nature neuroscience

Article

https://doi.org/10.1038/s41593-024-01852-8

Neural mechanisms of relational learning and fast knowledge reassembly in plastic neural networks

Received: 4 September 2023

Accepted: 15 November 2024

Published online: 15 January 2025

Check for updates

Thomas Miconi [1,2] & Kenneth Kay [3,4,5]

Humans and animals have a striking ability to learn relationships between items in experience (such as stimuli, objects and events), enabling structured generalization and rapid assimilation of new information. A fundamental type of such relational learning is order learning, which enables transitive inference (if A > B and B > C, then A > C) and list linking (A > B > C and D > E > F rapidly 'reassembled' into A > B > C > D > E > F upon learning C > D). Despite longstanding study, a neurobiologically plausible mechanism for transitive inference and rapid reassembly of order knowledge has remained elusive. Here we report that neural networks endowed with neuromodulated synaptic plasticity (allowing for self-directed learning) and identified through artificial metalearning (learning-to-learn) are able to perform both transitive inference and list linking and, further, express behavioral patterns widely observed in humans and animals. Crucially, only networks that adopt an 'active' solution, in which items from past trials are reinstated in neural activity in recoded form, are capable of list linking. These results identify fully neural mechanisms for relational learning, and highlight a method for discovering such mechanisms.

How do we gain broad knowledge from limited experience? Humans and animals can generalize and learn rapidly from limited experience, yet how these abilities are implemented in the brain remains an open question[1–3]. A possibly fundamental basis is the learning of relations between different experiences (such as 'stronger than', 'next to', 'same as' and 'part of') because relations specify a particular structure for subsequent generalization and inference. Additionally, such relational learning is thought to enable the construction and rapid modification of internal models of the world—variously termed 'relational memory', 'schemas' and 'cognitive maps'—which are increasingly recognized as essential to cognition[4,5].

One fundamental type of relational learning is order learning, which is broadly applicable to a range of concepts such as space, time, rank and number[6,7]. Critically, understanding order confers the ability to perform transitive inference, the ability to infer relative order between items not previously observed together. For example, after learning to choose between two stimuli that are 'adjacent' in an underlying ordered series—or 'list' (A > B, B > C, C > D, etc., where '>' designates 'chosen over')—humans and animals can choose correctly on 'nonadjacent' pairs not previously seen (A > C, B > D, etc.). Transitive inference has been observed in a wide range of species including humans, monkeys, rodents, birds and insects[6,8].

Remarkably, animals and humans can also rapidly learn a global ordering across separately learned lists, when presented with limited new information. That is, after separately learning lists (for example, A > B > C and D > E > F), participants that learn a linking pair (here C > D)

[1]ML Collective, San Francisco, CA, USA. [2]The Astera Institute, Berkeley, CA, USA. [3]Mortimer B. Zuckerman Mind Brain Behavior Institute, Columbia University, New York City, NY, USA. [4]Center for Theoretical Neuroscience, Columbia University, New York City, NY, USA. [5]Grossman Center for the Statistics of Mind, Columbia University, New York City, NY, USA. e-mail: thomas.miconi@gmail.com; kaykenneth@gmail.com

https://doi.org/10.1038/s41593-024-01852-8

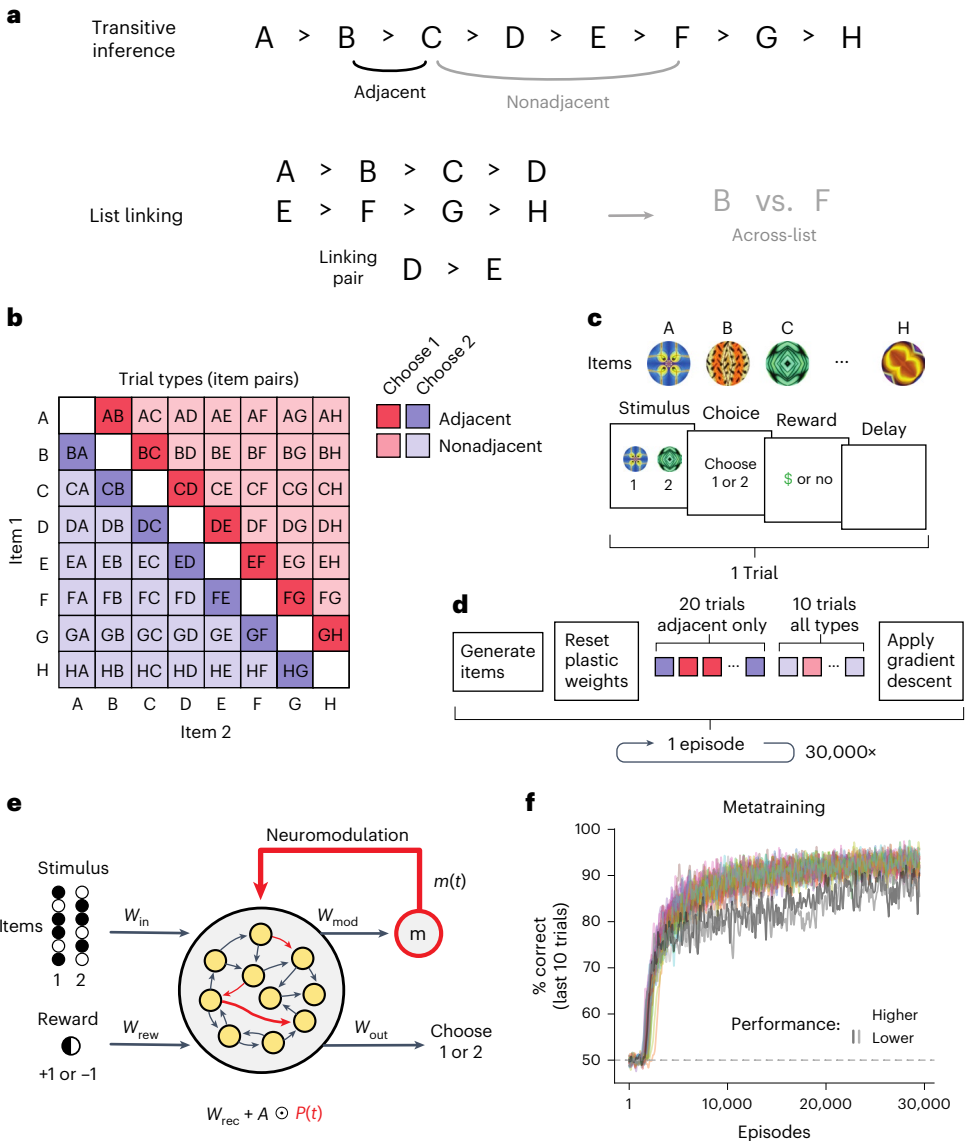


Fig. 1 | Tasks, model and overall performance. a, Schematic of task paradigms. Letters (A, B, C, etc.) refer to arbitrary stimuli (items). In transitive inference, participants learn to choose between 'adjacent' pairs of items (such as B over C) in accordance with an underlying ordering (list) and are subsequently tested on nonadjacent pairs (such as C over F). In list linking, participants first learn to choose between adjacent pairs of items within two separate lists (such as A to D and E to H), then encounter a 'linking pair' (here D versus E), after which participants are tested on across-list pairs (such as B versus F). **b**, Table of trial types. Each trial type is defined by the pair of items presented in the trial (identity of item 1 and item 2). The correct choice is the higher-ranked item (item 1 versus item 2). **c**, Trial structure. Each trial is composed of four time steps as follows: stimulus presentation (stimulus), response (choice), feedback signal (reward) and delay. For illustration, each step is depicted as would be presented in a real-world experiment, with random fractal images as items; actual stimuli are binary vectors, randomly generated for each episode. **d**, Episode structure. Each episode consists of 20 trials with only adjacent pairs, followed by 10 trials with all possible pairs. Plastic weights are reset between episodes. **e**, Schematic of the neural model. The model is a recurrent network augmented with plastic weights controlled by self-generated neuromodulation. See 'Main' and Methods for full description. **f**, Model performance across metatraining. Plotted is the mean performance on the last ten trials of episodes (30,000 total) for 30 separate metatraining runs. Curves smoothed with a boxcar filter of width 10. Here two runs show consistently lower performance, suggestive of a different task solution.

can then immediately infer across-list pairs (for example, B > E). This ability, known as 'list linking'[8–10], demonstrates not only rapid assimilation of new information but also the fast 'reassembly'[10] of existing knowledge—after the presentation of the linking pair, participants must somehow reorganize their existing representations of previously learned list items not in the linking pair.

Numerous models of order learning have been proposed to explain transitive inference and list linking (see refs. 8,11 for reviews). However, currently, no biologically plausible neural model of order learning reproduces the various behavioral patterns commonly reported in experimental work[11]. Recent experimental findings suggest that relational learning reorganizes neural representations of linked items, potentially enabling inferential responses as a result[10,12], but leave open the question of how this reorganization occurs. In modeling work, neural networks can be explicitly trained (through an artificial optimization algorithm) to learn a given list or set of lists, including list linking; these networks exhibit internal representations and behavioral patterns consistent with experimental results[10,13–15]. However, because these approaches use hand-designed, nonbiological learning algorithms (usually based on backpropagation), or leave the learning process largely unspecified, they do not explain how order learning can be implemented in biologically plausible neural processes.

https://doi.org/10.1038/s41593-024-01852-8

To address this challenge, here we take a different approach—rather than use a prespecified learning algorithm to train a neural system on one (or several) particular lists, we instead metatrain a learning neural system to be able to learn arbitrary new lists. Building upon previous work in metalearning, or 'learning-to-learn'[16–20], we metatrained neural networks endowed with biologically plausible (Hebbian) synaptic plasticity and self-controlled neuromodulation. These networks are able to actively modify their own connectivity in response to external input (such as sensory stimuli and rewards), thereby enabling autonomous, self-directed learning across trials. Importantly, the learning algorithm that these networks employ is not manually specified in advance, but instead entirely 'discovered' by the metatraining process.

Results

Model overview

We metatrain a recurrent neural network, endowed with synaptic plasticity and neuromodulation, to be able to autonomously learn an arbitrary implicit serial order for a set of arbitrary stimuli, over the course of several trials in which pairs of stimuli are presented, following the classic paradigm of transitive inference[6,8–11]. Schematics of the task and model are shown in Fig. 1a–e.

The task is organized into episodes, each of which is composed of a number of trials (Fig. 1b–d). In each episode, the agent is tasked with learning an implicit ordering over completely new random stimuli (items A, B, C, etc.; Fig. 1a). The stimuli are high-dimensional binary vectors (size = 15; we obtained similar results with one-hot vectors), randomly generated anew for each episode. The number of items in the list to be learned varies from episode to episode between four and nine (inclusive); all results below use eight items. Each episode consists of 30 trials, where each trial consists of the simultaneous presentation of two stimuli, a binary response by the agent (choose item 1 or item 2), and a binary feedback signal $R(t)$ indicating whether the response was correct or not (that is, whether the chosen stimulus was in fact the higher ranking of the two in the overall ordering).

The first 20 trials of an episode include only adjacent pairs, that is, pairs of stimuli with adjacent ranks in the series. The last ten trials include all possible pairs (excluding identical pairs such as AA or BB), unless specified otherwise. Performance in a given episode is assessed as the proportion of correct responses over the last ten trials.

Within each episode, the agent undergoes synaptic changes (plasticity), gated by a self-generated modulatory signal (Fig. 1e and Methods). From these synaptic changes, a successful agent would be able to learn the correct ordering of all stimuli over the course of the episode. To generate such agents, after each episode, we apply gradient descent to the structural parameters of the network (the base weights and plasticity parameters, as well as other parameters; see below), to improve within-episode plasticity-based learning. The loss optimized by gradient descent is the total reward obtained over the whole episode.

The gradients are computed by a simple reinforcement learning algorithm, namely, Advantage Actor Critic (A2C)[21], which is readily interpretable as modeling dopamine-based learning in the brain[19]. See Methods and Supplementary Note 5 for details.

Metalearning discovers solutions for transitive inference

Across multiple runs, the metatraining procedure described above consistently generated a high-performing learning agent (Fig. 1f). Subsequent analysis of trained networks indicated that all runs that reach the higher performance level yield a similar solution, which is presented below. Interestingly, some runs yielded agents that performed at relatively lower levels (Fig. 1f, gray traces), suggestive of a different solution in these agents. We return to these lower-performing agents in a subsequent section ('A suboptimal solution cannot perform list linking'); in the following sections, we analyzed a single network representative of the higher-performing agents (unless stated otherwise).

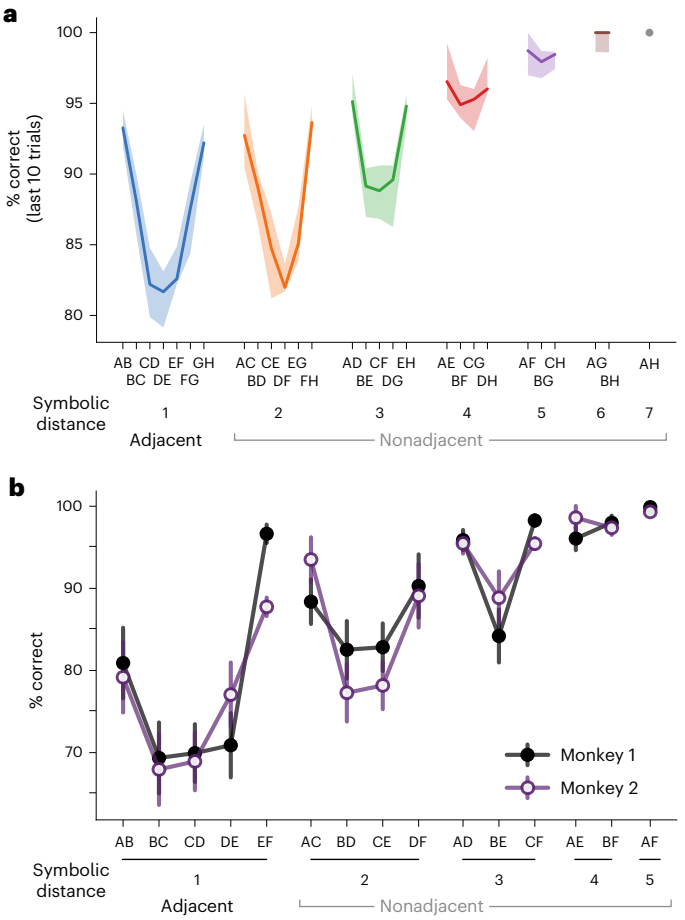


Fig. 2 | Reproduction of experimentally observed behavioral patterns. **a**, Performance (% correct) by trial type. Trial types (item pairs; Fig. 1b) are arranged according to the difference in rank between items ('symbolic distance'). Network responses were taken exclusively from the last ten trials of episodes, and for 2,000 separately run episodes split into ten subsets of 200 episodes each. Solid lines and shaded areas indicate median and interquartile range of mean performance for each pair across subsets. Note the higher performance for higher symbolic distance pairs (symbolic distance effect), and higher performance for pairs that include highest- or lowest-ranked items (here A or H (end-anchor effect)). **b**, Experimental data from monkeys (redrawn with permission from Brunamonti et al. (Fig. 1 of ref. 22)), displaying the same behavioral patterns (mean and s.e. of the mean for success rate over 30 sessions, each with at least 14 test trials for each pair, for either participant). See also Fig. 2 of ref. 11 and human data in Supplementary Fig. 5 of ref. 14.

Metatrained networks reproduce experimentally observed behavioral patterns

To assess the behavior of a successful learning network, we ran a single episode (20 learning trials with only adjacent pairs, followed by 10 test trials with all possible pairs) with eight stimuli (A to H) randomly generated for that episode. Figure 2a reports performance on the last ten (test) trials of this episode, separately for each pair, with pairs arranged according to 'symbolic distance', that is, the absolute difference in rank between items in the pair.

The network expressed two classic behavioral patterns characteristic of humans and animals in transitive inference experiments[8,11]. First, we observed the so-called symbolic distance effect[6,8,10,11,22]—performance is higher for the pairs with higher symbolic distance (upward trend from left to right, Fig. 2a; compare with Fig. 2b, showing monkey experimental data). In particular, performance is the lowest for adjacent pairs (AB, BC, etc.). This effect is especially notable because the first 20 trials involve only adjacent pairs (as done in experiments, serving as a 'training set' for the task)—performance

https://doi.org/10.1038/s41593-024-01852-8

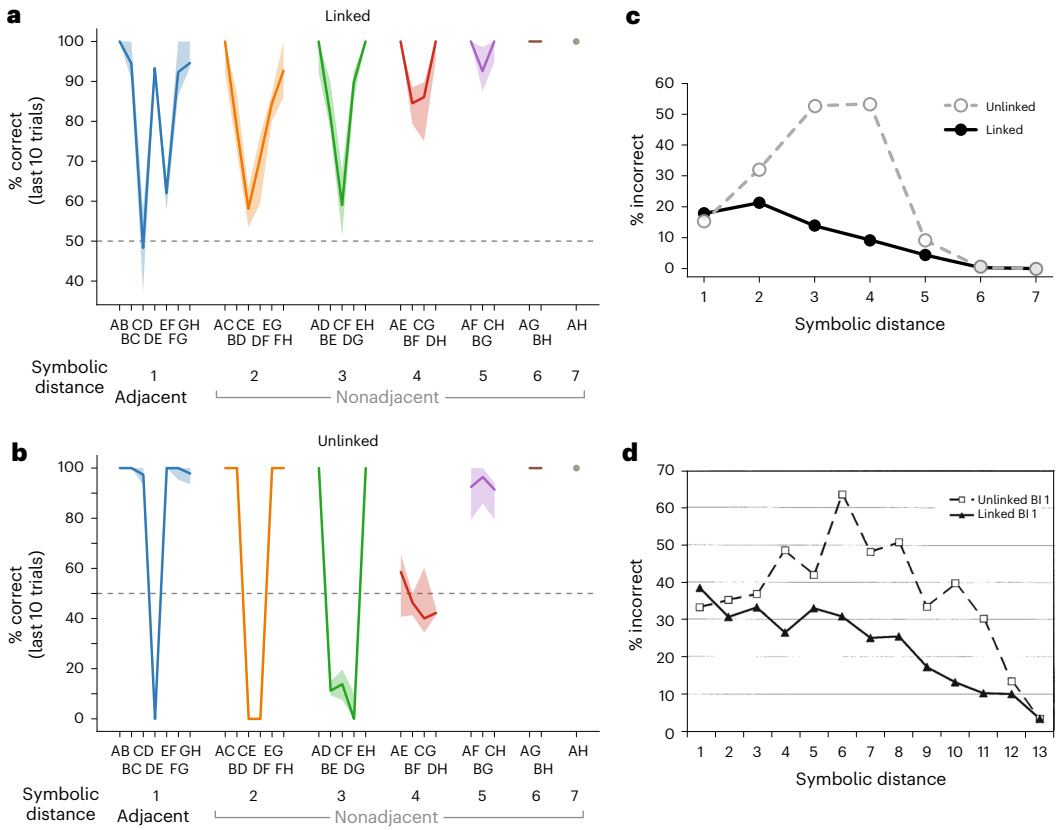


Fig. 3 | List-linking performance and comparison to experimental data. **a**, Performance (% correct) by trial type for episodes testing list linking. The network was presented with trials for learning ABCD, EFGH and DE, then tested on a single trial of any type (any item pair). Performance was measured solely for the final test trial. Conventions are the same as in Fig. 2. Note the relatively lower performance for pairs immediately adjacent to the linking pair (here CD and EF), as reported in experiments[9]. **b**, Performance in unlinked ('sham') condition. Instead of the linking pair DE, the network was presented with EF. **c**, Symbolic distance effect across lists, in linked and unlinked ('sham') conditions. Data are the same as in **a** and **b**. Note that the linked condition produces a symbolic distance effect that is largely monotonic, while the sham condition (unlinked) produces a bump in error for intermediate symbolic distances. **d**, Monkey experimental data, following the basic conventions in **c**, from ref. 24. See also Fig. 4b of ref. 10 for human experimental data.

is thus worse on the 'training set' compared to the 'test set', contrary to standard expectation.

Second, we observe an end-anchor effect[6,11,14,22] (also known as 'serial position effect')—performance is consistently higher for item pairs that involve the highest- or lowest-ranked items rather than other pairs of equivalent symbolic distance (for example, performance on AC and FH is higher than performance on CE or DF). This effect is seen as a U shape for sets of pairs of the same symbolic distance (Fig. 2a compared with Fig. 2b; refs. 8,11). Thus, the network successfully demonstrates transitive inference and reproduces behavioral patterns consistently observed in animal and human experiments.

Notably, in additional experiments, we found that the network is robust to so-called 'massed presentation' of one single pair, which is known to disrupt performance in certain learning models of transitive inference but not in living subjects[11,23] (see Supplementary Note 9 for details).

Metatrained networks rapidly reassemble existing knowledge

Monkeys and humans can rapidly 'link' separately learned lists after learning an item pair relating the two lists. That is, after learning A > B > C > D and E > F > G > H separately, and then learning D > E, they can quickly infer ordering across the entire joint list (C > F, B > G, etc.)[8–11]. This list linking ability implies that the presentation of a pair can affect the subjective ranking not just of items in the linking pair (here D and E) but also of other previously learned items not shown in the current trial.

We ran the metatrained network on ten trials using adjacent pairs from ABCD, then ten trials using adjacent pairs from EFGH, and then finally four trials with D and E. Then we estimated performance on a single 'test' trial, which could use any pair from the whole ABCDEFGH ordering. This was repeated over 2,000 runs, again with different randomly generated stimuli for each run. Results for the last 'test' trial are shown in Fig. 3a. Examination of performance on pairs including items from both sublists (for example, CE, CF, BG, etc.) confirmed that the network successfully linked the two lists into a coherent global ordering. Interestingly, performance was consistently poor for the pairs immediately adjacent to the linking pair, especially from the earlier-learned list (here CD), a pattern reported in monkey experiments (Table 2 of ref. 9).

A control experiment in which a different, nonlinking pair (EF) was shown for one trial only (instead of four trials with the linking pair DE, as above) produced no evidence of list linking (unlinked or 'sham' condition; Fig. 3b). In this case, performance was far below chance on pairs consisting of items whose rank within their respective lists conflicted with their overall rank in the global list (for example, for pair CE, E has high rank within its own list EFGH, while C has low rank within-list ABCD, yet C is higher than E in the overall combined list ABCDEFG). This suggests that the rank of an item generalizes across unlinked lists; such transfer of rank across lists is also observed in experiments[11,24]. This effect produces a characteristic bump in error rate for intermediate symbolic distances with nonlinked lists but not with linked lists (because pairs where within-list and across-list ranks conflict are more likely to have intermediate symbolic distance), which the metatrained network also reproduces (Fig. 3c,d).

https://doi.org/10.1038/s41593-024-01852-8

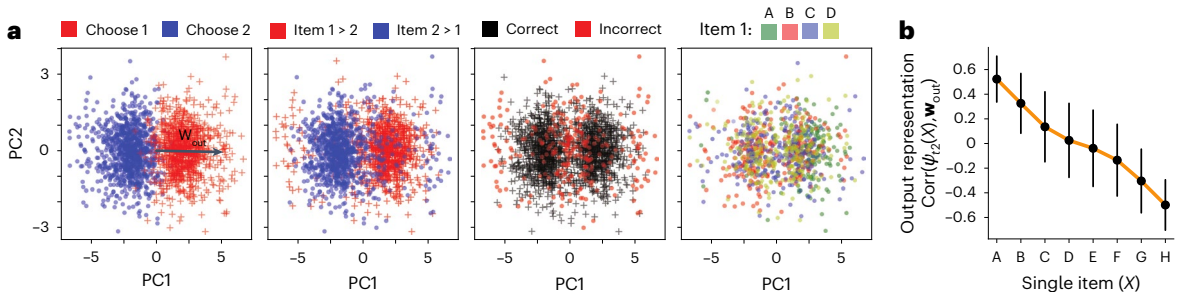


Fig. 4 | A simple representational scheme for transitive inference. Neural activity taken from trial 20 (first trial with nonadjacent pairs) across 2,000 separately run episodes (items generated independently for each episode). **a**, Principal component analysis (PCA) of neural activity. Each data point corresponds to $\mathbf{r}(t)$ at time step 2 of an individual trial. All plots show the same data, colored according to different trial type categories. Left, network response (choose item 1 vs. choose item 2); middle-left, correct choice (item 1 versus item 2); middle-right, correct versus incorrect; right, identity of item 1 (only four items shown for clarity). Note that PC1 is aligned with $\mathbf{w}_{out}$ and effectively distinguishes between the different responses of network (left). **b**, Encoding of item rank. Alignment (correlation) of each item's step 2 neural representation $\boldsymbol{\psi}_{t2}(X)$ with the output weight vector $\mathbf{w}_{out}$. Mean ± s.d. over episodes. Note the monotonic ordering (ranking) of items.

Neural mechanisms of transitive inference

A simple representational scheme for transitive inference

To understand the trained network's operation, we first examined network activity with principal component analysis (PCA). $\mathbf{r}(t)$ is the vector of neural activity at a given time (one element per neuron; Methods). We observed that the first principal component (PC1) of network activity $\mathbf{r}(t)$ at time step 2 of trial 20 (the last trial with adjacent-only pairs) is strongly aligned with the vector of output weights $\mathbf{w}_{out}$, with correlation $r > 0.9$ (Fig. 4, left). As shown by the segregation of trials by the network's choice in Fig. 4a (red vs. blue points, left), position along this axis largely determines the network's choice response across trials. This is as expected because network response is read as the projection of $\mathbf{r}(t = 2)$ onto output weights (Methods).

Further, we found that $\mathbf{r}(t = 2)$ did not appear to contain information about the identity or rank of either individual item in the pair. Using various classification methods, we failed to reliably decode the rank of either the first or the second item from neural data at step 2 (Supplementary Fig. 5). Furthermore, neural activity vectors were not consistently separated by the rank of the first item (Fig. 4, right).

To understand how pairs of items are represented, we first examined the representations of isolated single items. We presented each stimulus $X \in$ (A, B, C, D, E, F, G, H) to the network as item 1 in isolation (not paired with any other) and observed the resulting neural activity $\mathbf{r}(t)$ at time steps 1 and 2. When showing item X in isolation, we denote $\mathbf{r}(t = 1)$ with the symbol $\boldsymbol{\psi}_{t1}(X)$ (the feedforward, step 1 representation of X, determined solely by the fixed, nonplastic input weights) and $\mathbf{r}(t = 2)$ with the symbol $\boldsymbol{\psi}_{t2}(X)$ (the learned, step 2 representation, produced by applying one step of recurrence through the learned, plastic weights to $\boldsymbol{\psi}_{t1}(X)$, and which determines the network's response for this trial).

We found that the plasticity-learned representation $\boldsymbol{\psi}_{t2}(X)$ of each stimulus X aligns with the network's decision axis (the output weight vector $\mathbf{w}_{out}$) in proportion to item rank. That is, $\boldsymbol{\psi}_{t2}(A)$ has large positive correlation with $\mathbf{w}_{out}$, $\boldsymbol{\psi}_{t2}(H)$ has large negative correlation with $\mathbf{w}_{out}$ and intermediate items follow a monotonic progression (Fig. 4b).

In the actual task, stimuli are not presented in isolation, but in pairs. How are those learned representations of single items combined to represent pairs of items? Examining input weights W_{in}, we found that the input weights for the two items in the pair (that is, items 1 and 2; Fig. 1e) are strongly anticorrelated ($r \approx -0.9$). As a result, a given item's representation when shown as item 2 is essentially the negative of its representation when shown as item 1. Therefore, when a pair of items (X, Y) is presented as input, the network automatically computes a subtraction between the representations of both items ($\mathbf{r}(t = 1) \approx \boldsymbol{\psi}_{t1}(X) - \boldsymbol{\psi}_{t1}(Y)$). At the next time step, application of the recurrent weights approximately transforms this subtraction into $\boldsymbol{\psi}_{t2}(X) - \boldsymbol{\psi}_{t2}(Y)$ (neglecting the nonlinearity). Because $\boldsymbol{\psi}_{t2}(X)$'s alignment with the output weight vector $\mathbf{w}_{out}$ is proportional to item rank, the alignment of $\mathbf{r}(t = 2) \approx \boldsymbol{\psi}_{t2}(X) - \boldsymbol{\psi}_{t2}(Y)$ with $\mathbf{w}_{out}$, which determines the network's response, is proportional to the difference in rank between items X and Y.

This scheme provides a simple, intuitive mechanism for transitive inference—once the correct representation $\boldsymbol{\psi}_{t2}(X)$ for each individual item X has been learned, the subtractive operation immediately generalizes to nonadjacent pairs. Furthermore, more distant pairs imply a larger difference in the projection of either individual item along the decision axis; this suffices to cause a symbolic distance effect, where more distant item pairs are more accurately judged. A similar subtractive process has also been observed in models trained by abstract algorithms[13–15].

We note that the anticorrelated input weights for the two items in a stimulus pair were not specified by design, but emerged as a result of metatraining. We also note that these anticorrelated inputs are not strictly necessary for success—similar performance and abilities are obtained if W_{in} is frozen to its initial random values (not updated during metatraining) instead (Supplementary Note 6).

Representation learning

How does the metatrained network learn such order-encoding representations over the course of an episode? To investigate this how, we first assessed the dynamics of the neuromodulatory output $m(t)$ as this output is a simple, one-dimensional signal with a critical role in the network's learning (Methods). As shown in Fig. 5a,b, across trials, $m(t)$ at time step 3 (corresponding to the time of reward delivery for this trial) is consistently negative, regardless of reward received. By contrast, $m(t)$ at time step 4 (corresponding to the delay between trials) is strongly dependent on reward received at time step 3, being highly negative for negative rewards (incorrect trials) but positive or near zero for positive rewards (correct trials; Fig. 5c). This suggests that neuromodulatory outputs at time steps 3 and 4 of a given trial have different roles in learning.

We then examined changes in the plastic weights (and the resulting representations) at each successive time step of a trial. Reasoning that error trials would be most useful in clarifying how the network learns, we selected runs in which the network sees the pair DE or ED at trial 20, but not in any previous trial (we focus on DE for illustration only; similar results hold for other pairs; Extended Data Fig. 2). This presentation of a new pair of items with intermediate ranks, not previously encountered, invariably leads to an erroneous response in this trial. At this stage, the network has only encountered D and E as the lowest- and highest-ranked items, respectively, within two separate sublists (A–D and E–H) misrepresenting their true order, D > E. We sought to characterize the actual representational changes resulting from this error.

https://doi.org/10.1038/s41593-024-01852-8

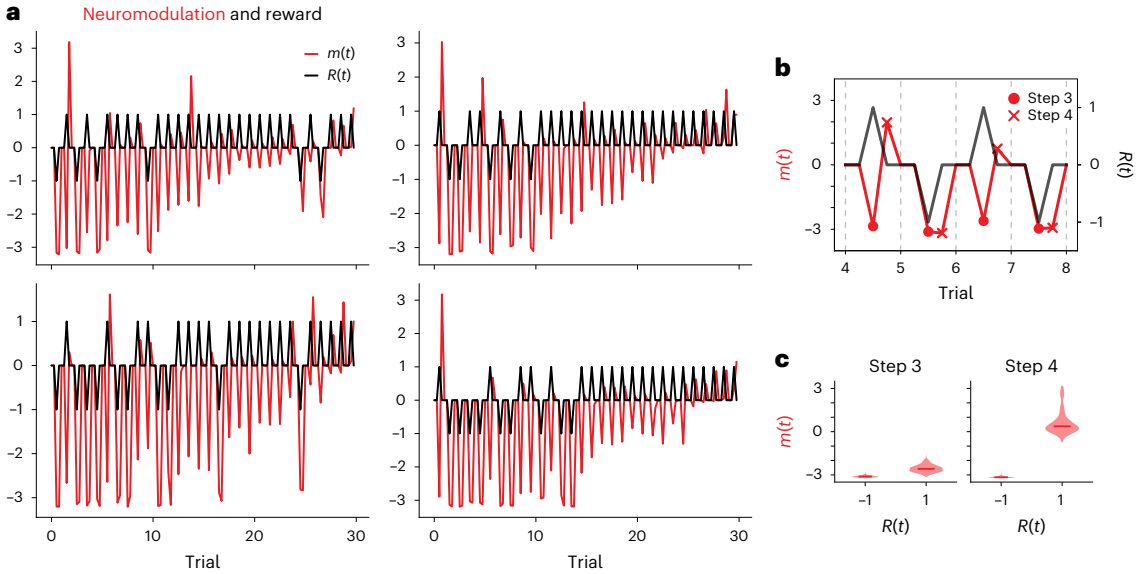


Fig. 5 | Two distinct patterns of neuromodulation. a, Neuromodulatory output $m(t)$ and reward signal $R(t)$ for four example episodes (each episode having independently generated items). For clarity, $m(t)$ is plotted as 0 for the first two time steps of each trial, when its value is irrelevant ('Main' and Methods). Each 'spike' in the black (reward) curve corresponds to the reward delivered at step $t = 3$ of one trial (positive or negative). **b**, Expanded view of trials 4–8 from the top-right example in **a**. Note how $m(t = 3)$ (red circle) is consistently negative independently of reward sign, while $m(t = 4)$ (red cross) is negative for negative rewards and positive or near 0 for positive reward. **c**, Violin plot of $m(t = 3)$ and $m(t = 4)$ for an early trial (trial 5) across 2000 separately run episodes, shown separately for correct (reward = +1) versus incorrect (reward = −1) responses.

In particular, we computed the network's learned (step 2) representations of each isolated item $\boldsymbol{\psi}_{t2}(X)$. Moreover, we computed this representation across time in the trial, using frozen plastic weights $P(t)$ extracted at each of the four successive time steps (of trial 20). This allowed us to observe $\boldsymbol{\psi}_{t2}(X)$ as it was represented by the network at each specific time step of the trial. Then, we computed the correlation of each of these successive representations with the output weight vector $\mathbf{w}_{\text{out}}$, as in Fig. 4b, thereby enabling us to observe how the network's estimated rank of item X changes across time steps as a result of plasticity (Fig. 6, top row).

We found that plasticity in step 3 (when the reward signal is delivered to the network) produces small weight changes restricted to items D and E (Fig. 6, bottom row, third column). By contrast, step 4 produces not only relatively larger changes for items D and E, but, crucially, a substantial change (of appropriate sign) for items C and F (Fig. 6, bottom row, fourth column), neighboring items not presented in the trial. This extension of representation changes to additional items serves to 'reassemble' previously separate orderings into a single global ordering (Fig. 6, top row, compare right (step 4) and middle-right (step 3)), consistent with the network's ability to perform list linking.

Reinstatement of recoded representations supports learning

How do these plastic changes occur mechanistically? In the model, plasticity at time step k can only access activity over the preceding two time steps (equations (5) and (7)). This appears to pose a problem, as stimuli are presented at time step 1, yet the crucial plasticity (as shown above) occurs at time step 4. This problem implies that the network can somehow reinstate representations of relevant stimuli (including nonpresented items) at time step 2, thus enabling the appropriate plasticity to take place at time step 4.

However, in initial analyses, we failed to find reinstatement of original (feedforward) stimulus representations. Feedforward representations $\boldsymbol{\psi}_{t1}(X)$ were strongly present in neural activity at time step 1 (that is, upon stimulus presentation), as expected, but were not found at time step 2 or any other time step (Fig. 7a). Thus, the delayed learning at time step 4 observed in Fig. 6 did not involve reactivation (or persistence) of the original stimulus representations.

Could this lack of reactivated representation reflect a different kind of representation? We considered that the network is not uniformly plastic—each connection has a different baseline nonplastic weight $W_{i,j}$ and a different plasticity coefficient $A_{i,j}$. Therefore, to produce the appropriate synaptic changes through Hebbian plasticity, the relevant reinstated representations should not be identical to the original (feedforward) representation of each item $\boldsymbol{\psi}_{t1}(X)$. Rather, such plasticity would involve recoded versions of these representations, which would produce appropriate learning (that is, ensure that Hebbian learning adjusts learned representations of the relevant items in the correct direction along the decision axis) when taking into account the heterogeneous plasticity across individual synapses.

To identify these putative recoded representations, we designed an optimization-based procedure that captures the above hypotheses (Supplementary Note 8). This procedure yielded a predicted recoded version of the feedforward representations ($\boldsymbol{\psi}_{t1}(X)$), which we termed $\tilde{\boldsymbol{\psi}}_{t1}(X)$, and of the decision axis ($\mathbf{w}_{\text{out}}$), which we termed $\tilde{\mathbf{w}}_{\text{out}}$.

We found that the relevant items are indeed represented in neural activity, in the predicted recoded forms $\tilde{\boldsymbol{\psi}}_{t1}(X)$, precisely at time step 2 and with appropriate signs. Figure 7b shows the correlation between $\mathbf{r}(t)$ and the predicted recoded feedforward representations $\tilde{\boldsymbol{\psi}}_{t1}(X)$ of all items, for each step of trial 20, again using only runs in which the pair shown at trial 20 was either DE or ED. Critically, in addition to current-trial items (D and E), neighboring items (C and F) are also reinstated. Furthermore, we found that the recoded representation of the decision axis (output weight vector) is represented specifically at time step 3, with the same sign as the network's response for this trial (Extended Data Fig. 3 and data for other pairs is shown in Fig. 8). Together with the error-selective $m(t)$ signal at $t = 4$, and neuromodulated Hebbian learning as governed by equations (5) and (7), this suffices to explain the shift in learned step 2 representations of the above items in the appropriate direction, as observed in Fig. 6 (Supplementary Note 1).

Importantly, the recoded representations $\tilde{\boldsymbol{\psi}}_{t1}(X)$ are markedly different from the original representations evoked by the actual stimuli $\boldsymbol{\psi}_{t1}(X)$. Correlation between $\tilde{\boldsymbol{\psi}}_{t1}(X)$ and $\boldsymbol{\psi}_{t1}(X)$ for any item X was consistently below 0.1 in magnitude, and could be of either sign. This clarifies

https://doi.org/10.1038/s41593-024-01852-8

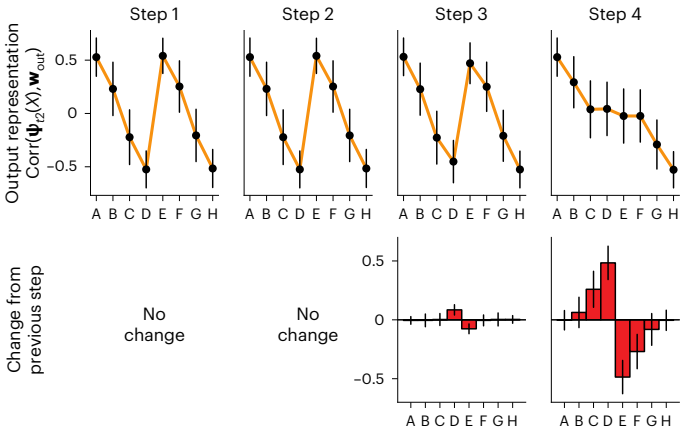


Fig. 6 | Presentation of an item pair alters representations of nonpresented neighboring items. Activity taken from network when presented with pair DE or ED for the first time at trial 20, over 1000 separately run episodes. Top row, encoding of items across time steps. Plotted is the alignment (correlation) of each item's step 2 (learned) representation $\psi_{t2}(X)$ with the output weight vector $\mathbf{w}_{out}$. Bottom row, changes in alignment from one time step to the next (that is, difference between each top-row plot over time steps). Plots show mean ± s.d. over episodes. Note that most learning occurs in step 4 (delay step); further, note that learning affects the representation of not only the pair presented in the trial (DE), but also neighboring items that were not shown in the trial (C and F). No weight changes occur in steps 1 and 2 due to reset of activations at the start of trials (Methods).

why we did not find reactivation of the original stimulus representations (Fig. 7a). Therefore, reinstated representations that support delayed learning in the network are not generally similar to the original feedforward representations, although they contain the appropriate information to enable learning for the corresponding items (Discussion and Supplementary Note 8).

Joint reinstatement of adjacent stimuli requires the model to identify which items are adjacent to one another. How is this learned? We found that the relevant learning occurs at time step 3, corresponding to the small, reward-independent spike in $m(t = 3)$ previously seen (Fig. 5). This modulatory spike induces Hebbian learning between the feedforward representations of item pairs presented at time step 1, and these pairs' reinstated, recoded representations at time step 2. The result is that future presentations of either item in the pair will induce reinstatement of the other item. Since stimuli up to trial 20 are always comprised of pairs of adjacent items, this learning mechanism explains joint reinstatement of adjacent stimuli (Fig. 7b). Importantly, the consistently negative sign of $m(t = 3)$ ensures that this joint reinstatement occurs with a common sign. See Supplementary Note 4 for details and additional experiments.

A more precise step-by-step summary of the discovered neural algorithm is provided in Supplementary Note 1. We also evaluated a non-neural, toy-model implementation of the algorithm, verifying that this overall algorithm suffices to produce appropriate representations within an episode (Supplementary Note 2).

A suboptimal solution cannot perform list linking

As mentioned earlier ('Metalearning discovers solutions for transitive inference'), the metatraining process sometimes yields networks that perform at a substantially lower level (Fig. 1f, gray traces), suggesting that these networks implement a different learning method. Indeed, analysis of these networks revealed an alternative solution to order learning, based on simple reward-modulated Hebbian learning: neuromodulatory output $m(t)$ was strongly sensitive to reward at $t = 3$ (reward delivery time; Supplementary Fig. 9). As denoted in equation (7), this signal automatically modulates Hebbian learning between feedforward item input (at step 1) and response (at step 2). Representation learning only appears to occur at time step 3 (reward delivery time; Supplementary Figs. 10 and 11). Finally, we note that after the first several trials, step 3 learning in these networks only occurred in incorrect trials (negative rewards).

This 'passive', nonreinstating Hebbian solution was capable of passable transitive inference, yet, tellingly, failed to perform list linking (Extended Data Fig. 1), unlike the 'active', cognitive solution described previously. As the passive solution only involves current-trial information, it is unable to rapidly modify representations of items not currently presented (knowledge 'reassembly'). See Supplementary Note 11 for details.

Discussion

In this study, we metatrained neural networks, endowed with synaptic plasticity and neuromodulation, in a classical transitive inference task paradigm. The metatraining process consistently produced networks that were capable of learning arbitrary orderings for transitive inference, and, further, that reproduced multiple important experimentally observed behaviors (symbolic distance effect, end-anchor effect, cross-list rank transfer, resistance to massed presentations). To our knowledge, these networks are the only neural models described thus far to do so[11].

Surprisingly, despite the simplicity of the networks, the learning mechanism employed (as discovered by the metalearning process) relied on a cognitive act—relevant previous stimuli were actively and selectively reinstated in neural activity (in recoded form), enabling trial-delayed (temporally remote) changes in their representations.

As a result of this reinstatement, and despite not being explicitly trained to do so, this solution was able to perform list linking, a task paradigm of fast knowledge reassembly[10] (observed in both monkeys[8,9,24] and humans[10]), moreover reproducing experimentally observed patterns of errors (Fig. 3 and ref. 9).

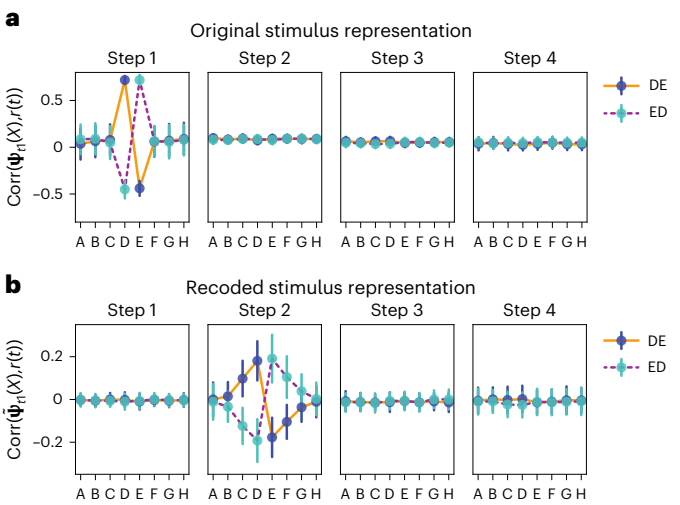


Fig. 7 | Metatrained networks reinstate item representations in a recoded form. Activity taken from network when presented with pair DE or ED for the first time at trial 20, over separately run episodes (DE, 136 episodes and ED, 142 episodes). **a**, Original stimulus representation. Alignment (correlation) between neural activity $\mathbf{r}(t)$ and stimulus-evoked representations of individual stimuli (ψ_{t1}), across items and time steps. **b**, Recoded stimulus representation. Same calculation as **a**, but using $\tilde{\psi}_{t1}$ (putative recoded representations found via optimization; Supplementary Note 8). The network expresses recoded representations of D and E at time step 2, doing so with opposite sign depending on whether the item is item 1 or item 2. Critically, neighboring items (C and F, and less strongly B and G) are also reinstated in recoded form and with appropriate signs. All plots indicate mean ± s.d. over episodes.

https://doi.org/10.1038/s41593-024-01852-8

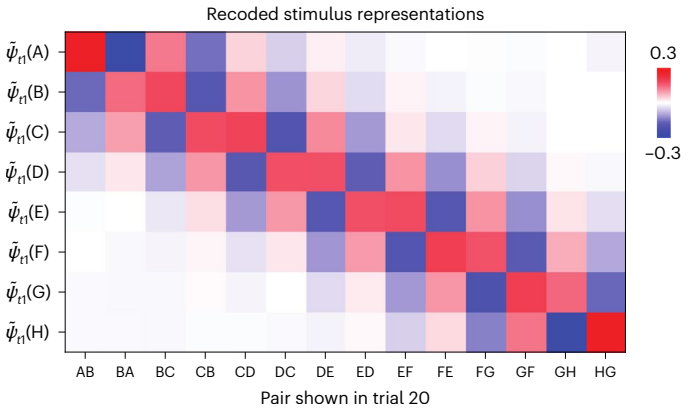


Fig. 8 | Reinstating recoded stimulus representations—additional item pairs. Alignment (correlation) between $\mathbf{r}(t)$ and $\tilde{\boldsymbol{\psi}}_{t1}(X)$ for all items at time step 2 of trial 20. Values were averaged across all 2,000 separately run episodes. Note that data for DE and ED are the same as in Fig. 7b for time step 2.

By contrast, a minority of the resulting networks learned order through simple reward-modulated Hebbian learning, applied to immediate stimuli, responses and rewards. This 'passive', nonreinstating solution was still capable of passable transitive inference at test time, suggesting that transitive inference stricto sensu (which James once called 'the broadest and deepest law of man's thought'[25] (p. 646)) does not require advanced cognition[15]. Importantly, however, this nonreinstating solution was not capable of performing list linking (Supplementary Fig. 9). This striking difference between the metatrained models echoes differences in list-linking (and order learning) ability across animal species[6,8] and human participants[10], and, further, is apropos to the inability of classical learning models to perform list linking[8,11].

The reinstatement found in the dominant solution is reminiscent of neural activity that appears to reflect reactivation of memory traces, such as hippocampal 'replay'[26–28], which has been suggested as a component of relational learning[28–32]. However, traditional replay tends to occur spontaneously and also often outside of periods of overt task engagement[26,27,30]. By contrast, reinstatement in our model is evoked by stimulus presentation at fixed time points in a trial, and is directly conjoined to representation learning within a given trial. Such stimulus-evoked, learning-period reactivation is more broadly consistent with the finding of 'retrieval-mediated learning'[33,34], which has been linked to relational learning in the associative inference task paradigm[29,35]. Our results suggest that such reactivation can also have a role in order learning paradigms, not directly for response production, but rather to modify previously established representations in accordance with relational structure (knowledge 'reassembly').

We also found that representations of previous stimuli were reinstated in a recoded form. Reactivation is often measured by testing for the re-occurrence of patterns similar to those evoked by the original stimuli[30,31], or by computing the similarity between neural activity during and after encoding[28]. Our results emphasize that reinstated representations need not be identical, or even similar, to those directly elicited by stimulus presentation, but may differ from them seemingly arbitrarily—as long as they produce the desired effect on representations (Fig. 7, compare plots, and Supplementary Note 8).

The 'active', reinstating solution uses reinstatement for learning, rather than for inference at decision time. By design, agent responses (produced at step 2, after exactly one pass through the plastic recurrent weights) cannot involve complex processing, but must operate as a model-free 'learned reflex'. By contrast, the delayed, self-generated learning at step 4 is model-based: it relies on reinstatement, and the agent must know which items neighbor each other (a simple model of its environment) to reinstate the correct ones. Therefore, the active, reinstatement-based solution combines model-based learning with model-free inference, reminiscent of the so-called Dreamer algorithm from the machine learning literature[36].

Finally, our results illustrate the potential of modeling approaches based on plastic artificial neural networks. Many recent studies have trained recurrent neural networks with backpropagation on various tasks, identifying rich dynamics that support the performance of these tasks[14,37–39]. The present study extends this approach to plastic networks, capable of controlling their own learning, and metatrained (using gradient descent) to discover new learning processes that can autonomously perform cognitive learning tasks. Our results demonstrate the power of this approach for investigating the space of neural mechanisms that support learning and cognition.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41593-024-01852-8.

References

1. Tenenbaum, J. B., Kemp, C., Griffiths, T. L. & Goodman, N. D. How to grow a mind: statistics, structure, and abstraction. *Science* **331**, 1279–1285 (2011).
2. Tervo, D. G. R., Tenenbaum, J. B. & Gershman, S. J. Toward the neural implementation of structure learning. *Curr. Opin. Neurobiol.* **37**, 99–105 (2016).
3. Lake, B. M., Ullman, T. D., Tenenbaum, J. B. & Gershman, S. J. Building machines that learn and think like people. *Behav. Brain Sci.* **40**, e253 (2017).
4. Eichenbaum, H. Memory: organization and control. *Annu. Rev. Psychol.* **68**, 19–45 (2017).
5. Behrens, T. E. J. et al. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron* **100**, 490–509 (2018).
6. Jensen, G. Serial learning. In *APA Handbook of Comparative Psychology: Perception, Learning, and Cognition* (ed. Call, J.) 385–409 (American Psychological Association, 2017).
7. Gazes, R. P., Templer, V. L. & Lazareva, O. F. Thinking about order: a review of common processing of magnitude and learned orders in animals. *Anim. Cogn.* **26**, 299–317 (2023).
8. Gazes, R. P. & Lazareva, O. F. Does cognition differ across species, and how do we know? Lessons from research in transitive inference. *J. Exp. Psychol. Anim. Learn. Cogn.* **47**, 223 (2021).
9. Treichler, F. R. & van Tilburg, D. Concurrent conditional discrimination tests of transitive inference by macaque monkeys: list linking. *J. Exp. Psychol. Anim. Behav. Process.* **22**, 105 (1996).
10. Nelli, S., Braun, L., Dumbalska, T., Saxe, A. & Summerfield, C. Neural knowledge assembly in humans and neural networks. *Neuron* **111**, 1504–1516 (2023).
11. Jensen, G., Terrace, H. S. & Ferrera, V. P. Discovering implied serial order through model-free and model-based learning. *Front. Neurosci.* **13**, 878 (2019).
12. Morton, N. W., Schlichting, M. L. & Preston, A. R. Representations of common event structure in medial temporal lobe and frontoparietal cortex support efficient inference. *Proc. Natl Acad. Sci. USA* **117**, 29338–29345 (2020).
13. De Lillo, C., Floreano, D. & Antinucci, F. Transitive choices by a simple, fully connected, backpropagation neural network: implications for the comparative study of transitive inference. *Anim. Cogn.* **4**, 61–68 (2001).
14. Kay, K. et al. Emergent neural dynamics and geometry for generalization in a transitive inference task. *PLOS Comput. Biol.* **20**, e1011954 (2024).

15. Lippl, S., Kay, K., Jensen, G., Ferrera, V. P. & Abbott, L. F. A mathematical theory of relational generalization in transitive inference. *Proc. Natl Acad. Sci. USA* **121**, e2314511121 (2024).
16. Schmidhuber, J. Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook (diploma thesis). www.bibsonomy.org/bibtex/2a96f7c3d42103ab94b13badef5d869f0/brazovayeye (1987).
17. Miconi, T. Backpropagation of Hebbian plasticity for continual learning. In *Proceedings of NIPS Workshop on Continual Learning* (2016).
18. Wang, J. X. et al. Learning to reinforcement learn. Preprint at https://doi.org/10.48550/arXiv.1611.05763 (2016).
19. Wang, J. X. et al. Prefrontal cortex as a meta-reinforcement learning system. *Nat. Neurosci.* **21**, 860 (2018).
20. Miconi, T., Rawal, A., Clune, J. & Stanley, K. O. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *Proceedings of International Conference on Learning Representations* (2019).
21. Mnih, V. et al. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger K. Q.) Vol. 48, 1928–1937 (PMLR, 2016).
22. Brunamonti, E. et al. Neuronal modulation in the prefrontal cortex in a transitive inference task: evidence of neuronal correlates of mental schema management. *J. Neurosci.* **36**, 1223–1236 (2016).
23. Lazareva, O. F. & Wasserman, E. A. Transitive inference in pigeons: measuring the associative values of stimuli B and D. *Behav. Process.* **89**, 244–255 (2012).
24. Treichler, F. R., Raghanti, M. A. & van Tilburg, D. N. Linking of serially ordered lists by macaque monkeys (*Macaca mulatta*): list position influences. *J. Exp. Psychol. Anim. Behav. Process.* **29**, 211 (2003).
25. James, W. *The Principles of Psychology* Vol. 2. (Henry Holt and Company, 1890).
26. Buzsáki, G. Hippocampal sharp wave-ripple: a cognitive biomarker for episodic memory and planning. *Hippocampus* **25**, 1073–1188 (2015).
27. Foster, D. J. Replay comes of age. *Annu. Rev. Neurosci.* **40**, 581–602 (2017).
28. Tambini, A. & Davachi, L. Awake reactivation of prior experiences consolidates memories and biases cognition. *Trends Cogn. Sci.* **23**, 876–890 (2019).
29. Shohamy, D. & Daw, N. D. Integrating memories to guide decisions. *Curr. Opin. Behav. Sci.* **5**, 85–90 (2015).
30. Barron, H. C. et al. Neuronal computation underlying inferential reasoning in humans and mice. *Cell* **183**, 228–243 (2020).
31. Comrie, A. E., Frank, L. M. & Kay, K. Imagination as a fundamental function of the hippocampus. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* **377**, 20210336 (2022).
32. Kurth-Nelson, Z. et al. Replay and compositional computation. *Neuron* **111**, 454–469 (2023).
33. Iordanova, M. D., Good, M. & Honey, R. C. Retrieval-mediated learning involving episodes requires synaptic plasticity in the hippocampus. *J. Neurosci.* **31**, 7156–7162 (2011).
34. Hall, G. Learning about associatively activated stimulus representations: implications for acquired equivalence and perceptual learning. *Anim. Learn. Behav.* **24**, 233–255 (1996).
35. Zeithamova, D., Dominick, A. L. & Preston, A. R. Hippocampal and ventral medial prefrontal activation during retrieval-mediated learning supports novel inference. *Neuron* **75**, 168–179 (2012).
36. Hafner, D., Lillicrap, T., Ba, J. & Norouzi, M. Dream to control: learning behaviors by latent imagination. Preprint at https://doi.org/10.48550/arXiv.1912.01603 (2019).
37. Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).
38. 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).
39. Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F. & Lin, H. (eds). Organizing recurrent network dynamics by task-computation to enable continual learning. In *Proceedings of Advances in Neural Information Processing Systems* Vol. 33, 14387–14397 (Curran Associates, 2020).

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2025

Methods

Terminology

Metalearning, or 'learning-to-learn', consists of training an agent so that it can solve new instances of a general learning problem[16,18,19,40]. That is, instead of training the agent separately on each new instance of the problem, we metatrain an autonomous learning agent so that it acquires the ability to learn autonomously and efficiently any given instance of the problem, including new instances never seen during training. Classic work by Harlow[41] first showed that animals possess this ability in an item-value association metatask—over the course of multiple episodes, monkeys became progressively better at learning (and exploiting) which of two possible items was associated with a food reward. Typical metalearning problems in the literature include maze solving[42,43], bandit tasks[18,42], fast association between stimuli and labels[18,43,44] and item-value problems such as the Harlow metatask[19,41]. Here we apply a metalearning framework to plastic neural networks, training them to be able to learn new ordered series of arbitrary stimuli from pairwise presentations, as in classic transitive inference experiments[8,11,45].

In the following, to avoid confusion, we will maintain separate usage for the words learning and training. We use the word 'learning' to denote within-episode learning of one particular ordered list, driven by synaptic plasticity in plastic weights. In contrast, we use the word 'training' to denote the gradient-based optimization that occurs between episodes, and modifies the structural parameters of the network, with the goal to improve within-episode plasticity-driven learning. This distinction follows that between 'inner loop' and 'outer loop', respectively, in standard metalearning terminology[18,42].

In each trial, after the network produces a response, it is given a binary feedback signal $R(t)$ that indicates whether this response was correct or not. We call this signal 'reward', although we stress that it is merely an additional input to the network, and has no predefined effect on network structure by itself.

Finally, we use 'stimulus' and 'item' interchangeably.

Network organization

The network is a recurrent neural network endowed with Hebbian plasticity and self-controlled neuromodulation, containing $n = 200$ neurons (Fig. 1). The general structure of the network and learning process largely follows those of a previous study[20]. The code was written entirely in PyTorch and run on Google Colab. All code for the following experiments is available online (Code availability).

The inputs $\mathbf{i}(t)$ consist of a vector that concatenates the stimuli for the current time step, the reward signal $R(t)$ for the current time step (0, 1 or −1), and the agent's response at the previous time step (if any), in accordance with common metalearning practice[18,19,42]. The network's output is a probability distribution over the two possible responses ('choose stimulus 1' or 'choose stimulus 2'). At response time, the agent's actual response for this trial is sampled from this output distribution; at other times, the outputs are ignored.

Recurrent connections within the network undergo synaptic changes over the course of an episode, modeled as simple neuromodulated Hebbian plasticity as described in the 'Synaptic plasticity' section.

Formally, the fully connected recurrent network acts according to the following equations:

$$\mathbf{x}(t) = W_{\text{in}}\,\mathbf{i}(t) + (W_{\text{rec}} + A \odot P(t))\,\mathbf{r}(t-1) \tag{1}$$

$$\mathbf{r}(t) = \tanh(\mathbf{x}(t)) \tag{2}$$

$$o(t) = \text{Softmax}(W_{\text{out}}\,\mathbf{r}(t)) \tag{3}$$

$$m(t) = W_{\text{mod}}\,\mathbf{r}(t) \tag{4}$$

Here $\mathbf{i}(t)$ is the vector of inputs provided to the network, $\mathbf{x}(t)$ is the vector of neural activations (the linear product of inputs by weights), $\mathbf{r}(t)$ is the neural firing rates (activations passed through a nonlinearity), $o(t)$ is the output of the network (that is, the probability distribution over the two possible responses) and $m(t)$ is the (scalar-valued) neuromodulatory output whose role is to gate synaptic changes (plasticity; see below).

W_{rec} and A are the base weights and plasticity parameters (plasticity learning rates) of the recurrent connections, respectively. They are structural parameters of the network, and do not change during an episode, but rather are slowly optimized between episodes by gradient descent. By contrast, $P(t)$ is the plastic component of the weights, reset to 0 at the start of each episode and changing over an episode according to the plasticity rule described below. The symbol $\odot$ denotes the pointwise (Hadamard) product of two matrices.

We note that these equations are simply the standard recurrent neural network equations, except that the total recurrent weights are the sum of base weights W_{rec} and the plastic weight term $A \odot P(t)$.

Crucially, within an episode, only the plastic recurrent weights $P(t)$ are updated, with the updating governed by the plasticity rule described below. All other parameters (W_{in}, W_{rec}, W_{out}, W_{mod} and A) are fixed and unchanging within an episode, but are optimized by gradient descent between episodes.

In all experiments, the final trained output weight matrix W_{out} were found to have two highly anticorrelated rows ($r < 0.9$), corresponding to the two opposite possible decisions (choose item 1 vs. choose item 2). Thus, in the text, we generally summarize the W_{out} matrix by a single vector $\mathbf{w}_{\text{out}}$, computed as the difference between the two rows of W_{out}. All results are unchanged (up to sign) if $\mathbf{w}_{\text{out}}$ was instead taken to be either row of W_{out}.

Synaptic plasticity

To model synaptic learning, recurrent connections are endowed with modulable Hebbian plasticity[20]. Each connection maintains a so-called Hebbian eligibility trace $H(t)$, which is a decaying running average of the product of outputs and inputs.

$$H(t+1) = \eta\,\mathbf{x}(t)\,\mathbf{r}(t-1)^{\mathrm{T}} + (1-\eta)H(t) \tag{5}$$

Since the network is recurrent, $\mathbf{x}(t)$ corresponds to the 'outputs', while $\mathbf{r}(t-1)$ corresponds to the 'inputs' across the recurrent connections (cf. equation (1)).

Finally, the network continually produces a neuromodulation signal $m(t)$ that gates the Hebbian trace into actual changes in plastic weights P.

$$P(t+1) = P(t) + m(t)\,H(t) \tag{6}$$

We emphasize that $P(t)$ is initialized to 0 at the beginning of each episode, and changes according to the above equations (without reinitialization) over the course of an episode. Also note that while $H(t)$ is decaying, $P(t)$ does not decay within an episode.

In equation 5, η is a network-wide parameter, optimized by gradient descent between episodes. Notably, across many experiments, we observed that training consistently settles on a value of $\eta \approx 1$. This means that $H(t)$ is almost fully updated at each time step, with essentially no memory of events occurring before time $t-2$. As such, we can approximate the actual plasticity process as follows:

$$P(t+1) \approx P(t) + m(t)\,\mathbf{x}(t-1)\,\mathbf{r}(t-2)^{\mathrm{T}} \tag{7}$$

This observation has an important role in the elucidation of network behavior, as described in the main text.

Task settings

Our objective is not simply to train successful learning networks but also to fully understand the neural mechanism of learning in these

trained networks. To this end, we deliberately sought to simplify trial structure as much as possible while retaining the essential structure of real-world experiments.

First, we restricted each trial to the shortest possible duration that still allowed for successful training, which we found to be exactly four time steps—stimulus presentation at step 1, network response at step 2, external feedback (reward) at step 3 and, lastly, a delay time step before the start of the next trial, as step 4.

Second, we reset neural activations $\mathbf{x}(t)$, $\mathbf{r}(t)$ and Hebbian eligibility traces $H(t)$ at the start of each trial (but not plastic weights $P(t)$). This is to ensure that neural activity dynamics remain confined to a single trial, with only plastic weights carrying memory of past events from one trial to the next. If we do not reset neural activations between trials, trained networks consistently settle on strategies that process information from successive trials in parallel; that is, activations in trial k represent information not just from trial k but also from trial $k-1$. While potentially interesting, this phenomenon complicates the investigation of mechanisms underlying within-episode learning. We therefore chose to reset activations at the start of each trial.

Rewards in the last ten episodes are upweighted by a factor of 4; this only modifies the metalearning loss for outer-loop gradient descent and does not affect the reward signal registered by the network.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Code availability

All code for the experiments described above, as well as saved parameter files for representative networks for both discovered solutions, is available at https://github.com/ThomasMiconi/TransitiveInference.

References

40. Thrun, S. & Pratt, L. (eds). *Learning to Learn: Introduction and Overview* 3–17 (Kluwer Academic Publishers, 1998).
41. Harlow, H. F. The formation of learning sets. *Psychol. Rev.* **56**, 51–65 (1949).
42. Duan, Y. et al. RL^2: fast reinforcement learning via slow reinforcement learning. Preprint at https://doi.org/10.48550/arXiv.1611.02779 (2016).
43. Miconi, T., Clune, J. & Stanley, K. O. Differentiable plasticity: training plastic networks with gradient descent. In *Proc. 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) Vol. 80, 3559–3568 (PMLR, 2018).
44. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D. & Lillicrap, T. Meta-learning with memory-augmented neural networks. In *Proc. 33rd International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger, K. Q.) Vol. 48, 1842–1850 (PMLR, 2016).
45. Bryant, P. E. & Trabasso, T. Transitive inferences and memory in young children. *Nature* **232**, 456–458 (1971).

Acknowledgements

We thank L. Abbott, V. P. Ferrera, S. Lippl, G. Jensen and X. X. Wei for fruitful discussions about this work and comments on an earlier version of the paper. This work was supported by the Simons Collaboration for the Global Brain (521921 and 542981), the NSF NeuroNex award (DBI-1707398), the Gatsby Charitable Foundation and NIMH K99 (MH126158-01A1 to K.K.).

Author contributions

T.M. designed and ran the code. T.M. and K.K. conceived the experiment and wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s41593-024-01852-8.

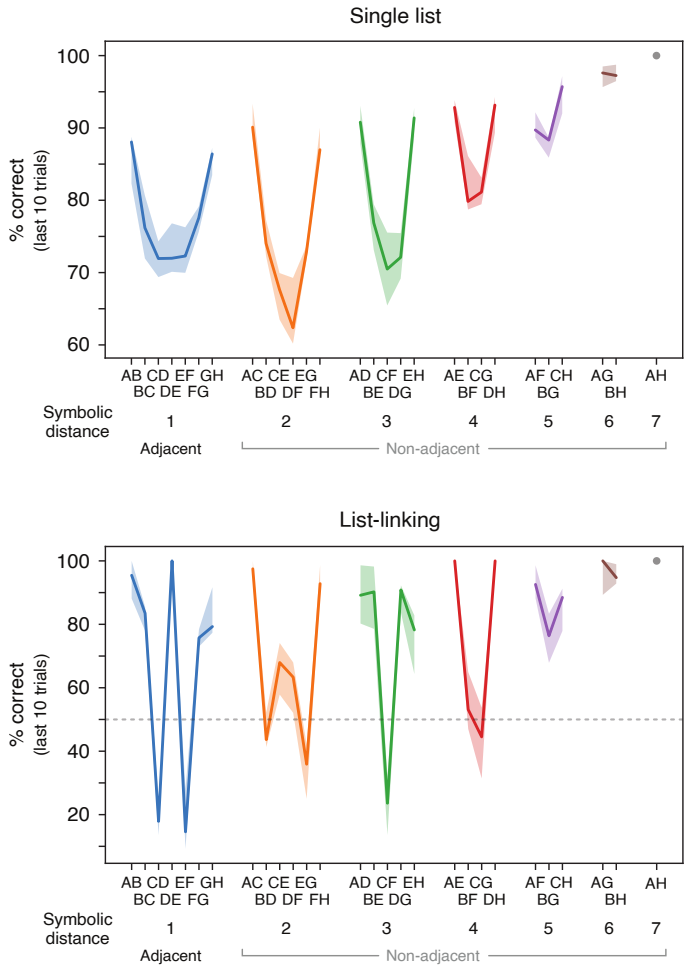
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41593-024-01852-8.

Correspondence and requests for materials should be addressed to Thomas Miconi or Kenneth Kay.

Peer review information *Nature Neuroscience* thanks the anonymous reviewers for their contribution to the peer review of this work.

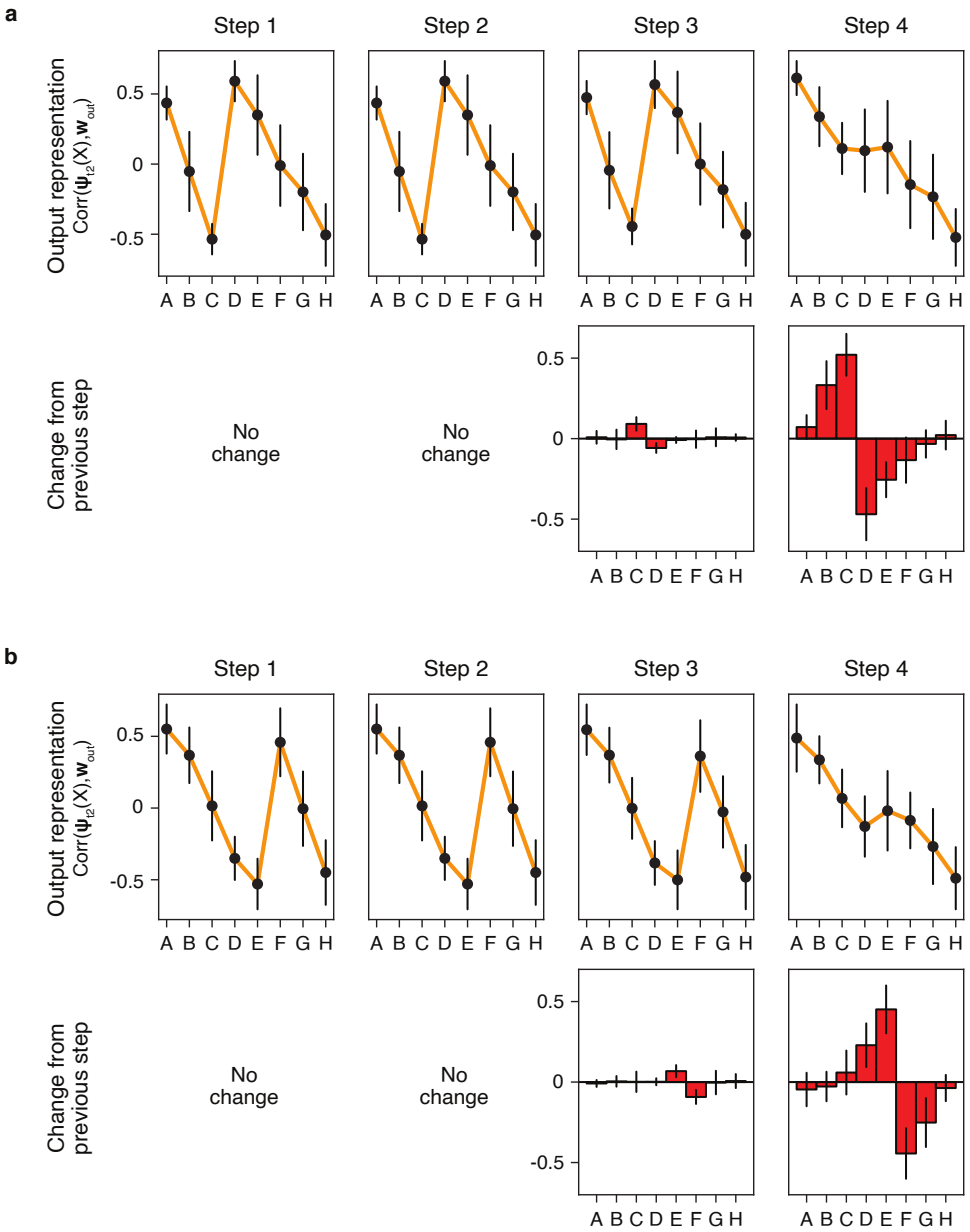
Reprints and permissions information is available at www.nature.com/reprints.

https://doi.org/10.1038/s41593-024-01852-8



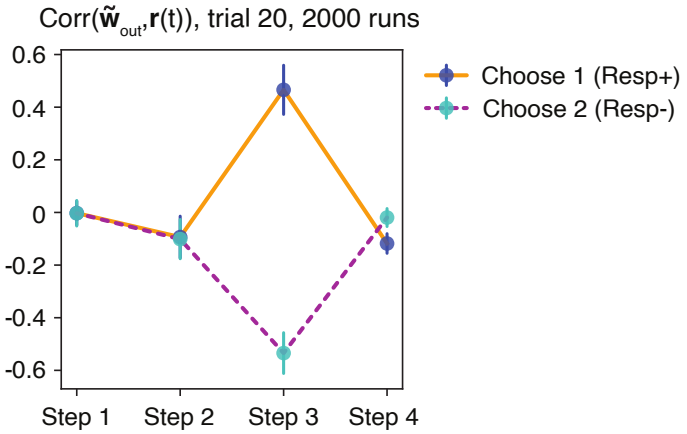
Extended Data Fig. 1 | Performance of the suboptimal network. Proportion of correct test trials, in standard (single-list) conditions (top; compare to high-performance network in Fig. 2a) and list-linking conditions (bottom; compare to high-performance network in Fig. 3a), over 2,000 episodes. Median and interquartile range of correct responses for each item pair over 10 sets of 200 episodes each. Note that the suboptimal network performs transitive inference and expresses the symbolic distance effect for a single list, but fails the list-linking task. Conventions as in Figs. 2 and 3.

https://doi.org/10.1038/s41593-024-01852-8



Extended Data Fig. 2 | Step-by-step representation changes for different withheld pairs. Conventions are as in Fig. 6 (showing mean ± s.d. over 1000 separately run episodes), but here the pair that is withheld until trial 20 is either CD (**a**) or EF (**b**) rather than DE. Representation learning proceeds similarly for these other pairs.

https://doi.org/10.1038/s41593-024-01852-8



Extended Data Fig. 3 | Reinstatement of recoded decision axis. Correlation between neural activity r(t) and recoded vector of output weights $\tilde{\mathbf{w}}_{out}$, at each time step of trial 20 (mean ± s.d. over 2000 separately run episodes), shown separately for runs in which network response at trial 20 was positive vs. negative. While actual response occurs at step 2, the recoded representation of the combined output weight vector $\tilde{\mathbf{w}}_{out}$ is strongly represented at step 3, with sign correlated with network response for this trial. This representation enables Hebbian association with recoded representations of stimuli reinstated at step 2 (Fig. 6).

nature neuroscience

Supplementary information

https://doi.org/10.1038/s41593-024-01852-8

Neural mechanisms of relational learning and fast knowledge reassembly in plastic neural networks

In the format provided by the authors and unedited

Supplementary Materials

S1 Summary of the discovered neural algorithm

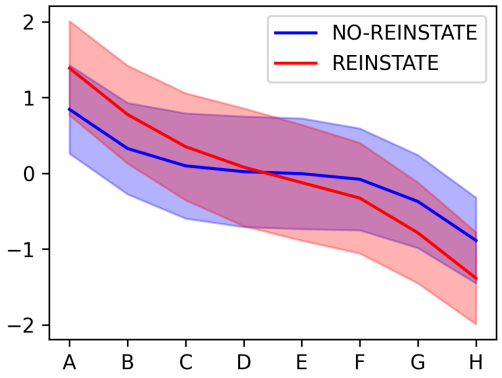
Here we provide a step-by-step summary of the discovered neural algorithm (higher-performance, list-linking-capable version).

1. *Step 1: feedforward stimulus representations.* At step 1, the network perceives the two items for the current trial. Neural activations $\mathbf{r}(t = 1)$ reflect the feedforward representations of both items, as determined by the fixed, non-plastic input weights $\mathbf{W}_{in}$. Input weights for the two items turn out to be strongly negatively correlated with each other, and thus the resulting representation of the pair is essentially a *subtraction* between the feedfoward representations of both items in isolation.

2. *Step 2: Trial decision and item reinstatement (multiplexed).* At step 2, recurrent (plastic) weights project the previous feedforward representations into *learned* representations (step-2 representations are learned because they are determined by plastic recurrent weights, which are modified by within-episode learning). These learned representations turn out to encode rank geometrically: stimulus rank is encoded by the alignment of the stimulus' step-2 representation with a fixed "decision axis", which turns out to be strongly correlated with the output weight vector $\mathbf{w}_{out}$. Because step-1 feedforward representations of pairs are subtractions between the representations of each item, the step-2 representation of the pair is also approximately the subtraction between the learned, step-2 representations of both items. The alignment of this subtraction with the decision axis (output weight vector) thus automatically encodes the rank difference between the items, and determines the network's response for this trial, thereby explaining transitive inference and the symbolic distance effect, as explained above. *In parallel to this*, the network also reinstates in its activity the (recoded) step-1 representations of both shown items, as well as other items previously "*coupled*" with these (see below).

3. *Step 3: Reward delivery and decision axis reinstatement.* At step 3, a feedback signal informs the network about response correctness. At this step, a moderate burst in $m(t)$, of constant sign and independent of response or reward, induces Hebbian learning between $\mathbf{r}(t = 1)$ and $\mathbf{r}(t = 2)$. This plasticity "stamps in" the coupling between step-1 feedforward representations and step-2 reinstatements of their recoded versions discussed above. This ensures their "coupling" and thus future joint reinstatements at step 2 of future trials. *In parallel*, the network reinstates in its activations the (recoded) representation of the decision axis, with the appropriate sign depending on the network's response for the current trial.

4. *Step 4: Representation learning.* At step 4, a larger, reward-sensitive burst of $m(t)$ induces Hebbian learning (with the appropriate sign depending



Supplementary Figure 1: Representation learning in the toy model. Final values for each item after 20 trials, mean and standard deviation over 1000 separate episodes, shown separately with or without reinstatement. Compare with Figures Figure 4 and Supplementary Figure 10.

on response correctness) between $\mathbf{r}(t = 2)$ and $\mathbf{r}(t = 3)$. This links the reinstated, recoded step-1 representations of the shown items *as well as* their coupled neighbours (which were all reinstated at step 2, as mentioned above) to the reinstated, recoded representation of the decision axis (which was reinstated at step 3, as mentioned above). The resulting modulated Hebbian learning modifies plastic weights in such a way that the step-2 representations of all reinstated items (both currently shown items and not-shown neighbouring items coupled with these) will be shifted in the appropriate direction along the decision axis, ensuring that this alignment encodes stimulus rank.

S2 Toy model implementation

We implement a non-neural, toy-model version of the discovered list learning algorithm as we understand it, in order to illustrate its ability to learn adequate representations within 20 trials. Each item is initially assigned a random value between -1 and 1 (representing its alignment with the output vector). We iterate the following process 20 times: select a pair of adjacent items, produce a response equal to the difference of their values (whose sign should therefore indicate the current putative order between these two items), and if this response is wrong (i.e. the current estimated values of the pair are in the wrong order), add a certain increment to the value of the first item and subtract the same increment from the value of the second item. The increment is equal to 1 if the response was positive or -1 if the response was negative. The increment represents the plasticity between representations of items (with opposite signs)

and the decision taken at response time (i.e. the activation along the output weight vector), which increases or decrease the similarity between the evoked (learned) representations of the items and the output weight vector, and thus the magnitude and sign of the response. Note that the modification is only applied when the response was incorrect.

Furthermore, we also apply the increment to the items just before and after the current pair, with the proper sign, but halved; this emulates the extension of learning to reinstated, coupled items in addition to the pair shown in the current trial, as demonstrated in Figure 7.

The actual code for this toy model is as follows (full code available at `http://github.com/ThomasMiconi/TransitiveInference`):

```
ALPHA=1.0
REINSTATE=True
for numepisode in range(NBEPISODES):
    vals = 2.0 * np.random.rand(8) - 1.0  # Initialize the values
    for numtrial in range(20):
      # Pick a pair of items
      pos1 = np.random.randint(7)
      pos2 = pos1 + 1
      response = vals[pos2] - vals[pos1]
      correct = True if response < 0 else False # vals1 should be > vals2
      incr =  ALPHA * (1 if response > 0 else -1) * (-1 if correct else 1)
      if correct and ONLYUPDATEONERROR:
          incr = 0
      vals[pos1] += incr
      vals[pos2] -= incr
      if REINSTATE:
          if pos1>0:
            vals[pos1-1] += .5 * incr
          if pos2<7:
            vals[pos2+1] -= .5 * incr
```

We run this toy model for 1000 episodes of 20 trials each. Supplementary Figure 1 shows the mean and standard deviation of the final values for each item at the end of each episode, separately with or without reinstatement (that is, with or without the application of half the increment to the neighbouring items in addition to the current pair).

To quantify the linearity of these representations, at the end of each episode of 20 trials, we compute the correlation between the final values of each item and the item's rank (from 1 to 8). Across the 1000 episodes, we get a mean correlation of -0.77 (standard deviation 0.17) with reinstatement, and -0.57 (standard deviation 0.22) without reinstatement. The difference is quantitatively moderate, as expected, but highly significant (Mann-Whitney U test between the two groups of 1000 correlations, with and without reinstatement respectively: $p < 10^{-90}$).

If we update on both correct and incorrect trials (switching the sign of the increment on correct trials to represent the changed sign of rewards), we get

somewhat lower linearity, with mean correlations of -0.63 (standard deviation 0.13) and -0.43 (standard deviation 0.12) with and without reinstatement, respectively.

S3 Learning multiple lists

Though not strictly relevant to the purpose of the present paper, we ensure that the networks can learn multiple successive orderings rather than just one. This is done with two interventions. First, we only reset plastic weights $\mathbf{P}(t)$ every third episode. Second, the "test" period of each episode (the last 10 trials) include 25% of pairs involving stimuli from the previous ordering since the last $\mathbf{P}(t)$ reset (if any). While not immediately relevant to the present work, this constitutes a simple example of meta-continual learning, where the goal is not just to be able to learn new problems efficiently, but also to keep in memory previously seen problems, at least up to a short horizon. Performance on pairs from previous lists was consistently high (above 70%), confirming that the model is able to learn and store several different lists.

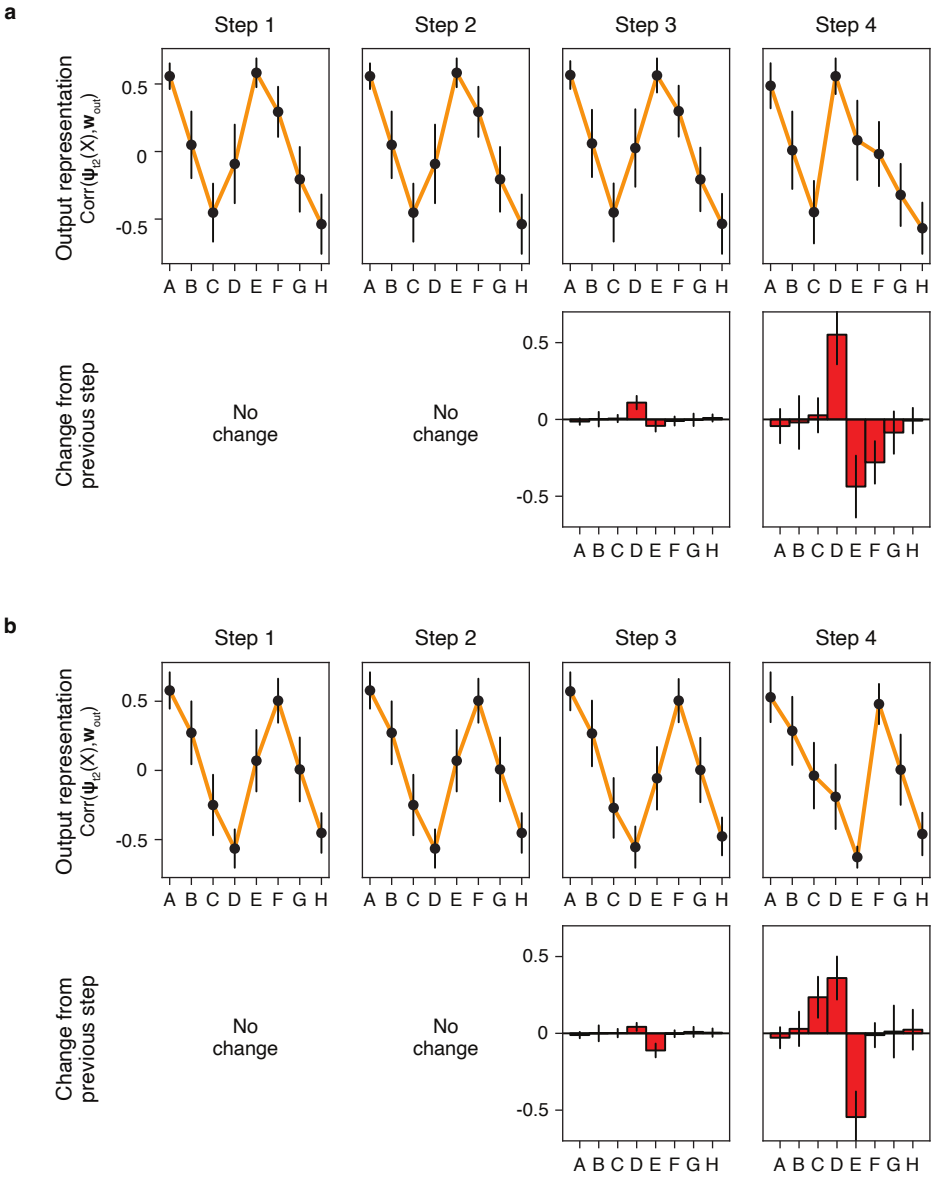
The above procedure was followed for all training. At test time, among the results reported in this paper, the list-linking examples are implemented as three successive episodes of different lengths (plus a single test trial), without initialization of $\mathbf{P}(t)$ between these episodes. All other reported results used a single test episode with $\mathbf{P}(t)$ initialized to 0 at the start of the episode.

S4 Coupling between adjacent items for future joint reinstatement

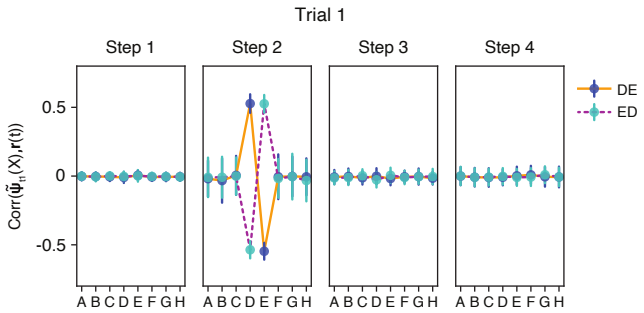
As explained in the main text, a critical component of the discovered "active" solution is that the network reinstates not just the current-trial items, but also other items — most importantly, the items adjacent to the current-trial items. This enables representation changes to occur for these other items, rather than just items presented in the trial. In turn, this learning process explains the network's ability to perform list-linking, i.e. rapid re-arrangement of existing knowledge in response to limited new information.

However, for reinstatement to include these other items, the network must know which items are adjacent to one another. How is this learned?

To address this question, we first consider runs in which the model sees pair DE or ED at trial 20 for the first time; but now, out of those runs, we select only the runs in which, in addition, the model also did not see pair CD or DC before. Fig. Supplementary Figure 2 shows that, for these runs, no transfer to item C occurs: at step 4 of trial 20 (when the model first observes pair DE), items D, E, and F see some change in their alignment with the decision axis, but item C does not. Similarly, not having seen pair EF prevents any transfer of information from pair DE to item F.



Supplementary Figure 2: Representation changes does not extend from current-trial items to adjacent items not shown in the current trial if they were not previously paired with current-trial items. Conventions as in Figure 6, showing mean and standard deviations over 1000 separately run episodes. Step-by-step changes in alignment when showing pair DE (in any order) at trial 20, after withholding both pair DE as well as a neighbouring pair until trial 20. **a**: additional withheld pair is CD. **b**: additional withheld pair is EF. At time step 4, and unlike Figure 6, no transfer of learning to the withheld neighbouring item occurs (flat bars for C in **a**, F in **b**). This confirms that transfer of information from currently shown items (here D and E) to neighbouring items only occurs if these other items have been shown in pair with the currently shown items before.



Supplementary Figure 3: Reinstatement of current-trial items (in recoded form) does not require within-episode learning, but reinstatement of neighboring items does. Same conventions as Figure 7, but for trial 1 (before any within-episode learning can take place), selecting only episodes when item pair DE (153 episodes) or ED (147 episodes) was shown in trial 1 (mean and standard deviation). $\tilde{\boldsymbol{\psi}}_{t1}(X)$ is represented in step-2 neural activity, but only for the two items D and E. Neighbouring items C and F are not reinstated, unlike Figure 7.

These observations indicate that extension of representation changes to adjacent items, not shown in the current trial, only occur if these additional item were previously shown as a stimulus pair together with one of the current-trial items. Thus, on each presentation of a pair of stimuli, the network not only adjusts their learned representations, but also learns to *couple* them, so that future trials including either of these items will transfer information to the other (even if this other item is not present).

How does the network mechanistically implement this coupling of adjacent items for future reinstatement? We observe that even at trial 1, when no learning yet has taken place in the plastic weights, step-2 neural activities do reinstate the recoded representations $\tilde{\boldsymbol{\psi}}_{t1}$ of the two current-trial items, each with the expected sign (Supplementary Figure 3). However, only these items are represented: unlike trial 20 (Figure 7), adjacent items on either side (C and F) are not reinstated.

This shows that, for the items shown in a given trial (but not for adjacent not-currently-shown items), step-2 reinstatement is not a learned outcome, but an "innate" process, meta-learned by gradient descent, and immediately active before any within-episode learning takes place.

Based on this, Hebbian learning occurring at step 3 would thus associate the initial feedforward (step-1) representations of each item with the reinstated, recoded representation of the *other* item (at step 2). As a result, future presentations of either of these items would automatically evoke step-2 reinstatement of the other, now-coupled item, even if it is not present - as observed in Figures Figure 7 and Figure 8. This would suffice to produce the observed joint reinstatement after enough learning trials occur.

Such Hebbian learning at time step 3 is in fact precisely what we observe in the neuromodulatory output $m(t)$, as denoted by the constant-sign, reward independent spike at $t = 3$ observed in Figure 5. Furthermore, this explains the constant, negative sign of this spike noted in Section 3.2: because current-trial stimuli are shown with opposite signs, but must be reinstated with identical sign (because they will be on the same "side" of whatever pair is shown in the future), Hebbian learning must be negatively modulated to ensure correct association.

This also explains the fact that $m(t)$ at time step 3 (reward time) is largely unaffected by the reward, which might otherwise seem counter-intuitive: step-3 $m(t)$ is not involved in order learning, but performs pair coupling between the two items for future joint reinstatement, which does not depend on reward for this trial.

S5 Meta-Reinforcement Learning algorithm: The A2C algorithm

The outer loop of our method ascends the gradient of episodic reward over the network's structural parameters (the various $\mathbf{W}$ matrices and $\mathbf{A}$, plus additional scalar parameters; see below). Gradients are estimated by A2C (Advantage Actor-Critic), a well known reinforcement learning (RL) algorithm that is commonly used in meta-RL experiments [18, 43, 19].

First, in addition to previously discussed outputs, the network also produces a scalar output $\hat{v}$ (the "value-prediction" output), through the weight vector $\mathbf{w}_{\hat{v}}$. This output tries to predict, at each point in time, the current value of $V(t) = \sum_{t'=t}^{T_{max}} \gamma^{t'-t} R(t')$, that is, the sum of future discounted rewards during the episode.

After the episode concludes, we retroactively go back through the history of rewards obtained at each time step, allowing us to compute the real, ground-truth $V(t)$ for each time step. Then the actual objective J to be maximized is computed as follows:

$$J = \sum_{t=0}^{T_{max}} \beta_1(t)\, \pi_t(a_t)\, \text{StopGradient}(V(t) - \hat{v}(t)) + \beta_2 H(\pi_t) - \beta_3 (V(t) - \hat{v}(t))^2 \quad (8)$$

Here, a_t is the action (i.e. the response) given at time t, π_t is the probability distribution over both possible actions/responses (determined at each time step by a SoftMax over both output neurons of the network, see Methods). H denotes a measure of dispersion of a probability distribution, usually implemented as the Shannon entropy; in practice, due to computational limitations, we set the H term as the sum of squares (an alternative measure of dispersion) rather than the actual Shannon entropy. $\beta_1(t)$ is a multiplier set to 1 for the first 20 trials (on adjacent pairs only) and 4 for the last 10 trials (on all pairs; see Methods), and β_2, β_3 are positive constants. Note that the second term encourages the

output distribution over possible actions to be as uniform as possible (with the goal of favouring exploration), while the third term optimizes the value predictor head $\hat{v}(t)$.

Intuitively, this algorithm can be thought of as a version of the standard REINFORCE method for estimating the gradient of rewards over weights that control stochastic responses: take the weight modification that would make the response you just gave more likely in the future for the same inputs, and multiply it by the actual rewards obtained. Furthermore, in A2C (as in other actor-critic methods), the actual rewards are centered by subtracting a network-generated estimate of future discounted rewards $\hat{v}(t)$. This centering has the effect of reducing the variance in the gradient estimate. In addition to this, training the value predictor also plays the role of an "auxiliary loss": a signal with a hard ground truth that provides information about environment structure. Finally, A2C also includes an additional entropy term $H(t)$ to encourage maximally uniform distribution over possible outputs, and thus (hopefully) exploration.

The parameters being subject to gradient descent between episodes are $\mathbf{W}$ matrices, $\mathbf{A}, \eta$, and a scalar multiplier on $m(t)$. Gradients of J over these parameters are computed automatically by PyTorch, using the Adam optimizer. Note that gradient computation involves backpropagation of gradients through time, since activities at time t influence activities at future times. One step of gradient descent is applied after each episode, backpropagating through time over all 30 trials of the episode. We reiterate that no gradient descent occurs during a given episode.

S6 Freezing input weights does not prevent successful meta-learning of transitive inference

As described in the main text, the discovered neural algorithm essentially performs a subtraction between the representations of the two items in a presented pair. This is done automatically due to the fact that the meta-trained input weights for both items (i.e. the two halves of $\mathbf{W}_{in}$) are strongly anti-correlated, that is, of nearly identical value and opposite signs.

While this meta-trained representation is convenient for transitive inference, here we show that it is not a necessary condition for success. We follow the exact same process as in the main text, but with *frozen* input weights. More precisely, we take the randomly initialized $\mathbf{W}_{in}$, zero out two thirds of the matrix (such that only one third of RNN neurons receive any sensory input, while the rest of RNN neurons receive no sensory input), and multiply it by 10 to adequately scale inputs. After this, meta-training proceeds as described above, except that the input weights $\mathbf{W}_{in}$ are frozen and undergo no change at all during the entire process.

Performance of the final meta-trained networks is essentially unchanged (Supplementary Figure 4). Crucially, the resulting networks are still fully capable of list-linking. This suggests that meta-training is able to sculpt connections

in such a way that the relevant subtraction is performed within the RNN itself (by mutual inhibition), instead of directly at the level of inputs.

S7 Individual item rank is not directly encoded in neural representation of pairs

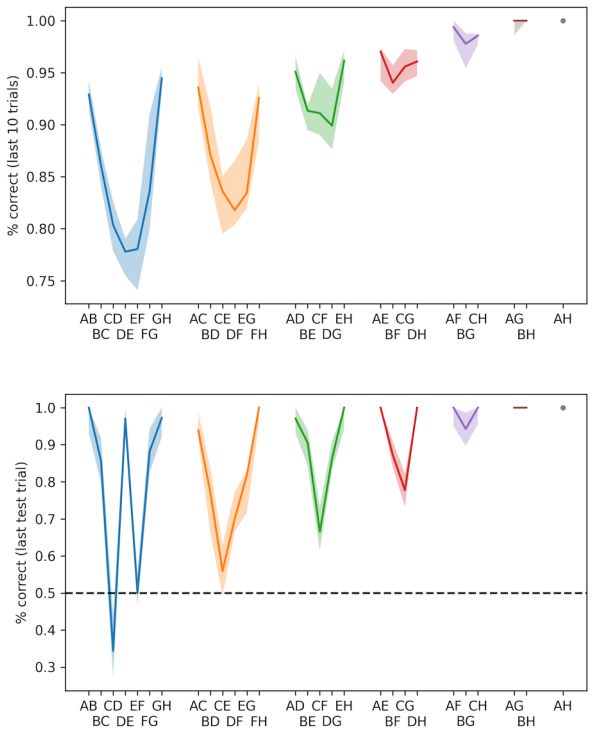
We run a full episode with a trained network over 2000 runs (each with different randomly generated stimuli), and use a linear decoder to decode various quantities from $\mathbf{r}(t = 2)$ (step 2 neural activations), separately for each trial. We train our decoder on data from the first 1800 runs, and test it on data from the last 200 runs (similar results hold with simple least-square regression, logistic regression and ridge regression, as well as a simple multi-layer perceptron with one hidden layer of 100 neurons). To estimate the model's prediction performance, we measure correlation between predicted and actual class over the 200 test runs (note that correlation remains informative about model performance even for highly imbalanced classes).

Results are shown in Supplementary Figure 5. The quantities we try to decode are:

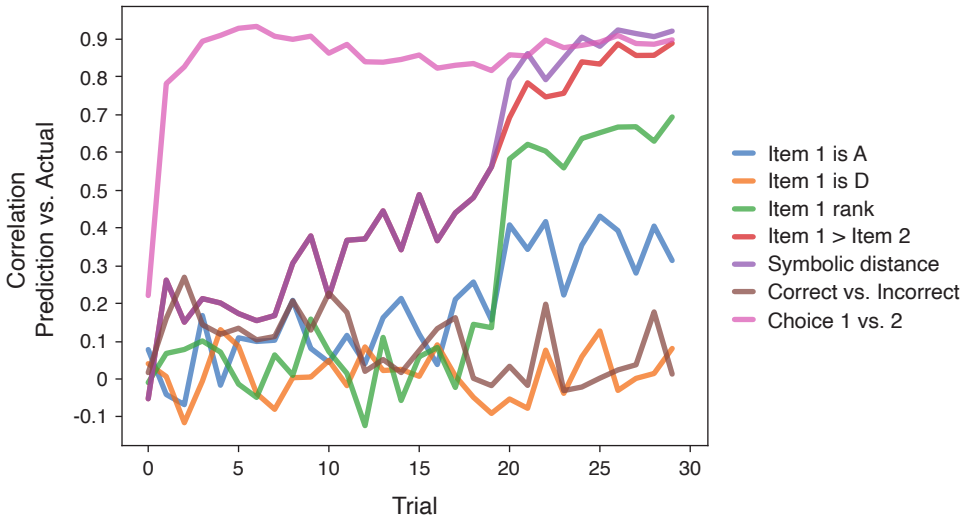
- *Cue1_is_A*: whether item 1 in this trial is A.
- *Cue1_is_D*: whether item 1 in this trial is D
- *Rank_of_Cue1*: the actual numerical rank of item 1
- *Ordered?* whether the items were shown in descending order (Stim1 ¿ Stim2) in this trial
- *Distance*: the distance in rank between the items (which is always 1 for the first 20 trials)
- *Resp_was_corr*: whether the network's response was correct for this trial
- *Resp+/-*: the network's response for this trial

Information about the network's response is clearly encoded in $\mathbf{r}(t = 2)$ from early on in the episode (pink curve). Information about the ordering of the two items (i.e. the actual correct response for this trial) and their distance slowly increases over time (purple and red lines). Note that before trial 20, all pairs shown are adjacent pairs, and thus item ordering and item distance (always -1 or +1) carry the same information, explaining their identical curves up to this point.

By contrast, information about the identity of each individual item is not encoded at all before trial 20. Even after trial 20, the apparent ability to decode item 1 rank is very similar to that of a classifier only given the correct response for this trial as input; that is, $\mathbf{r}(t)$ does not seem to possess additional information about individual item rank above and beyond that provided by knowledge of the correct response for this trial. Thus, the network does not seem to reliably encode individual item identity in its responses.



Supplementary Figure 4: Networks trained with frozen input weights still demonstrate transitive inference and list-linking, showing that anti-correlated input weights emerging from training are not a necessary requirement for success. Test performance with frozen, non-trainable input weights. Top: normal test trials. Bottom: List-linking (single test trial after training separately on $A-D$, $E-F$, and finally $D-E$). Conventions are as in Figures Figure 2 and Figure 3. Again, data from 2000 episodes is split into 10 subsets of 200 episodes each (simulating sessions), mean success rate is computed for each subset and each pair; solid line and shaded areas indicate median and inter-quartile range over the ten subsets for each pair.



Supplementary Figure 5: Decoding performance, at each trial in the episode, for various quantities. See text for details.

S8 Identifying recoded representations with an optimization procedure

Our analysis indicated that representation learning occurred at time step 4, and involved a shift in the step-2 representation of relevant items along the axis of the output weight vector (with either sign) (Figure 6). These step-2 representations, in turn, are the result of passing step-1 feedforward representation through one step of recurrence with the learned plastic weights $\mathbf{P}(t)$. Thus, if the step-2 representations change, it implies that $\mathbf{P}(t)$ also changed. How did this change happen?

Recall that plastic weights are shaped by Hebbian learning, which essentially amounts to $\mathbf{\Delta P}(t) \approx m(t)\,\mathbf{x}(t-1)\,\mathbf{r}(t-2)^{\mathsf{T}}$ (see Equation 7). For intuitive understanding, let us overlook the nonlinearity and assume a positive $m(t=4)$. If weight changes were purely Hebbian, including the step-1 representations in $\mathbf{r}(t=2)$ and the output weight vector in $\mathbf{r}(t=3)$ would increase plastic weights at time $t=4$ by $\mathbf{w}_{out}\boldsymbol{\psi}_{t1}^{\mathsf{T}}$, which would have exactly the appropriate effect for representation learning: future presentations of $\boldsymbol{\psi}_{t1}^{\mathsf{T}}$ to the recurrent network (at $t=1$)would produce an output ($\mathbf{r}(t=2)$) more similar to $\mathbf{w}_{out}$ (vs. less similar for a negative $m(t)$). This would constitute a straightforward example of Hebbian associative learning.

However, as seen in Figure 7, this does not occur. Neural activities at time steps 2 and 3 do not seem to contain step-1 representations of the items, or of the output weight vector. This leads to the following question: what signal in the neural activities at time step 2 and 3 produces the representation changes seen at step 4 in Figure 6 ?

We hypothesized that the network does reinstate representations of the items and decision axis, but under a *recoded* form, which would facilitate learning under the constraints imposed upon the network. One such constraint (we surmised) is heterogeneous plasticity. Recall that the recurrent weights are not purely Hebbian, but are rather the sum of a fixed (non-plastic) $\mathbf{W}_{rec}$ and a weighted plastic component $\mathbf{A} \odot \mathbf{P}(t)$ (see Equation 1). Thus, simple Hebbian learning between $\boldsymbol{\psi}_{t1}$ at time $t-2$ and $\mathbf{w}_{out}$ at time $t-1$ would indeed shift the $\mathbf{P}(t)$ component in the correct direction (towards increasing alignment between $\boldsymbol{\psi}_{t2}$ and $\mathbf{w}_{out}$), but the net effect on the *total* recurrent weights $\mathbf{W}_{rec} + \mathbf{A} \odot \mathbf{P}(t)$ would be more complex and unpredictable. To take an extreme example, if $\mathbf{A}$ were fully-negative and larger than $\mathbf{W}_{rec}$, the actual effect on total weights would be in the opposite direction.

What would be the "ideal" activation vectors $\mathbf{r}(t-2)$ and $\mathbf{r}(t-1)$ that would produce the desired weight change in *total* recurrent weights at time step t? Since no obvious analytical answer is available, we address this question by an optimization procedure. Recall that the desired change is to shift future step-2 representations of each item towards greater alignment with the output weight vector. Therefore, we seek to find the recoded representations $\tilde{\boldsymbol{\psi}}_{t1}(X)$ and $\tilde{\boldsymbol{w}}_{out}$ which, when represented in network activity at times $t-2$ and $t-1$ (and thus modifying $\mathbf{P}(t)$ proportionally to their outer product, due to Hebbian learning) would cause a net change in the *total* recurrent weights, such that future presentations of the original $\boldsymbol{\psi}_{t1}(X)$ would produce an output more similar to the original $\boldsymbol{w}_{out}$.

We initialize $\tilde{\boldsymbol{\psi}}_{t1}(X)$ (for all items X) and $\tilde{\boldsymbol{w}}_{out}$ to small random vectors, and repeatedly perform the following procedure:

1. Compute the outer product $\tilde{\boldsymbol{W}}_{prod} = \tilde{\boldsymbol{w}}_{out}\tilde{\boldsymbol{\psi}}_{t1}^{\mathsf{T}}$
2. Substituting this outer product for the plastic weights $\mathbf{P}(t)$ in the total recurrent weights, recompute the recurrent operation on the original step-1, feedforward representations: $\tilde{\mathbf{y}} = (\mathbf{W}_{rec} + \mathbf{A} \odot \tilde{\boldsymbol{W}}_{prod})\,\boldsymbol{\psi}_{t1}$ (recall that $\mathbf{W}_{rec}$ and $\mathbf{A}$ are fixed within each episode)
3. Compute the alignment between this output and the decision axis $\mathbf{w}_{out}$: $J = \mathrm{Corr}(\tilde{\mathbf{y}}, \mathbf{w}_{out})$
4. Perform one step of gradient ascent of J over $\tilde{\boldsymbol{w}}_{out}$ and $\tilde{\boldsymbol{\psi}}_{t1}$.

This procedure is performed in parallel over the batch of 2000 individuals, each with randomly generated stimuli. Because each run has its separate stimuli, we maintain a separate set of $\tilde{\boldsymbol{\psi}}_{t1}(X)$ for each of the 2000 runs. By contrast, we consider $\tilde{\boldsymbol{w}}_{out}$ as an intrinsic feature of the network, and thus use only one $\tilde{\boldsymbol{w}}_{out}$ for the whole batch. We use the Adam optimizer with learning rate 1e-3 and weight decay 1e-2 for regularization.

It is important to note that these recoded representations were not found by specifically exploring network activity at time steps 2 and 3. Rather, they constitute hypotheses about what learning-appropriate representations at these

times "should" look like, at least in part. Clearly many other factors affect the structure of ideal representations, e.g. not interfering with the main task: the reinstated representations should minimize interference with ongoing task-relevant signals, i.e. response at time step 2 and reward at time step 3 (see **(author?)** [46] and references therein for examples of interference-minimizing recoding of representations in neural data). From Fig.s Figure 7, Figure 8, and Extended Data Figure 3, we get confirmation that our "guessed" representations, based on heterogeneous plasticity alone, are adequate probes to detect the reinstatement of items by the network at these time steps, whereas the original unadapted $\boldsymbol{\psi}_{t1}(X)$ and $\boldsymbol{w}_{out}$ are not (Fig. Figure 7).

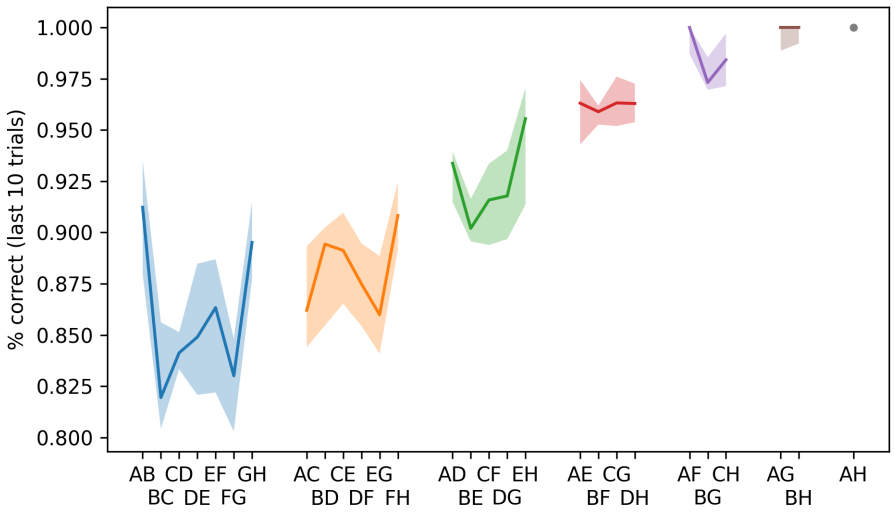
S9 Robustness to massed presentation

Some existing models of transitive inference are disrupted when one single pair is shown a large number of times [11]. Following existing practice, we ran the trained network on an extended episode, in which 20 additional trials (all with the pair DE) were inserted between the 20 adjacent-only trials and the 10 all-pair trials. No other modifications were made. Again, we performed 2000 runs, each with independently generated stimuli. As shown in Supplementary Figure 6, there was little impact on performance, whether on transitive inference, the symbolic distance effect, or the end-item effect.

We note that this robustness is immediately explained by the fact that plasticity mostly occurs in response to errors. On correct responses, the network applies only little neuromodulation at all at time $t = 4$ (Figure 5C), and thus representations are not modified much. This prevents the repeated presentations of a single pair from disrupting the representations of this pair, since not much change occurs once the pair is in the correct order.

S10 Importance of independent plasticity parameters

The method described above relies on independent plasticity parameters for each neuron, stored in the $\mathbf{A}$ matrix and meta-optimized between episodes (Section 4.2). Could similar results be achieved with a single, scalar plasticity parameter applying equally to all connections? We ran the same experiment as in the main text, replacing the $\mathbf{A}$ matrix with a single scalar α, identical for all connections (but also meta-optimized between episodes, just like the coefficients of $\mathbf{A}$ were in the original experiment). This resulted in total loss of ability to obtain the active, cognitive solution based on reinstatement and delayed learning. Among 30 runs, all runs settled on the lower-performance, passive plasticity solution, or failed entirely (Supplementary Figure 8, left). As a result, they were unable to perform list-linking (Supplementary Figure 8, right). This suggests that the flexibility provided by independent plasticity coefficients (which allows different connections to be more or less plastic) is important in enabling the more complex



Supplementary Figure 6: Performance for massed-presentation experiment, grouped by pair. Conventions are as in Figure 2 (median and inter-quartile range of success rate, for each pair, across 10 subsets of 200 episodes each). Compare with Figure 2.

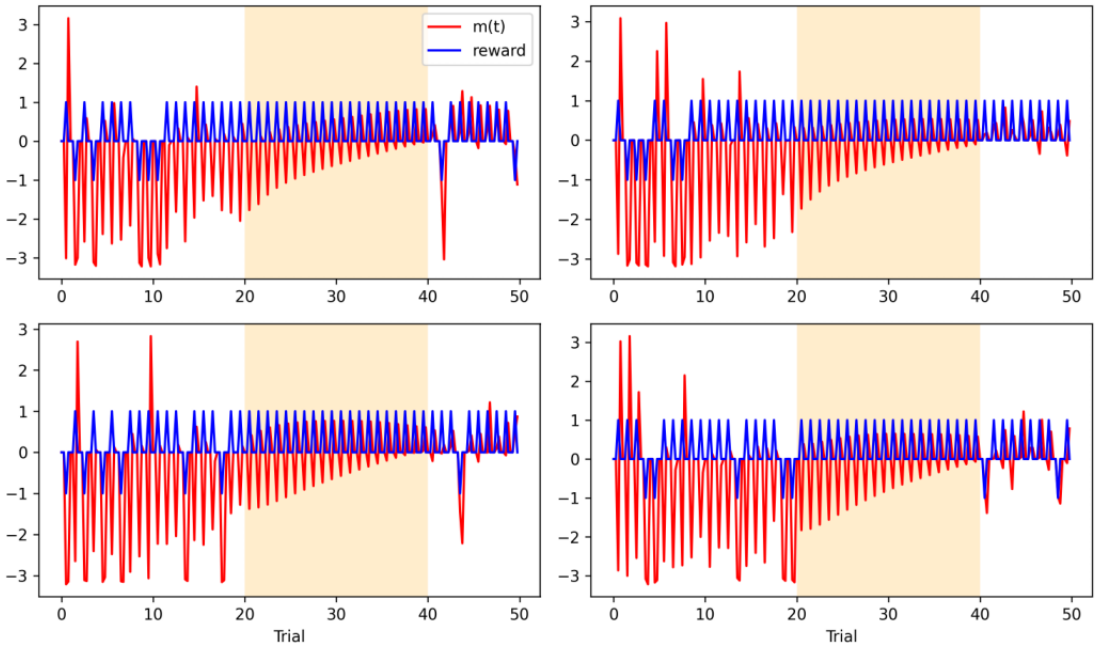
solution, based on reinstatement of recoded representations for delayed learning, and capable of fast knowledge reassembly.

S11 A suboptimal, passive solution to relational learning

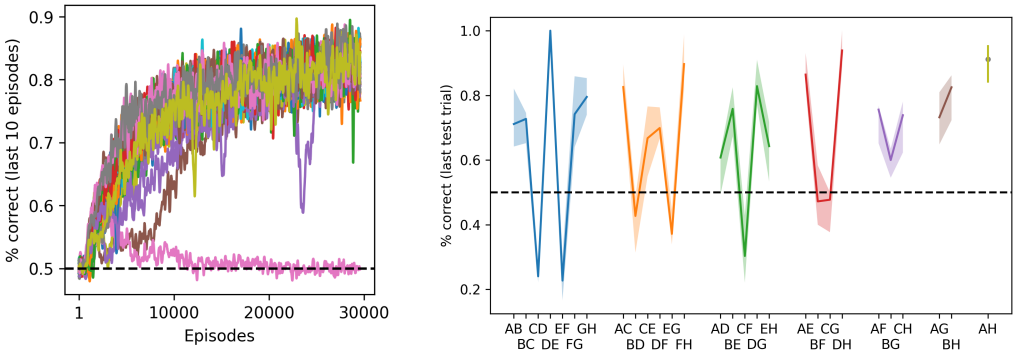
Here we describe results from the alternative, suboptimal solution sometimes found by the system, as discussed in section 3.4. As shown in Supplementary Figure 9, this solution uses reward-selective neuromodulation $m(t)$ at reward time, while $m(t = 4)$ seems unrelated to reward.

Figure 4 shows that this alternative solution still maps each single item to a learned representation that encodes its rank, though with a somewhat flatter curve near the middle of the series. As shown in Extended Data Figure 1, the network still exhibits a symbolic distance effect, confirming that the overall representational scheme is similar. However, performance in list-linking conditions is at or below chance.

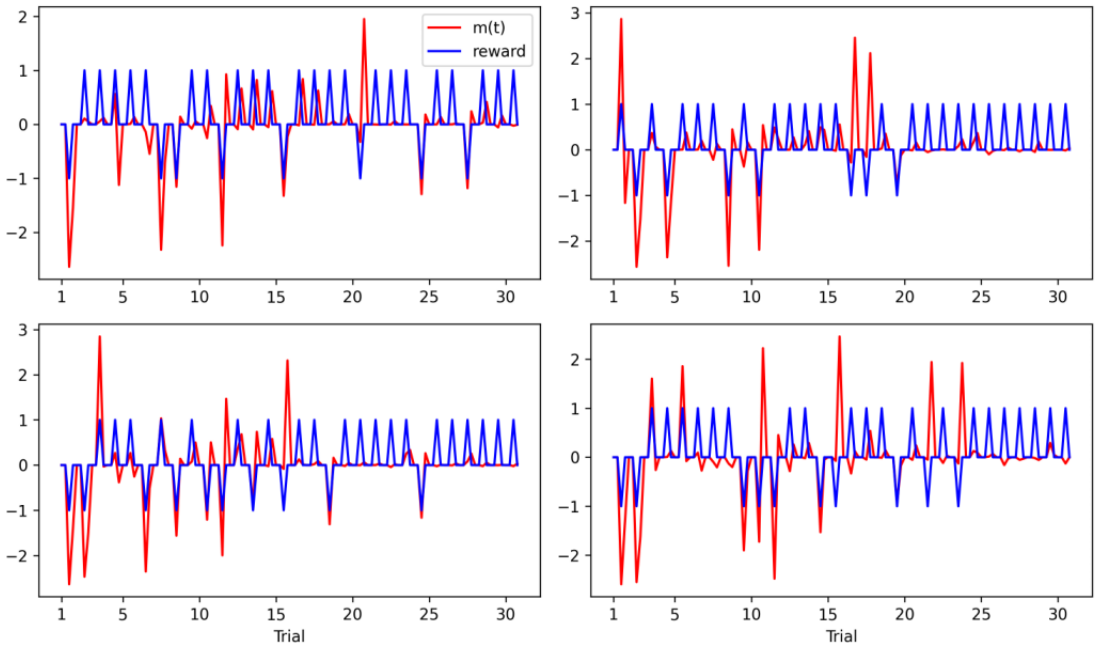
What might cause this deficit in list-linking? In Supplementary Figure 11, we see that representation changes at a given trial occur at reward time only. Importantly, representations are only shifted for the items presented at the current trial. This explains failure of list-linking (which requires transfer of



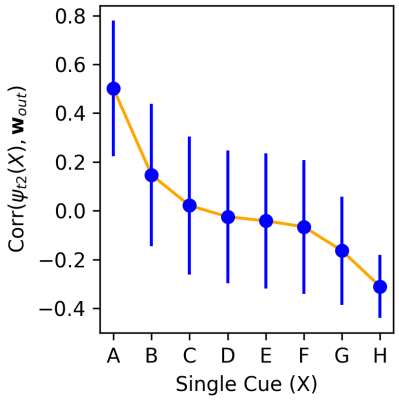
Supplementary Figure 7: Traces of neuromodulatory output $m(t)$ (red) and reward signal (blue) for 4 episodes with massed presentation of the pair DE between trials 20 and 40.



Supplementary Figure 8: Enforcing a common, scalar plasticity coefficient over the entire network prevents the emergence of the cognitive solution. Left: Performance is above chance, but at the lower level (compare with Figure 2). Right: All networks fail the list-linking task (representative example, again showing median and inter-quartile range of success rate for each pair over 10 sets of 200 episodes each; compare with Figure 2 and Figure 3).



Supplementary Figure 9: Traces of $m(t)$ (red) and reward signal (blue) over one episode, for four different runs, using a suboptimal network. Conventions are as in Figure 5



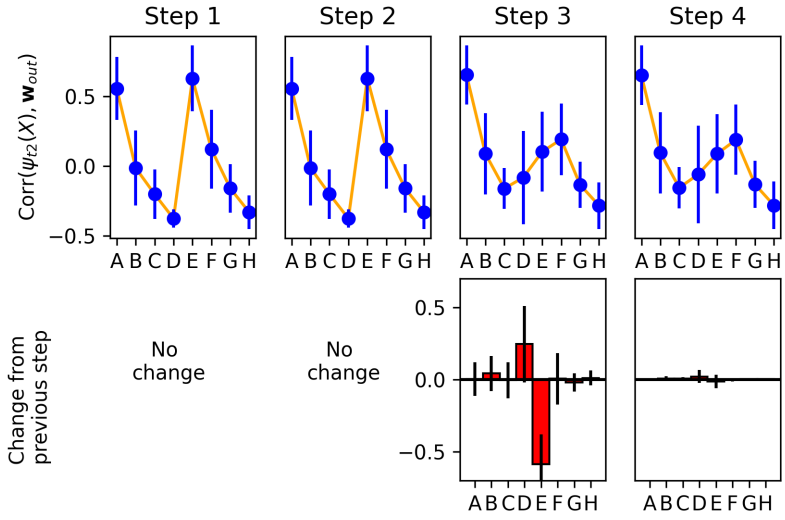
Supplementary Figure 10: Correlation between the step-2 representation of each item $\boldsymbol{\psi}_{t2}(X)$ and the output weight vector $\mathbf{w}_{out}$, at trial 20, for the suboptimal network. Mean and standard deviation over 2000 separately run episodes. Alignment between step-2 representation and output weight vector still monotonically encodes rank, but with a flatter profile than in the standard network (see Figure 4).

information from currently shown items to other not-shown items).

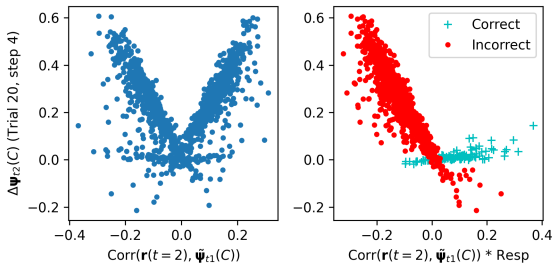
As mentioned in the main text, almost all meta-trained networks followed one of the two solutions described in the present paper, up to mutually compensating sign changes that leave the fundamental mechanism invariant. The balance of the two solutions may change according to various experimental choices (for example, forcing $\mathbf{A}$ to contain only positive values increased the proportion of suboptimal solutions). As shown in Figure 1, the process as described in the Methods greatly favours the higher-performance cognitive solution over the passive one.

S12 Similarities of representations at steps 1 and 2

Here we explore whether the recurrent operation between step 1 and step 2 preserves the structure of similarities between representations. The step-2 representations, being projections of the step-1 representations through the learned, recurrent plastic weights, do preserve a small amount of the structure of step-1 representations on average. However, this amount is variable and unreliable from one episode to the next. We computed the pairwise correlations between all pairs of step-1 representations, and similarly the pairwise correlations between all pairs of step-2 representations, and then computed the correlation between these two vectors as a measure of similarity preservation. Over a batch of 2000



Supplementary Figure 11: Same as Figure 6, but for the suboptimal, "passive" network. Correlation of each item's step-2 (learned) representation $\psi_t 2(X)$ with the output weight vector $\mathbf{w}_{out}$, measured separately at each time step of trial 20. Conventions as in Figure 6, showing mean and standard deviations over 1000 episodes. Representation changes occur at time step 3 (reward delivery) rather than time step 4, and only for the items shown in the current trial.



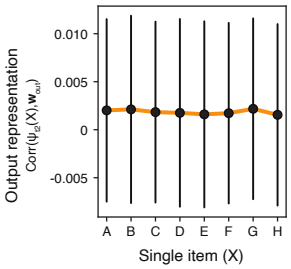
Supplementary Figure 12: The strength of reinstated, recoded item $\tilde{\boldsymbol{\psi}}_{t1}$ at time step 2 predicts the magnitude of the shift in step-2 representation $\Delta\boldsymbol{\psi}_{t2}$ that takes place for this same item at step 4 of the same trial (Supplementary Figure 12). This is consistent with the reinstatement of the item being critical for this item to undergo learning and adaptation of its learned representatio. Left: Signed amplitude of reinstated vector $\tilde{\boldsymbol{\psi}}_{t1}(C)$ at time step 2 predicts representation change for C at time step 4, in trial 20 (with pair DE or ED shown as stimulus for this trial). Right: Same data but multiplying x by the sign of the response given for this trial, with separate coloring for correct and incorrect responses. Data contains $\tilde{1}000$ runs with DE not shown before trial 20 (leading to incorrect responses) and $\tilde{1}00$ trials in which it was (leading to mostly correct responses).

runs (all with identical network parameters, but each with different stimuli and plastic weights), this value ranged from -0.34 to 0.78 with a mean of 0.21 and a standard deviation of 0.17. 12% of these values were negative.

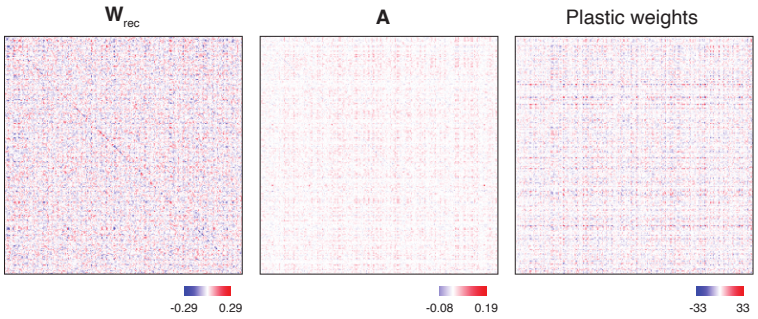
We also recomputed the same information as for Figure 4 but for step-1 representations (Supplementary Figure 13). As expected, the resulting curve was flat across cues: step-1 representations, being fixed linear projections (through the input weights) of the stimuli (which are themselves random binary vectors) do not carry information about item rank, unlike step-2 representations.

S13 Importance of joint training of weight matrices

Correct network operation relies critically on the mutual tuning between the various weight matrices, resulting from slow, outer-loop meta-optimization between episodes. We trained a separate network with the procedure described above, verified that it used the active, reinstating solution (with the same sign convention as the network used for other figures in this paper), and then swapped the output weights $\mathbf{W}_{out}$ of this new network into the network used above. The resulting mixed network's performance on a test episode was at chance for all pairs, as expected (see Supplementary Figure 15). Importantly, this was despite the fact that the network still correctly registered reward events (through



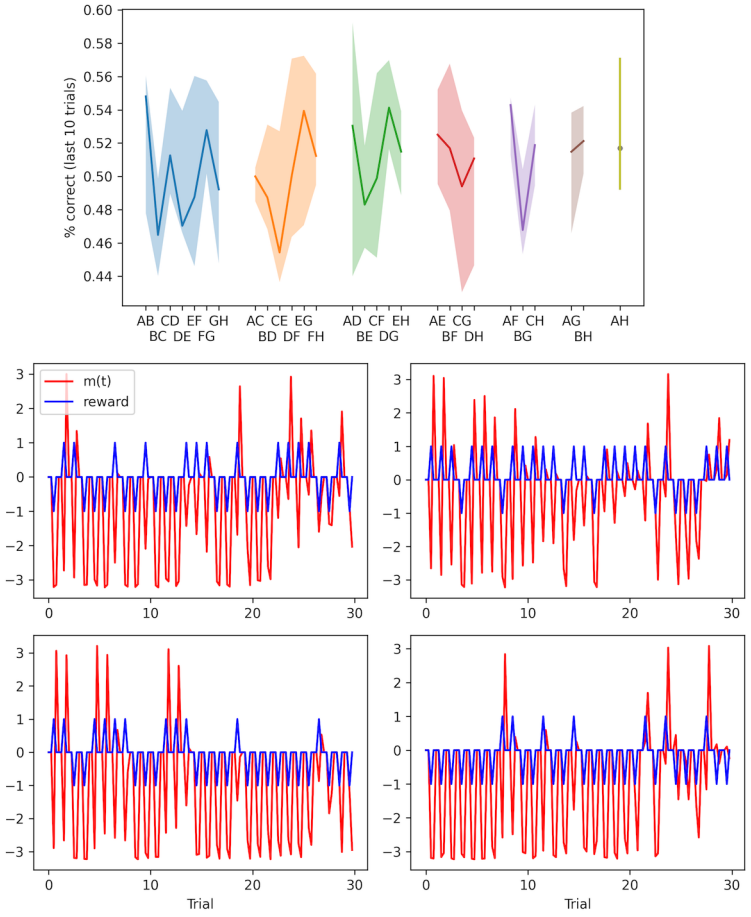
Supplementary Figure 13: Correlation between the step-1 representation of each item $\boldsymbol{\psi}_{t1}(X)$ and the output weight vector $\mathbf{w}_{out}$, at trial 20. Mean and standard deviation over 2000 separately run episodes. Unlike step-2 representations, alignment between step-1 representation and output weight vector carries no information about rank, as expected (compare with Figure 4).



Supplementary Figure 14: Weights of the trained recurrent weight matrix $\mathbf{W}_{rec}$, plasticity coefficient of recurrent connections $\mathbf{A}$, and plastic weights $\mathbf{P}(t_{last})$ at the end of an episode. All values are from the same representative network (for the "active" solution) used in other figures.

its unmodified input weights $\mathbf{W}_{in}$) and produced expected neuromodulatory outputs.

In addition, swapping input weight matrices $\mathbf{W}_{in}$ also drove performance to chance level, and eliminated any obvious pattern in neuromodulation. This is expected because $\mathbf{W}_{in}$ contains the weights that convey the reward signal to appropriate neurons in the main recurrent network.



Supplementary Figure 15: Swapping the weight matrices of successful networks abolishes performance. We swapped the output weights $\mathbf{W}_{out}$ of a new trained network into the same network used for other figures in this paper. Performance for all pairs in a test episode (across 2000 runs) were at chance (top). This is despite the fact that neuromodulation in response to stimuli and rewards remained unaffected (bottom). Conventions are as in Figures Figure 2 and Figure 5.