the succession of utterances must be integrated over time, from phonemes to words, to phrases, so that the meaning of speech depends on the whole sequence[35]. We explored the case of two stimuli presented with equal probability, but the analysis could be extended to more realistic cases in which the stimulus distribution is not uniform. It is expected that statistical regularities in the sequences of stimuli are going to be exploited by the network, resulting in more specialized connectivity. The relationship between sequence statistics and network structure should be further studied. For example, it would be of interest the case in which stimuli last more than one time-step, and they are interleaved by another set of stimuli that act as distractors. Then, the relationship between feedback and feedforward connections could be studied, in relation with the duration of each stimulus presentation, and with that of the distractors.

The structure–function relationship is central to neuroscience[36–38]. Connectivity at the macro, meso, and micro scale, neuron biophysics, plasticity mechanisms, among other structural traits, all act co-ordinately to give sophisticated adaptive behaviour. It is widely believed that the structural properties of networks have evolved to proficiently perform a function, many times in an optimal way[39,40]. However, brain structure could also be the result of other constraints, different from those imposed by adaptive behaviour. For example, neural network modularity might have emerged as an adaptative structural trait for solving tasks that have a modular or hierarchical component[41], but it could also have emerged as the result of previously acquired structural traits, such as constraints in the length of dendrites and axons, which preclude the possibility of a much wider connectivity. Thus, determining how much of the structure observed in the brain comes from task-related constraints and how much comes from other structural traits is central to understanding the structure/function relationship. A theoretical approximation to this issue consists on constructing neural network models that solve different kinds of tasks under a variety of structural constraints, and then study the patterns of connectivity that emerge and relate them to the observed connectivity in the brain. This approximation requires to sample as uniformly as possible from the set of networks that fit both the task and the structural constraints. However, optimization methods commonly employed in network parameter fitting may give a restricted set of solutions, thus biasing any conclusion about the structure/function relationship. Another issue is that some connectivity traits could emerge only in networks of certain size, and fitted to several tasks. In this case, fitting large networks with complex cost functions could have a high computational cost. Consequently, generating a relatively large sample of networks suitable for statistical treatment of their connectivity could result unfeasible. On the other hand, the FTP algorithm is very well suited for answering structure/function questions, since exact solutions can be computed starting from an arbitrary set of population firing states, as long as they define a system of equations that has a solution.

Hypotheses that link structure and function can be tested with the help of the FTP algorithm, by constructing networks that follow transition graphs that instantiate some null hypothesis. Following this approach, we constructed networks that had the same number of network states and neurons required to solve the s-task, but whose state transitions were chosen at random. With this tool at hand, we were able to show how network reciprocity depended on the memory demand and the size of the network. The same procedure can be followed to build any other set of networks under some relevant null hypothesis. Such networks can be easily constructed with the FTP algorithm, while they would be hard to construct with regular optimization algorithms.

Evidence for high reciprocity has been found experimentally, by measuring excitatory postsynaptic potentials of reciprocally connected neurons *in vitro*[21]. It has also been the centre of theoretical analysis. For example, it has been shown that high reciprocity is recapitulated in networks of binary neurons that have a maximum number of attractors[22]. Interestingly, the same work showed that reciprocity is lost when networks are optimized to store sequences of uncorrelated network states. However, we found that networks of high reciprocity are capable of displaying long firing state sequences when their dynamics codify the sequences of previously presented stimuli. Therefore, our work complements previous studies which have shown that reciprocity is one of the key connectivity features that support the computation performed by the neocortex.

Reciprocity was absent in networks taken at random from the set of all connectivities that solve the s-task (the T networks in Fig. 6). This implies that the observed reciprocity is the result of solving the s-task with the additional constraint of weights minimizing the Frobenius norm, the latter being understood as the consequence of an upper bound on the number of receptors and vesicles in a synapse. Thus, to explain one structural feature (reciprocity), a functional feature (solving the s-task), and another structural feature (Frobenius norm minimization) were required. It would be interesting to study to what extent other structural features encountered in biological neural networks, like modularity or sparsity of connections, can be explained as the answer to some computational demand of adaptive behaviour, or as the byproduct of another structural feature, or as the interaction of both factors, as is the case with the s-task.

In conclusion, we have provided an algorithm that inverts the usual process by which neural networks are constructed. It can be employed to probe the dependency between the firing statistics, connectivity, and function of a network in a way that is not matched by current optimization algorithms. Moreover, it is computationally inexpensive. For these reasons, we consider the FTP algorithm to be a powerful alternative method for constructing neural network models of brain function.

Symbol	Description
$\mathbf{y}$	Binary row vector of length N_{in}. Represents the firing state of input neurons. Codifies a stimulus
$\mathbf{z}$	Binary row vector of length N_{rec}. Represents the firing of the recurrent network
$\mathbf{u}$	Real-valued row vector of length N_{rec}. Represents the activation states of neurons in the recurrent network
$\mathbf{W}^{in}$	Matrix of synaptic weights from sensory neurons to the recurrent network. Columns are incoming connections
$\mathbf{W}^{rec}$	Matrix of synaptic weights among neurons of the recurrent network. Columns are incoming connections
N_{in}	Number of input neurons ($N_{in} = 2$ throughout this work)
N_{rec}	Number of neurons in the recurrent network
$\boldsymbol{N}$	Total number of neurons (input and integration)
$\boldsymbol{M}$	Total number of network firing states
$\boldsymbol{\theta}$	Row vector of neuron's thresholds
$\mathbf{c}$	Row vector, obtained after concatenating one $\mathbf{y}$ vector with one $\mathbf{z}$ vector
$\mathbf{C}$	Matrix whose rows are $\mathbf{c}$ vectors. Coefficient matrix in a system of linear equations
$\mathbf{U}$	Matrix composed of row vectors $\mathbf{u}$. Contains activation states reached by the network from the firing states in matric $\mathbf{C}$
$\mathbf{W}$	Matrix resulting from concatenating matrices $\mathbf{W}^{in}$ and $\mathbf{W}^{rec}$
$\mathbf{U}_{base}$	Matrix of row vectors $\mathbf{u}$ picked at random
$\mathbf{U}_{lc}$	Matrix composed of the rows in $\mathbf{U}_{base}$ and linear combinations thereof. There is one row for each network state
$\mathbf{Z}$	Binary matrix, obtained by applying threshold $\boldsymbol{\theta}$ to matrix $\mathbf{U}_{lc}$
$\boldsymbol{\Delta}$	Real-valued vector of length N_{rec}. Each component is the difference between activations after s_1 and s_2 presentation, when starting from the same network firing state
$\boldsymbol{f_r}$	Redundancy factor: quotient between the number of neurons and the number of sequences codified in an s-task
$\boldsymbol{f_{cc}}$	Multiplying factor to induce signal correlation
$\boldsymbol{f_{bc}}$	Number of network states reachable after s_1 or s_2 presentation, divided by the total number of network states

Table 1. Symbols with descriptions.

Methods

Nomenclature. Vectors are represented with bold lowercase letters and are considered row vectors. Matrices are represented with bold uppercase letters. In Table 1 we have summarized the principal symbols employed throughout the paper, together with their description.

Attaining system consistency in the s-task. We consider a network that solves an s-task, with 2 stimuli and sequences of length τ. Therefore, the network has to display at least a number of network states $M = 2^\tau$. If we number the network states and sort transitions as in Fig. 1b–d, we have that, if Eq. (3) has a solution, rows in $[\mathbf{C}\ \ \mathbf{U}]$ (the augmented matrix) should satisfy:

$$[\mathbf{y}_2\ \ \mathbf{z}^{T+1}\ \ \mathbf{u}^{T+1}] = [\mathbf{y}_1\ \ \mathbf{z}^{T}\ \ \mathbf{u}^{T}] - [\mathbf{y}_1\ \ \mathbf{z}^{P}\ \ \mathbf{u}^{P}] + [\mathbf{y}_2\ \ \mathbf{z}^{P+1}\ \ \mathbf{u}^{P+1}] \tag{6}$$

where T values are odd numbers between 1 and $2M - 1$ such that $[\mathbf{y}_1, \mathbf{z}^T]$ and $[\mathbf{y}_2, \mathbf{z}^{T+1}]$ are the T and $T+1$ rows of $\mathbf{C}$, and $\mathbf{u}^T$, $\mathbf{u}^{T+1}$ are the T and $T+1$ rows of $\mathbf{U}$. The row index P is and odd number between 1 and $2M - 1$. Note that $\mathbf{z}^T = \mathbf{z}^{T+1}$ and $\mathbf{z}^P = \mathbf{z}^{P+1}$, but $\mathbf{u}^T \neq \mathbf{u}^{T+1}$ and $\mathbf{u}^P \neq \mathbf{u}^{P+1}$. Equation (6) shows us how row vectors in matrix $\mathbf{U}$ should be linearly combined such that Eq. (3) has a solution. We have that:

$$\mathbf{u}^{T+1} = \mathbf{u}^T - \mathbf{u}^P + \mathbf{u}^{P+1} \tag{7}$$

This means that the number of linear combinations in $\mathbf{U}$ is $R = 2^\tau/2 - 1$, and $\text{rank}(\mathbf{U}) = 2^\tau/2 + 1$. Note that rewriting Eq. (7) we have:

$$u_{m,s_2,i} - u_{m,s_1,i} = \Delta_i \tag{8}$$

where $u_{m,s_1,i}$ and $u_{m,s_2,i}$ are the activations that neuron i adopts after the presentation of s_1 and s_2, and coming from a network state m. In other words, Eq. (8) tells that the difference in effects provoked by the stimuli is a constant for each neuron, regardless of which network state or transition we are dealing with. This fact is not surprising, since synaptic weights are held fixed, so each stimulus has the same effect at any time, which is specific for each neuron. Thus, making the system of equations in (3) consistent is equivalent to guarantee that activation values are chosen so that each stimulus has a constant effect on the synaptic activations.

To construct matrix $\mathbf{U}$ we first defined a vector of thresholds $\boldsymbol{\theta}$ with elements $\theta_i \in \{1/2, 3/2, 5/2\}$. Then, we constructed base matrix $\mathbf{U}_{base}$ with $M_{base} = 2^\tau/2 + 1$ row vectors such that:

$$\mathbf{U}_{base}(m, i) = \theta_i + r(m, i) + \frac{1}{2}$$

where $r(m, i)$ is an integer uniformly sampled from the $[-5,5]$ interval. We added the term 1/2 to avoid fitting errors when numerically solving the system, otherwise the activation values could be equal to the threshold values, which would result in erroneous firing states because of numeric precision issues. We chose a uniform distribution over integers for simplicity, although any other distribution could be employed. The same goes for the threshold values.

This initial randomly generated matrix $\mathbf{U}_{base}$ is required to be full rank. We computed the vector $\boldsymbol{\Delta}$ of Δ_i elements as the difference between the first two rows of $\mathbf{U}_{base}$. Next, we applied Eq. (8) to generate the remaining R rows as linear combinations of the third to the last row of $\mathbf{U}_{base}$, obtaining 2^τ row vectors that constitute the matrix $\mathbf{U}_{lc}$. Each row vector $\mathbf{u}$ in matrix $\mathbf{U}_{lc}$ had the neuron activations for one of the 2^τ network states. Applying Eq. (8) creates a dependency between $u_{m,s_1,i}$ and $u_{m,s_2,i}$. Hence, for each linear combination we chose at random which activation value (the one associated with s_1 or s_2) was defined in terms of the other. This was to ensure that u value distributions were equal for both stimuli. We constructed matrix $\mathbf{Z}$ by applying threshold $\boldsymbol{\theta}$ to $\mathbf{U}_{lc}$, and then we followed the ordering depicted in Fig. 1b–d to construct matrix $\mathbf{U}$ from $\mathbf{U}_{lc}$, and matrix $\mathbf{C}$ from $\mathbf{Z}$ and vectors $\mathbf{y}_1$, $\mathbf{y}_2$. Finally, we employed Eq. (4) to obtain the synaptic weight matrix $\mathbf{W}$. Since matrix $\mathbf{W}$ is the minimum Frobenius norm solution to Eq. (4) and defines a network that solves the s-task, we say that $\mathbf{W}$ defines a *Task + Frobenius* (*T + F)* network.

The algorithm described above works under the assumption that two conditions are met after thresholding: 1) the resulting vectors $\mathbf{z}$ are all different, and 2) they are linearly independent. If thresholding $\mathbf{U}_{lc}$ gives vectors $\mathbf{z}$ that appear more than once, this would result in lower performance in the task, since not all sequences of length τ will be encoded. On the other hand, if linear independency fails after thresholding, then matrix $\mathbf{C}$ will have more linear combinations than the contemplated in Eq. (5), meaning that combining the rows of $\mathbf{U}$ following Eq. (8) will not be enough, and some linear dependencies in $\mathbf{C}$ will be lost in the augmented matrix, making the system inconsistent. In our implementation of the algorithm, if any of these two conditions were not verified, then the algorithm was restarted from the beginning. This occurred with low probability, for $\tau < 5$. For higher τ, both conditions were always fulfilled in one attempt.

In the previous explanation we assumed that $N_{rec} = 2^\tau$, such that there is one neuron per sequence of length τ. It was possible to fit networks with lower number of neurons, but undesired linear dependencies in $\mathbf{C}$ after thresholding, or a number of network states bellow 2^τ occurred with higher probability, especially for $\tau > 3$.

Network simulation and synaptic weights statistics. Networks were evaluated in the s-task during at least $N_{iter} = 10.2^\tau$ time steps, to gather enough samples of each network state. To assess the similarity between the synaptic weight distribution and a normal distribution we computed the Kolmogorov–Smirnov two samples statistic, between the set of synaptic weights and a set of normally distributed values of the same mean, variance, and sample size than that of the synaptic weights.

Equation (4) gives the matrix $\mathbf{W}$ with lowest Frobenius norm[12]. Since we are considering networks with $N = 2^\tau + 2$ total neurons (including input and integration neurons), there are infinite solution weight matrices for the same system of equations defined by $\mathbf{C}$ and $\mathbf{U}$. These solutions lay in a subspace of $\mathbb{R}^N$, of dimension $N - \text{rank}(\mathbf{C})$. The set of all solutions can be obtained by computing the sum between $\mathbf{W}$ and a matrix $\Delta\mathbf{W}$ that satisfies:

$$\mathbf{C}\,\Delta\mathbf{W} = \mathbf{0} \tag{9}$$

$$\Delta\mathbf{W} = \ker(\mathbf{C})\mathcal{M} \tag{10}$$

where $\ker(\mathbf{C})$ is an orthonormal basis of the null space of $\mathbf{C}$, of dimensions $N \text{x} (N - \text{rank}(\mathbf{C}))$, $\mathbf{0}$ is a matrix of zeros, and $\mathcal{M}$ is a linear mapping of dimensions $(N - \text{rank}(\mathbf{C}))\text{x}N_{rec}$. This set of solutions share the same stimulus–response mapping. We say they conform an isofunction space.

In several occasions, networks with $N_{rec} > 2^\tau$ were desired. Hence, we defined $N_{rec} = 2^\tau f_r$, where f_r stands for 'redundancy factor', as the network has f_r-times more neurons than required to solve the s-task with that specific τ.

Computation of fitting times for FTP and SGD. We assessed the efficiency of the FTP algorithm and an SGD algorithm by measuring the time required to find networks of $N_{rec} = 1024$ neurons that solve an s-task with $\tau = 1$ to $\tau = 10$. Since in our neuron model the firing state is a non-differentiable function of the activation states, we relied on a surrogate derivative:

$$\frac{\partial z}{\partial u} := \max\{0, 1 - |\upsilon|\}$$

where $\upsilon = u - \theta$. We took this surrogate derivative from Bellec et al.[15], although we obtained better results not dividing by the threshold, as originally proposed. One response neuron was added, whose output at a given time step t was defined as $o_t = \tanh(\mathbf{w}^{\mathbf{out}}\mathbf{z}_t)$, where $\mathbf{w}^{\mathbf{out}}$ contains the synaptic weights between integration and output neurons. The loss function to minimize was the quadratic error $E = (o_{target}(s) - o)^2/2$, with $o_{target}(s_1) = 1$ and $o_{target}(s_2) = -1$. We implemented back propagation thought time (BPTT), unfolding the network in sequences of 30 timesteps and taking minibatches of 30 sequences. We achieved best performance by initializing $\mathbf{W}^{in}$ and $\mathbf{w}^{\mathbf{out}}$ with zeros, while matrix $\mathbf{W}^{rec}$ was initialized with samples from a standard normal distribution and then dividing it by its eigenvalue of higher absolute value. Adaptive learning rates were implemented thought *Adam*[14], with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and $\alpha = 10^{-4}$. Normalizing gradients to unit length also proved to be helpful in accelerating convergence. Networks were trained until the mean error over the last 10 minibatches was below 0.01. After training, networks were tested with 10,000 stimuli presentations. The maximum testing error across all τ values was 0.03.

Imposing activity constraints. To construct networks with desired FR we generated $\mathbf{U}_{base}$ as previously described, but adjusted the sign of $r(m, i)$ such that, after thresholding, matrix $\mathbf{C}$ had a fraction of ones that equalled the target FR. To induce signal correlation, we scaled vector $\boldsymbol{\Delta}$ by a factor f_{cc}. This manipulation causes neurons to have very different firing rates for s_1 and s_2, leading to an increased signal correlation. By following this procedure, we constructed networks in Fig. 3. The target FR values were taken from the range between 0.1 to 0.9 spikes/time step, in steps of 0.1 spikes/time step. The f_{cc} values were taken from the range between 1 and 10 in unitary steps. A total of 30 networks were constructed for each combination of FR and f_{cc} values within those ranges. For each network, the average FR was computed over the FR of all neurons in the network. Similarly, the average correlation coefficient (CC) was computed from the Spearman correlation coefficient computed for all neuron pairs.

We also employed a genetic algorithm (GA) to evolve a population of matrices $\mathbf{U}_{base}$ to adjust their mean FR and absolute CC. We employed a population of $N_{pob} = 200$ individuals, each one composed of one matrix $\mathbf{U}_{base}$ and one vector $\boldsymbol{\theta}$. For each individual we constructed matrices $\mathbf{U}$ and $\mathbf{C}$, and computed an approximate value of FR and correlation, under the assumption that each network firing state occurred with equal probability. The fitness F of an individual was computed as:

$$F = 1 - \frac{\left|FR - FR_{target}\right| + \left||CC| - \left|CC_{target}\right|\right|}{2}$$

where FR and CC are the firing rate and correlation computed over the network firing states, and FR_{target} and CC_{target} are the firing rate and correlation we want the network to have. If an individual produced an inconsistent system, or a system with not enough network states, its fitness was set to zero. We picked the $T = 0.1N_{pob}$ individuals with the highest fitness as parents. Then, we picked parents at random and mutated $\mathbf{U}_{base}$ by adding gaussian noise to each matrix element, of zero mean and standard deviation $\sigma = 0.1$. One of the individuals of each generation was an unmutated copy of the best individual of the previous generation (elitism). Threshold vectors $\boldsymbol{\theta}$ were not mutated. The GA was run until the average fitness surpassed $F_{target} = 0.95$. Firing rates and correlations shown as black dots in Fig. 3d were computed by running the network constructed from the elite $\mathbf{U}_{base}$ during 30.2^{τ} time steps.

Imposing structural constraints. Solving Eq. (4) gives networks with minimum Frobenius norm. These networks do not comply with basic structural features observed experimentally, such as the low probability of self-connections, or Dale's principle. To impose such structural features, we constructed a matrix $\Delta\mathbf{W}_d$ such that $\mathbf{W}_d = \mathbf{W} + \Delta\mathbf{W}_d$. Matrix $\mathbf{W}_d$ is a matrix which fulfils the desired structural constraints. Most probably $\Delta\mathbf{W}_d$ is not within the null space of $\mathbf{C}$, and thus $\mathbf{W}_d$ will not be a solution to the system defined by $\mathbf{C}$ and $\mathbf{U}$. Hence, we defined a matrix:

$$\Delta\mathbf{W} = \ker(\mathbf{C})\mathcal{M}_{sc} \tag{11}$$

where $\mathcal{M}_{sc} = \ker(\mathbf{C})^{+}\Delta\mathbf{W}_d$, and $\ker(\mathbf{C})^{+}$ is the Moore–Penrose pseudoinverse of $\ker(\mathbf{C})$. Matrix $\mathcal{M}_{sc}$ is a linear mapping that incorporates the desired structural constraints, making $\Delta\mathbf{W}$ the change in matrix $\mathbf{W}$ within the null space of $\mathbf{C}$ that is closest to $\Delta\mathbf{W}_d$, in the least squares sense.

We imposed three structural constraints: no self-connections, Dale's principle, and a certain degree of sparsity. Thus:

$$\Delta\mathbf{W}_d = \Delta\mathbf{W}_{self} + \Delta\mathbf{W}_{Dale} + \Delta\mathbf{W}_{sp}$$

The (i, j) element in matrix $\Delta\mathbf{W}_{self}$ (which deletes self-connections in the integration neurons) was defined as:

$$\Delta\mathbf{W}_{self}(i,j) = \begin{cases} -\mathbf{W}(i,j) & i = j + 2 \\ 0 & i \neq j + 2 \end{cases}$$

where $\mathbf{W}(i, j)$ is the (i, j) element of matrix $\mathbf{W}$.

Matrix $\Delta\mathbf{W}_{Dale}$ was defined as:

$$\Delta\mathbf{W}_{Dale}(i,j) = \begin{cases} 0 & (\mathcal{T}_j = Ex \ \wedge \ w(i,j) > 0) \vee (\mathcal{T}_j = Inh \ \wedge \ w(i,j) < 0) \\ -\mathbf{W}_{c_1}(i,j) & (\mathcal{T}_j = Ex \ \wedge \ w(i,j) < 0) \vee (\mathcal{T}_j = Inh \ \wedge \ w(i,j) > 0) \end{cases}$$

where $\mathbf{W}_{c_1} = \mathbf{W} + \Delta\mathbf{W}_{self}$ and $\mathcal{T}_j \in \{Ex, In\}$ indicates if neuron j was chosen to be excitatory (Ex) or inhibitory (Inh). Matrix $\Delta\mathbf{W}_{Dale}$ sets to zero the synaptic weights that violate Dale's principle. Neuron j was chosen to be excitatory if $\eta_j = \sum_j \mathbf{W}_{c_1}(i,j) > 0$. Otherwise, it was chosen to be inhibitory. If more excitatory/inhibitory neurons were required, neurons with negative/positive η closest to 0 were set as excitatory/inhibitory as needed.

Matrix $\Delta\mathbf{W}_{sp}$ to enforce sparsity was defined as:

$$\Delta\mathbf{W}_{sp}(i,j) = \begin{cases} -\mathbf{W}_{c_2}(i,j) & \left|\mathbf{W}_{c_2}(i,j)\right| < \alpha(sp) \\ 0 & otherwise \end{cases}$$

where $\mathbf{W}_{c_2} = \mathbf{W}_{c_1} + \Delta\mathbf{W}_{Dale}$. The value $\alpha(sp)$ is the sp-percentile of the absolute values in $\mathbf{W}_{c_2}$. In this manner $\Delta\mathbf{W}_{sp}$ will set to zero the weights with the lowest absolute value, such that a sparsity sp is enforced.

The loss function $\mathcal{L}$ at iteration k was defined as the average of the absolute $\Delta\mathbf{W}_d(i, j)$ values:

$$\mathcal{L}(k) = \left\langle \left| \Delta \mathbf{W}_d(i,j) \right| \right\rangle_{i,j}$$

where $\langle\rangle_{i,j}$ stands for the average across indexes (i, j). The structural constraints were imposed through an iterative process, in which neurons were classified as excitatory or inhibitory at each iteration according to their η, a matrix $\Delta\mathbf{W}$ was computed using Eq. (11), and a new $\mathbf{W}$ was obtained. The process was stopped when the loss fell below a desired value e_1, in which case the fitting process was considered successful. The process was also stopped if $(\mathcal{L}(k) - \mathcal{L}(k-1))/\mathcal{L}(k) < e_2$. When this latter condition was met, the fitting process was considered unsuccessful, since the error was not decreasing fast enough and would probably converge to an unacceptable value above zero. We used $e_1 = 10^{-3}$ and $e_2 = 10^{-4}$. If the process was successful, values that violated any of the constraints were set to zero. These values were expected to be small enough since the error was small. We computed:

$$e_{clip} = \left\langle \left| \mathbf{U}_{sc}(i,j) - \mathbf{U}(i,j) \right| \right\rangle_{i,j}$$

where $\mathbf{U}_{sc} = \mathbf{C}\mathbf{W}_{sc}$, with $\mathbf{W}_{sc}$ being the resulting synaptic weight matrix after the constraining process, to verify that the deviation from the original $\mathbf{U}$ was negligible. If the process was unsuccessful, or the clipping error $e_{clip} > 10^{-3}$, then the original $\mathbf{W}$ was considered not to be suitable for the structural fitting.

We measured the efficiency of the process by computing the number of networks generated (*# attempts*) and the running time t expended until reaching the first successfully constrained network. We varied τ from $\tau = 3$ to $\tau = 6$. For each τ we varied the number of integration neurons in steps of 16 neurons, from a minimum number of 4.2^{τ} to the maximum value 2^6. For each combination of τ and neuron number we generated networks with the FTP algorithm, and subjected them to structural constraining (no self-connections, 4:1 Ex:In ratio, and a minimum sparsity $sp = 40\%$). We obtained 10 measurements of *# attempts* and t, from which we computed the mean and SD depicted in Fig. 4e,f.

Network construction from random transition graphs.

To construct random transition graphs that have an associated consistent system, we first defined a row vector $\mathbf{\Delta}$ and a matrix $\mathbf{U}_{lc}$ composed of $M = 2^{\tau}$ row vectors $\mathbf{u}$ such that $\mathbf{u}_{i+1} - \mathbf{u}_i = \mathbf{\Delta}$, where $\mathbf{u}_i$ is the i^{th} row. Here, indexes i are odd numbers between 1 and $M-1$. Next, we constructed matrix $\mathbf{U}$ in a way that ensures that each node in the graph was reachable, meaning that every node had to receive at least one edge. This is equivalent to say that every row in $\mathbf{U}_{lc}$ is found at least once in $\mathbf{U}$. Therefore, we set the first M rows in $\mathbf{U}$ equal to matrix $\mathbf{U}_{lc}$. The remaining rows in $\mathbf{U}$ were taken from $\mathbf{U}_{lc}$, picking $M/2$ pairs of indexes $i, i+1$, choosing i values at random from the set of odd numbers between 1 and $M-1$. In this way, and unlike the transition graphs that solve an s-task, nodes could receive just one edge, or more than two.

So far, if a row vector appeared in matrix $\mathbf{U}$ more than once, then it appeared only in odd rows, or only in even rows, but not in both. This is because indexes were ordered from 1 to M in the first half of $\mathbf{U}$, and ordered in pairs of $i, i+1$ indexes in the second half. The resulting graph would be one in which any given node is reachable as the result of the presentation of either s_1 or s_2, but not from both. In other words, if node b is reachable from node a after s_i presentation, then node b is reachable from node c only after s_i presentation, where c is any other node from which b is reachable. In terms of the colour code of the graph in Fig. 1b, any node receives arrows of the same colour. We wanted graphs as random as possible, so nodes reachable through different stimuli were desired. We defined these nodes as *bicoloured* nodes. In terms of indexes in matrix $\mathbf{U}$, a bicolored node translates into a row vector $\mathbf{u}_i$ that appeared in matrix $\mathbf{U}$ in both odd and even rows. For example, if we had indexes $(1, 2, 3, 4)$ for the first 4 rows of $\mathbf{U}$, with $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{\Delta})$ and $(\mathbf{u}_3, \mathbf{u}_4, \mathbf{\Delta})$ each being linearly combined, then we wanted to change this series to $(1, 2, 2, 4)$, or $(1, 2, 3, 1)$. This requires to generate new linear combinations, in particular, $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_4, \mathbf{\Delta})$ should be linearly combined, for the first example, and $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{\Delta})$ in the second example. Thus, we modified matrix $\mathbf{U}$ to generate $f_{bc}M/4$ bicolored nodes, where f_{bc} stands for 'bicolored fraction' and is a number between 0 and 1. The maximum number of bicolored nodes is $M/4$, since we generated one node for each series of indexes i to $i+3$. Finally, we constructed matrix $\mathbf{Z}$ by thresholding matrix $\mathbf{U}_{lc}$, and then matrix $\mathbf{C}$, which rows were in the form:

$$\begin{pmatrix} \mathbf{y}_1\mathbf{z}_{k(1)} \\ \mathbf{y}_2\mathbf{z}_{k(1)} \\ \vdots \\ \mathbf{y}_1\mathbf{z}_{k(M)} \\ \mathbf{y}_2\mathbf{z}_{k(M)} \end{pmatrix}$$

where $\mathbf{z}_i$ is the ith row vector in matrix $\mathbf{Z}$, and vector k is a permutation of the list of integers from 1 to M.

Following the above procedure, we constructed random transition graphs that satisfied the linear combinations required so that a consistent system of equations could be constructed. Given that these networks do not solve the s-task but are the minimum Frobenius norm solution to a random transition graph, we call them *Frobenius* (*F*) networks. The procedure avoids index sequences like (1221), since this ordering gives a consistent system only if $\mathbf{\Delta} = \mathbf{0}$, in which case stimuli cannot be distinguished by the network. The procedure also avoids index sequences of the type (11) (one node leads to another node through both stimuli, s_1 and s_2). If this were the case, one possibility is that s_1 and s_2 produce the same u values (u being an element of vector $\mathbf{u}$). Therefore, $\mathbf{\Delta}$ is a vector of zeros and stimuli cannot be discriminated. Another possibility is that stimuli lead to different vectors $\mathbf{u}$, but these vectors in turn lead to the same vector $\mathbf{z}$ after thresholding. This situation is possible, but would require careful selection of u values in relation to θ, and for this it was avoided.

We constructed networks that followed random transition graphs and compared their properties with the properties of networks that solved the s-task. In particular, we measured the reciprocity of the network, defined as the Spearman correlation between weights of incoming and outgoing synapses. Reciprocity was computed over matrix $\mathbf{W}^{norm}$, a normalized version of the synaptic weights constructed by taking the absolute values of $\mathbf{W}$ and scaling them between 0 and 1. Since imposing structural constraints like Dale's principle, or sparsity, may generate many zero-valued weights, reciprocity was computed for synaptic weight pairs such that both w_{ij}^{norm} and w_{ji}^{norm} were both non zero.

For each network generated to solve the s-task we also picked a network from its isofunction space, that is, from the set of all networks that had the same stimulus–response mapping (as described above). We refer to these networks as *Task* (T) networks, since they solve the s-task but they are not the minimum Frobenius norm solution. The linear mapping $\mathcal{M}$ in Eq. (10) has entries $\mathcal{M}(i,j) = r_{i,j} \max_{i,j} \left(\mathbf{W}(i,j)\right)$, where $r_{i,j}$ is a random number, different for each entry, sampled uniformly from the $[-1, 1]$ interval, and $\mathbf{W}$ is the synaptic weight matrix from which an isofunction network is desired. We defined mapping $\mathcal{M}$ in this way to obtain isofunction networks with synaptic weight values within the range of the weights in the original network.

In addition, for each network that solves the s-task we constructed an isofunction network with structural constraints: no self-connections and Dale's principle with excitatory and inhibitory neurons in equal numbers.

Received: 8 July 2020; Accepted: 27 January 2021
Published online: 15 February 2021

References

1. Kriegeskorte, N. & Douglas, P. K. Cognitive computational neuroscience. *Nat. Neurosci.* **21**, 1148–1160 (2018).
2. Bassett, D. S., Zurn, P. & Gold, J. I. On the nature and use of models in network neuroscience. *Nat. Rev. Neurosci.* **19**, 566–578 (2018).
3. Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, X. J. Task representations in neural networks trained to perform many cognitive tasks. *Nat. Neurosci.* **22**, 297–306 (2019).
4. Strata, P. & Harvey, R. Dale's principle. *Brain Res. Bull.* **50**, 349–350 (1999).
5. Song, H. F., Yang, G. R. & Wang, X. J. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS Comput. Biol.* **12**, 1–30 (2016).
6. Richards, B. A. *et al.* A deep learning framework for neuroscience. *Nat. Neurosci.* **22**, 1761–1770 (2019).
7. Barrett, D. G., Morcos, A. S. & Macke, J. H. Analyzing biological and artificial neural networks: challenges with opportunities for synergy?. *Curr. Opin. Neurobiol.* **55**, 55–64 (2019).
8. Bengio, Y. & LeCun, Y. Scaling Learning Algorithms Towards AI. In *Large-Scale Kernel Machines* (eds Bottou, L. *et al.*) 321–358 (The MIT Press, Cambridge, 2007).
9. Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nat. Neurosci.* **18**, 1025–1033 (2015).
10. Maheswaranathan, N., Williams, A. H., Golub, M. D., Ganguli, S. & Sussillo, D. Universality and individuality in neural dynamics across large populations of recurrent networks. *Adv. Neural Inf. Process. Syst*. 15629–15641 (2019)
11. Penrose, R. & Todd, J. A. On best approximate solutions of linear matrix equations. *Math. Proc. Camb. Philos. Soc.* **52**, 17–19 (1956).
12. Neri, F. *Linear algebra for computational sciences and engineering* (Springer, Berlin, 2016).
13. Barone, P. & Joseph, J. Prefrontal cortex and spatial sequencing in macaque monkey. *Exp. Brain Res.* **78**, 447–464 (1989).
14. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. 1–15 (2014). https://doi.org/10.1145/1830483.1830503
15. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R. & Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. *Adv. Neural Inf. Process. Syst.* **2018**, 787–797 (2018).
16. Lennie, P. & Place, W. The cost of cortical computation. *Curr. Biol.* **13**, 493–497 (2003).
17. Mizuseki, K. & Buzsáki, G. Preconfigured, skewed distribution of firing rates in the hippocampus and entorhinal cortex. *Cell Rep.* **4**, 1010–1021 (2013).
18. Schneidman, E., Berry, M. J., Segev, R. & Bialek, W. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature* **440**, 1007–1012 (2006).
19. Bekkers, J. M. Neurophysiology: are autapses prodigal synapses?. *Curr. Biol.* **8**, 52–55 (1998).
20. Calamai, P. H. & Moré, J. J. Projected gradient methods for linearly constrained problems. *Math. Program.* **39**, 93–116 (1987).
21. Song, S., Sjöström, P. J., Reigl, M., Nelson, S. & Chklovskii, D. B. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol.* **3**, e68 (2005).
22. Brunel, N. Is cortical connectivity optimized for storing information?. *Nat. Neurosci.* **19**, 749–755 (2016).
23. Schulz, D. P. A., Sahani, M. & Carandini, M. Five key factors determining pairwise correlations in visual cortex. *J. Neurophysiol.* **114**, 1022–1033 (2015).
24. de la Rocha, J., Doiron, B., Shea-Brown, E., Josić, K. & Reyes, A. Correlation between neural spike trains increases with firing rate. *Nature* **448**, 802–806 (2007).
25. Constantinidis, C. & Goldman-Rakic, P. S. Correlated discharges among putative pyramidal neurons and interneurons in the primate prefrontal cortex. *J. Neurophysiol.* **88**, 3487–3497 (2002).
26. Alemi, A., Baldassi, C., Brunel, N. & Zecchina, R. A three-threshold learning rule approaches the maximal capacity of recurrent neural networks. *PLoS Comput. Biol.* **11**, 1–23 (2015).
27. Such, F. P. *et al.* Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning (2017). http://arxiv.org/abs/1712.06567.
28. Salimans. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *Proc. 4th IEEE Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2011* 476–485 (2011). https://doi.org/10.1109/ICSTW.2011.58
29. Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354 (2019).
30. Silver, D. *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science (80-)* **362**, 1140–1144 (2018).
31. Kojima, S. & Goldman-Rakic, P. S. Delay-related activity of prefrontal neurons in rhesus monkeys performing delayed response. *Brain Res.* **248**, 43–50 (1982).
32. Guo, Z. V. *et al.* Maintenance of persistent activity in a frontal thalamocortical loop. *Nature* **545**, 181–186 (2017).
33. Orhan, A. E. & Ma, W. J. A diverse range of factors affect the nature of neural representations underlying short-term memory. *Nat. Neurosci.* **22**, 275–283 (2019).
34. Murray, J. D. *et al.* Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proc. Natl. Acad. Sci. U.S.A.* **114**, 394–399 (2017).

35. De Vries, M. H., Christiansen, M. H. & Petersson, K. M. Learning recursion: multiple nested and crossed dependencies. *Biolinguistics* **5**, 10–35 (2011).
36. Honey, C. J., Thivierge, J. P. & Sporns, O. Can structure predict function in the human brain?. *Neuroimage* **52**, 766–776 (2010).
37. Bullmore, E. & Sporns, O. Complex brain networks: Graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **10**, 186–198 (2009).
38. Vázquez-Rodríguez, B. *et al.* Gradients of structure–function tethering across neocortex. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 21219–21227 (2019).
39. Kording, K. P. Bayesian statistics: relevant for the brain?. *Curr. Opin. Neurobiol.* **25**, 130–133 (2014).
40. Knill, D. C. & Pouget, A. The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends Neurosci.* **27**, 712–719 (2004).
41. Ellefsen, K. O., Mouret, J. B. & Clune, J. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput. Biol.* **11**, 1–24 (2015).

Acknowledgements
This research was supported by grants from Agencia Nacional de Promoción Científica y Tecnológica (ANPCYT) PICT 2016-2145 and PICT 2017-3208) and from Universidad de Buenos Aires (UBACYT 20020170100568BA). The work of Camilo J. Mininni was supported by a postdoctoral fellowship from ANPCYT.

Author contributions
CJM conceived the project, implemented the algorithms and wrote the initial draft. CJM and BSZ reviewed and edited the final manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to C.J.M.