Neural network models for the evolution of associative learning.

Emiliano Méndez Salinas, Franz J. Weissing and Magdalena Kozielska

Overview of Appendices:

Appendix A. Calculation of the parameters that optimise network performance

Appendix B. Updating behaviour of different learning rules

Appendix C. Variability between replicates in the simple learning task

Appendix D. Fitness landscapes and evolutionary trajectories for linear networks for a simple learning task and E_0=0.5

Appendix E. Evolution of weights for linear networks with bias in the simple learning task and E_0=0.5.

Appendix F. Summary plots of all results.

Appendix A. Calculation of the parameters that optimise network performance

In this appendix, we will derive for all linear networks analytical criteria on the network parameters (and the initial estimate E_0) that need to be satisfied in order to optimise network performance or, equivalently, to minimise the expected discrepancy between the reward probability estimated by the network and the "true" reward probability. With the help of these criteria, the optimal parameter setting can be calculated numerically.

A1. Networks of type N1, fixed number *S* of observations

We start with a simple network of type N1 (see Fig 2 in the main text), which is characterised by three parameters w_R, w_E, and b, which correspond to the weights of the connections from R_t (reward value at time t) and E_t to E_{t+1} and the bias of the output node E_{t+1}. In such a network, the reward probability is updated according to the linear relationship

$$E_{t+1} = w_R R_t + w_E E_t + b\,. \tag{A1}$$

Eq. A1 is a linear recurrence equation that can be solved explicitly. Starting with E_0 at time zero and iterating eq. (A1) S times, we obtain the following expression for the estimate E_S after S updates:

$$E_S = w_R \cdot \left(\sum_{t=0}^{S-1} w_E^{S-1-t} R_t \right) + w_E^S \cdot E_0 + b \cdot \sum_{t=0}^{S-1} w_E^t \tag{A2}$$

To simplify the equations in the following derivations, we introduce the following notation:

$$S_1 = S_1(w_E) = \sum_{t=0}^{S-1} w_E^t = \sum_{t=0}^{S-1} w_E^{S-1-t} = \frac{1-w_E^S}{1-w_E}$$

$$S_2 = S_2(w_E) = \sum_{t=0}^{S-1} w_E^{2t} = \sum_{t=0}^{S-1} w_E^{2(S-1-t)} = \frac{1-w_E^{2S}}{1-w_E^2} = S_1 \cdot \frac{1+w_E^S}{1+w_E}$$

$$C = C(w_E, E_0, b) = w_E^S \cdot E_0 + b \cdot \sum_{t=0}^{S-1} w_E^t = w_E^S \cdot E_0 + b \cdot S_1 \ .$$

With the above notation, eq. A2 is of the form:

$$E_S = w_R \cdot \left(\sum_{t=0}^{S-1} w_E^{S-1-t} R_t \right) + C \, . \tag{A3}$$

This equation gives us the estimated reward probability for a given sequence $R_0, R_1, \ldots, R_{S-1}$ of rewards. The R_t take on the values of 0 and 1, where the outcome $R_t = 1$ occurs with probability P. Hence, the reward probability P corresponds to the expected value of the individual rewards R_t. We now calculate the expected value of the reward estimate E_S given that the "true" reward probability is P:

$$\mathbf{E}(E_S \mid P) = w_R \cdot \left(\sum_{t=0}^{S-1} w_E^{S-1-t} P \right) + C = w_R P \cdot S_1(w_E) + C \, . \tag{A4}$$

For the calculations below, we also need the expected value of E_S^2 given P. To this end, the expression on the right-hand side of eq. A3 needs to be squared, and the expectation given P needs to be calculated. A straightforward calculation yields the outcome:

$$\mathbf{E}(E_S^2 \mid P) = [\mathbf{E}(E_S \mid P)]^2 + w_R^2 \cdot S_2(w_E) \cdot P(1-P) \, . \tag{A5}$$

Now we can calculate the "loss function", that is, the expected squared error $\mathbf{E}(\delta^2 \mid P)$ made by the network when estimating the reward probability, given that the reward probability is P:

$$\mathbf{L}(P) = \mathbf{E}(\delta^2 \mid P) = \mathbf{E}((E_S - P)^2 \mid P) = \mathbf{E}(E_S^2 \mid P) - 2 \cdot \mathbf{E}(E_S \mid P) \cdot P + P^2 \, . \tag{A6}$$

Plugging eqs A4 and A5 into eq. A6 and rearranging yields the following formula for the expected squared error $\mathbf{E}(\delta^2 \mid P)$ for a given value of *P*:

$$\mathbf{L}(P) = [(1 - w_R S_1)^2 - w_R^2 S_2]P^2 + [w_R^2 S_2 - 2C(1 - w_R S_1)]P + C^2 . \quad \text{(A7)}$$

The network performance needs to be judged for a whole spectrum of *P* values, where *P* is uniformly distributed over the unit interval. Therefore, the expected squared error made by the network is obtained by averaging $\mathbf{L}(P)$ over all *P*. This average is given by an integral that, in view of eq. A7, can be readily calculated:

$$\bar{\mathbf{L}} = \mathbf{E}(\mathbf{L}(P)) = \mathbf{E}(\delta^2) = \int_0^1 \mathbf{L}(P)\, dP = \tfrac{1}{3} w_R^2 (S_1^2 + \tfrac{1}{2} S_2) - \tfrac{2}{3} S_1 w_R + C S_1 w_R + \tfrac{1}{3} - C + C^2 . \quad \text{(A8)}$$

Now we can calculate the optimal parameter combination w_R, w_E, *b* (and E_0 for an evolving initial estimate), that is, the parameter combination that minimizes $\bar{\mathbf{L}}$. Tho this end, we calculate the partial derivatives of expression of $\bar{\mathbf{L}}$ in eq. A8 with respect to these parameters:

$$\frac{\partial \bar{\mathbf{L}}}{\partial w_R} = \tfrac{2}{3}(S_1^2 + \tfrac{1}{2} S_2) w_R - \tfrac{2}{3} S_1 + C S_1$$

$$\frac{\partial \bar{\mathbf{L}}}{\partial w_E} = \tfrac{1}{3} w_R^2 \left(2 S_1 \frac{\partial S_1}{\partial w_E} + \tfrac{1}{2} \frac{\partial S_2}{\partial w_E} \right) - \tfrac{2}{3} w_R \frac{\partial S_1}{\partial w_E} + w_R \left(\frac{\partial C}{\partial w_E} S_1 + C \frac{\partial S_1}{\partial w_E} \right) - \frac{\partial C}{\partial w_E} + 2C \frac{\partial C}{\partial w_E}$$

$$\frac{\partial \bar{\mathbf{L}}}{\partial b} = S_1 w_R \frac{\partial C}{\partial b} - \frac{\partial C}{\partial b} + 2C \frac{\partial C}{\partial b} = w_R S_1^2 - S_1 + 2C S_1$$

$$\frac{\partial \bar{\mathbf{L}}}{\partial E_0} = S_1 w_R \frac{\partial C}{\partial E_0} - \frac{\partial C}{\partial E_0} + 2C \frac{\partial C}{\partial E_0} = S_1 w_R w_E^S - w_E^S + 2C w_E^S$$

The partial derivatives in the second of these equations are given by:

$$\frac{\partial S_1}{\partial w_E} = \frac{1}{1 - w_E} \left(\frac{1 - w_E^S}{1 - w_E} - S\, w_E^{S-1} \right)$$

$$\frac{\partial S_2}{\partial w_E} = \frac{1}{1 - w_E^2} \left(\frac{1 - w_E^{2T}}{1 - w_E^2} - 2S\, w_E^{2S-1} \right)$$

$$\frac{\partial C}{\partial w_E} = E_0 S\, w_E^{S-1} + \frac{b}{1 - w_E} \left(\frac{1 - w_E^S}{1 - w_E} - S\, w_E^{S-1} \right).$$

To find the optimal values of network parameters (and E_0) one needs to equate the partial derivatives of $\bar{\mathbf{L}}$ with respect to the parameters to be optimized with zero. For example, for network N1 and fixed E_0, the following system needs to be solved:

$$\left\{ \frac{\partial \bar{\mathbf{L}}}{\partial w_R} = 0,\ \frac{\partial \bar{\mathbf{L}}}{\partial w_E} = 0,\ \frac{\partial \bar{\mathbf{L}}}{\partial b} = 0 \right\}.$$

For all scenarios considered in the main text, we obtained a unique solution when solving the corresponding system of equations numerically. In all cases, this solution corresponded to a minimum of the average loss function which we also obtained numerically (Table A1).

Table A1. Minimal values of the loss function and the corresponding network weights for different networks and different scenarios. Parameters that were fixed are in normal fond, calculated parameters are in bold.

Network type	w_R	w_E	b	E_0	Minimal expected error $\bar{\mathbf{L}} = \mathbf{E}(\partial^2)$
Fixed number of updates $S = 10$					
RW rule (E_0=0.5)	**0.135**	0.865[1]	0.0	0.5	0.0160
RW rule (E_0=0.0)	**0.196**	0.804[1]	0.0	0.0	0.0221
RW rule (evolving E_0)	**0.135**	0.865[1]	0.0	**0.500**	0.0160
N1$_0$ (E_0=0.5)	**0.133**	**0.870**	0.0	0.5	0.0159
N1$_0$ (E_0=0.0)	**0.0952**	**1.000**	0.0	0.0	0.0159
N1$_0$ (evolving E_0)	**0.0833**	**1.000**	0.0	**0.0833**	0.0139
N1 (E_0=0.5)	**0.0833**	**1.000**	**-0.0417**	0.5	0.0139
N1 (E_0=0.0)	**0.0834**	**1.000**	**0.00833**	0.0	0.0139
N1 (evolving E_0)[2]	**0.0834**	**1.0**	**-0.0426**	**0.510**	0.0139
Variable number of updates					
RW rule (E_0=0.5)	**0.167**	0.833[1]	0.0	0.5	0.0278
RW rule (E_0=0.0)	**0.333**	0.667[1]	0.0	0.0	0.0556
RW rule (evolving E_0)	**0.167**	0.833[1]	0.0	**0.500**	0.0278
N1$_0$ (E_0=0.5)	**0.167**	**0.833**	0.0	0.5	0.0278
N1$_0$ (E_0=0.0)	**0.333**	**0.667**	0.0	0.0	0.0556

$N1_0$ (evolving E_0)	**0.167**	**0.833**	0.0	**0.500**	0.0278
N1 (E_0=0.5)	**0.166**	**0.835**	**-0.0006**	0.5	0.0278
N1 (E_0=0.0)	**0.300**	**0.517**	**0.109**	0.0	0.0443
N1 (evolving E_0)	**0.166**	**0.835**	**-0.0006**	**0.500**	0.0278

[1] For the Rescorla-Wagner rule w_E was set to $1-w_R$ (see section A2).

[2] One example. Infinitely many networks minimize the expected error.

A2. Rescorla-Wagner learning rule

Rescorla-Wagner rule can be seen as a special case of $N1_0$ type of network where $\beta = w_R$ and $w_E = 1-w_R$. Therefore, we found the minimal value of loss function and optimal learning rate β as described above but by constraining w_E to $1-w_R$ (see Table A1).

A3. Networks of type N1, variable number of observations

For the more complicated learning task with a variable number of observations T (see Section 4.2. of the main text), the expected average loss needs to be calculated for all possible values of T. To indicate the dependence of $\bar{\mathbf{L}}$ on T, we rewrite eq. A8 in the form

$$\bar{\mathbf{L}}_T = \tfrac{1}{3} w_R^2 (S_1(T)^2 + \tfrac{1}{2} S_2(T)) - \tfrac{2}{3} S_1(T) w_R + C(T) S_1(T) w_R + \tfrac{1}{3} - C(T) + C(T)^2 . \qquad \text{(A9)}$$

Let q_T denote the probability that the number of observations is equal to T. For the scenario considered in the main text, q_T is given by the geometric distribution:

$$q_T = q_1 (1-q_1)^{T-1} = \frac{1}{S}\left(1-\frac{1}{S}\right)^{T-1}, \qquad \text{(A10)}$$

where q_1 is the probability that the current observation is the final one. The expected number of observations is $S = 1/q_1$, yielding the second equation in A10 in terms of S. Therefore, the expected average loss for this scenario is given by taking the average of $\bar{\mathbf{L}}_T$ over all T:

$$\bar{\mathbf{L}} = \sum_{T=1}^{\infty} q_T \bar{\mathbf{L}}_T = \frac{1}{S} \cdot \sum_{T=1}^{\infty} \left(1-\tfrac{1}{S}\right)^{T-1} \bar{\mathbf{L}}_T . \qquad \text{(A11)}$$

In our simulations, the number of observations was limited to 20. Therefore, q_T was given by a truncated geometric distribution, leading to the expected average loss:

$$\bar{\mathbf{L}} = \sum_{T=1}^{\infty} q_T \bar{\mathbf{L}}_T, \text{ where } q_T = \begin{cases} \frac{1}{S}\left(1-\frac{1}{S}\right)^{T-1} \text{ for } & T < 20 \\ \left(1-\frac{1}{S}\right)^{2T-1} \text{ for } & T = 20 \end{cases} \quad \text{(A12).}$$

As for the scenario with the fixed number of updates, the values of w_E, w_R, b (and E_0) that least to minimisation of the loss function were obtained numerically.

A4. General linear networks

The above calculations are also relevant for more complicated networks with a linear transfer function. Let us illustrate this for a network of type N2. In this case, the estimate of the reward probability is updated according to the equation

$$E_{t+1} = \left(\sum_{i=1}^{3} w_{Ri} w_{iE}\right) R_t + \left(\sum_{i=1}^{3} w_{Ei} w_{iE}\right) E_t + b, \quad \text{(A13)}$$

where the w_{Ri} and w_{Ei} values represent the weights of connections between the input nodes corresponding to reward and previous estimate to a node x_i in the processing layer, while the w_{iE} represent the connection weights between node x_i and the output node. Notice that eq. A9 describes a *linear* relationship, just as eq. A1 for a network of type N1. This means that all our previous calculations apply to a network of type N2 if we define the compound parameters $w_R = \sum_{i=1}^{3} w_{Ri} w_{iE}$ and $w_E = \sum_{i=1}^{3} w_{Ei} w_{iE}$. As before, we can calculate optimal values for w_R and w_E, but now there are infinitely many combinations of the parameters w_{Ri}, w_{Ei}, and w_{iE} leading to the same update dynamics A9 and the same minimum of expected average loss. This immediately explains why optimal networks of type N2 do not perform better than optimal networks of type N1. However, the evolutionary dynamics can be different in both cases, because networks of type N2 (a) have more evolutionary degrees of freedom, and (b) provide more targets for mutations, potentially hampering adaptive evolution.

A5. Non-linear networks

Analytical expressions for the loss function (expected error) $\bar{\mathbf{L}} = \mathbf{E}(\delta^2)$ could not be found for networks of type **N3** with a non-linear activation function. For such networks, we estimated $\mathbf{E}(\delta^2)$ as follows: Given the network parameters (and the initial estimate E_0 evolved in the simulation), the final estimate E_S can be determined iteratively for any reward sequence (like 1,0,0,1,1,1,0,1,0,0) by updating the estimate S times. For a given reward probability P, the

probability of a reward sequence of length S is given by $P^m(1-P)^{S-m}$, where m is the number of rewards $R=1$ in the sequence. Taking weighted averages of all reward sequences (with the probability of a reward sequence as weighing factor) yields $\mathbf{E}(E_S|P)$, the expected value of the estimate E_S given the true value P. Similarly, we can calculate $\mathbf{E}((E_S-P)^2|P)$, the expected squared deviation between E_S and P made by the network, given the true value of P. We considered 10^5 evenly spaced values of the reward probability P, covering the interval [0,1]. For each value of P, we determined the expected error made by the network as described above. Finally, we calculated the average error over all 10^5 P values, yielding the desired estimate of $\mathbf{E}(\delta^2)$. The best performing network in the population is the one with the lowest $\mathbf{E}(\delta^2)$.

In the more complicated learning task, the number of updates S is not fixed but a stochastic variable. Here, we first calculated the expected loss $\mathbf{E}(\delta^2|S)$ for any given number of updates S. Subsequently, $\mathbf{E}(\delta^2)$ was determined by calculating the weighted average of $\mathbf{E}(\delta^2|S)$, with the probability of encountering S updates as a weighing factor (see section A3).

Appendix B. Updating behaviour of different learning rules

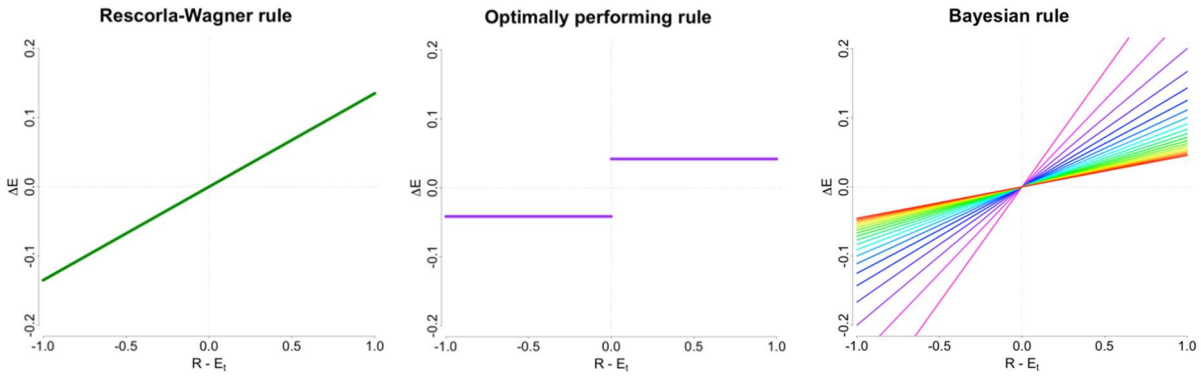


Figure A1 Rescorla-Wagner plots for different learning rules. When the change in estimates ($\Delta E = E_{t+1}-E_t$) in response to an observation R is plotted against the difference R-E_t, the Rescorla-Wagner rule (eq. (1)) produces a straight line with slope β through the origin. The green line corresponds to the best Rescorla-Wagner rule for the simple learning scenario (Section 4.1. of the main text; $\beta^* = 0.135$). The optimally performing rule (eq. (5) in the main text) produces two horizontal line segments (purple). For the Bayesian rule early information has larger effect (steeper slope) than later information, thus for a given R-E_t value, change in estimate (ΔE) is larger for early updates - small values of s (violet end of the rainbow spectrum) and smaller for later updates - large values of s (red end of the rainbow spectrum).

Appendix C. Variability between replicates in the simple learning task

Figure 3 in the main text presents average expected error from 10 replicate simulations. Here, we present an example of variability in network performance in different replicates. $N1_0$ and $N2_0$ networks always reach performance slightly better than the RW rule, but the expected error might get larger in some generations. Similarly, N1 and N2 networks always reach performance of the Optimally Performing Rule, but the expected error might get larger in some generations. Additionally, the time taken to reach the optimal performance vary between replicates. N1 networks tend to take longer to reach optimal theoretical performance and the expected error is more often above the optimum in comparison to N2 network.

Performance of the non-linear networks is more variable between replicates especially for the $N3_0$ networks that usually have the highest error, but sometimes reach performance better than the RW rule and $N1_0$ and $N2_0$ networks.

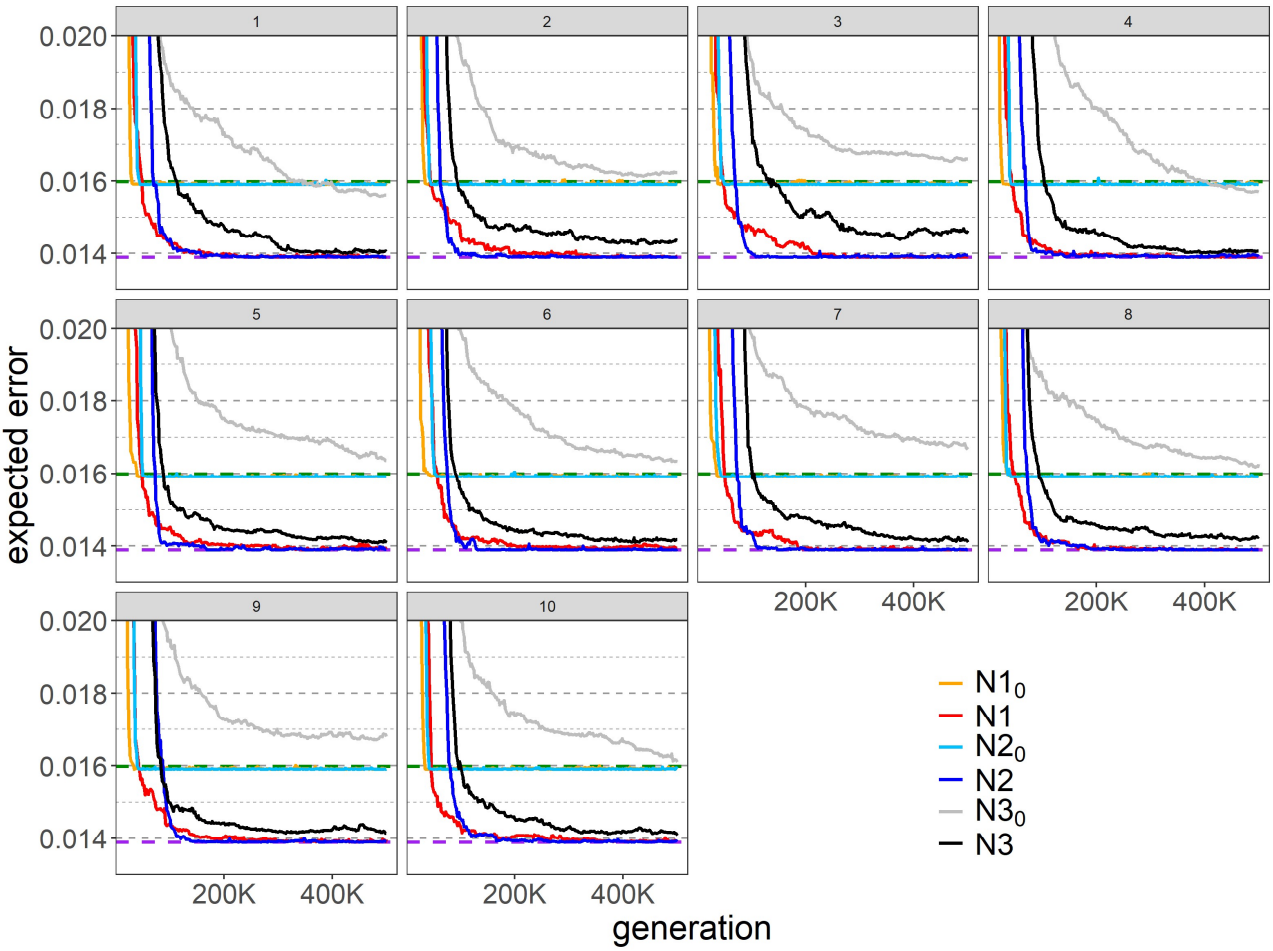


Figure A2 Performance of evolved networks in the simple learning task with $E_0 = 0.5$. The graph shows how the expected error decreases (and hence performance increases) over the generations for each type of network. For comparison, the expected error of the Rescorla-Wagner rule (green dashed line) and the Optimally Performing Rule (purple dashed line) are also shown. Each panel shows an independent replicate. Each solid line represents the best-performing ANN in the population at the given time.

Appendix D. Fitness landscapes and evolutionary trajectories for linear networks for a simple learning task and E_0=0.5

Plotting evolutionary trajectories from the individual based simulations on the fitness landscape (Figure A4) shows that starting from different initial weights combinations, evolution leads to the line of attraction (although not via the shortest route along the selection gradient). Once this line is reached evolution follows along it to the optimal combination of weights (fitness peak). Interestingly, this line of attraction is close, but not equal to the "Rescorla-Wagner line" where $w_R+w_E=1$, represented by the dashed line.

When the weights of evolved $N2_0$ networks are used to calculate the corresponding weight w_E and w_R (corresponding $N1_0$ network; see Appendix A) we can see that evolutionary trajectories on the fitness landscape (Figure A4) are often similar to the ones of $N1_0$ networks, although some combinations of initial weights lead to more direct path toward the peak.

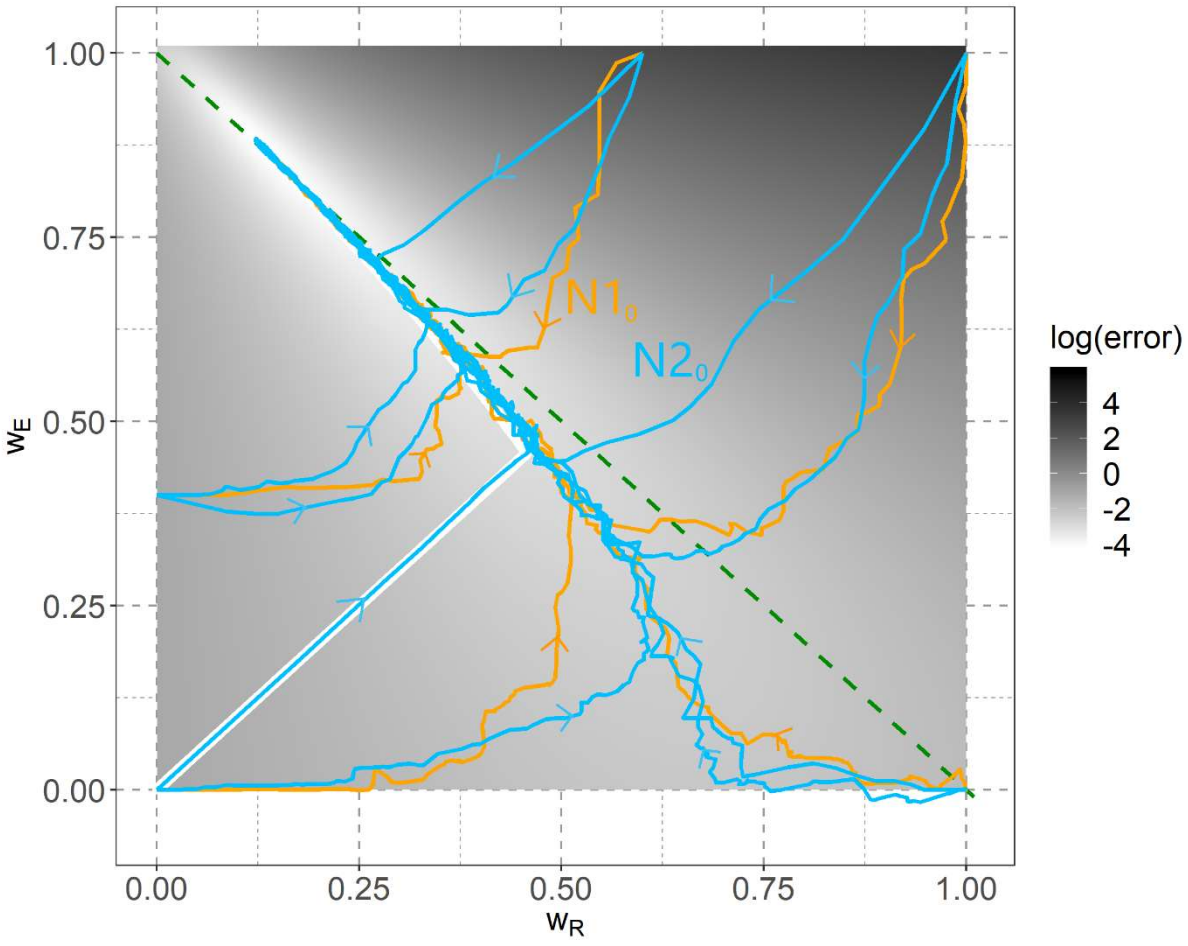


Figure A3 The fitness landscape and evolutionary trajectories for network $N1_0$ and $N2_0$. Lowest expected error is achieved when w_E and w_R values satisfy equation (7) in the main text. In orange – example of evolutionary trajectories taken in individual-based simulations for network $N1_0$ (starting from different initial weight values). In blue – example of evolutionary trajectories taken in individual-based simulations for $N2_0$ starting with different initial weights (see Appendix A for how w_E and w_R can be calculated for $N2_0$ - two blue lines

starting at the same point show two replicates with different initial combination of weights that can be reduced to the same w_E and w_R). Population average w_E and w_R are plotted every 100 generations. In white – path taken by hill-climbing algorithm (path that leads from the starting point (w_R=0, w_E=0) to the minimal error, where each step is taken by increasing or decreasing one or both weights by 0.001 - resolution of our fitness landscape - toward largest error reduction). Dashed green line – RW line where $w_E + w_R = 1$.

Appendix E. Evolution of weights for linear networks with bias in the simple learning task and E_0=0.5.

To get a better understanding of the relationships between network's weights, performance and behaviour, we plotted them together in a figure that shows their respective states at different points in time during their evolution for a N1 network. Compared to the fitness landscape, it helps to have a clearer perception of the temporal dimension of the evolution of the networks. Change in weight values happens very rapidly during the early stages of evolution until they reach a stage in which their weights' values, performance and updating behaviour is equal to that of the RW Rule. After that, evolution proceeds more slowly, with weights, performance and updating behaviour gradually converging to that of the OPR rule.

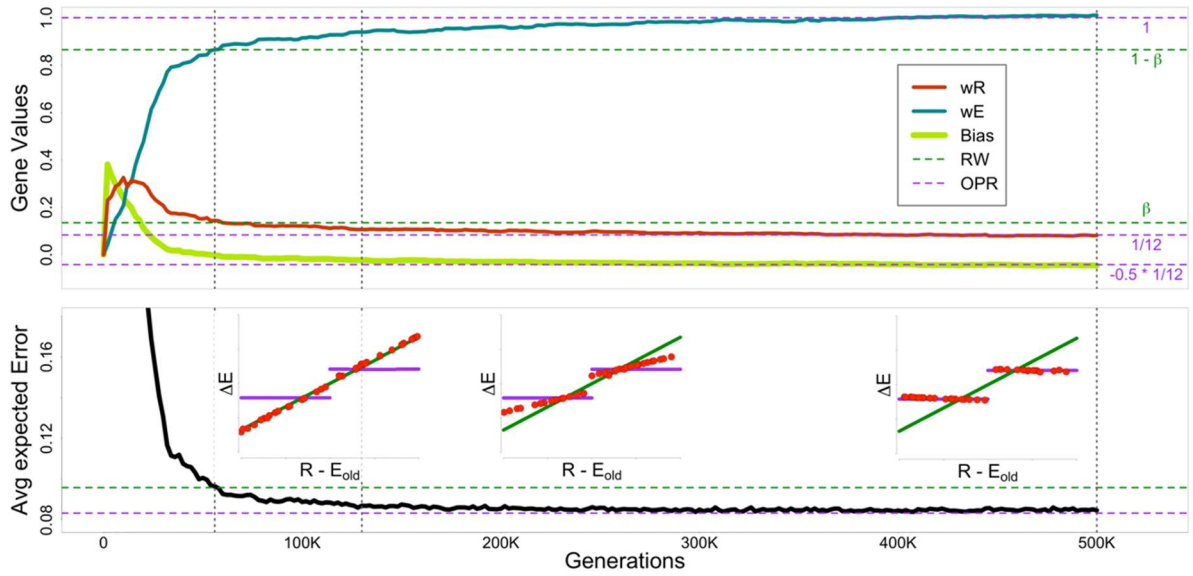
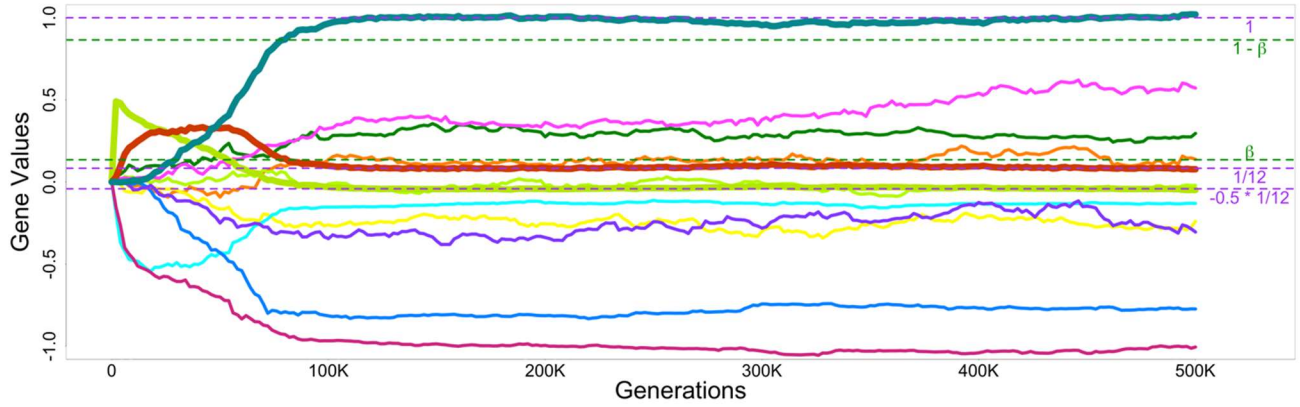


Figure A4 Evolution of weights, performance and updating behaviour of N1 for the simple learning task and initial estimate E_0=0.5. The left vertical dotted line (around generation 60K) marks the time point where the performance of N1 matches performance of RW Rule. At this time point, the weights of N1 are very similar to the parameters of the best RW rule (green dotted horizontal lines in the upper panel). The updating behaviour of N1 is also observed to be practically the same as that of the best RW Rule at this time point (left-side insert in the lower panel). As evolutionary time goes on, N1 evolves toward OPR. By the end of the simulation, marked by the vertical dotted line on the right, N1 has reached equilibrium, its performance is optimal and its weights match those of OPR (purple dotted horizontal lines on the upper panel), resulting in the same updating behaviour as OPR (right-side insert in the lower panel).

To get a better understanding of variability between 2-layer networks in different replicates, we plotted weights of two random replicates. In addition, we plotted calculated corresponding weights of 1-layer network (w_E and w_R – see Appendix A). This helps to understand why networks with very different weights can all have the same performance. As on Figure A6, the networks first go through the stage in which their performance (not shown), bias and w_R and w_E weights are equivalent to the RW rule and then evolve towards the OPR.

(a)



(b)

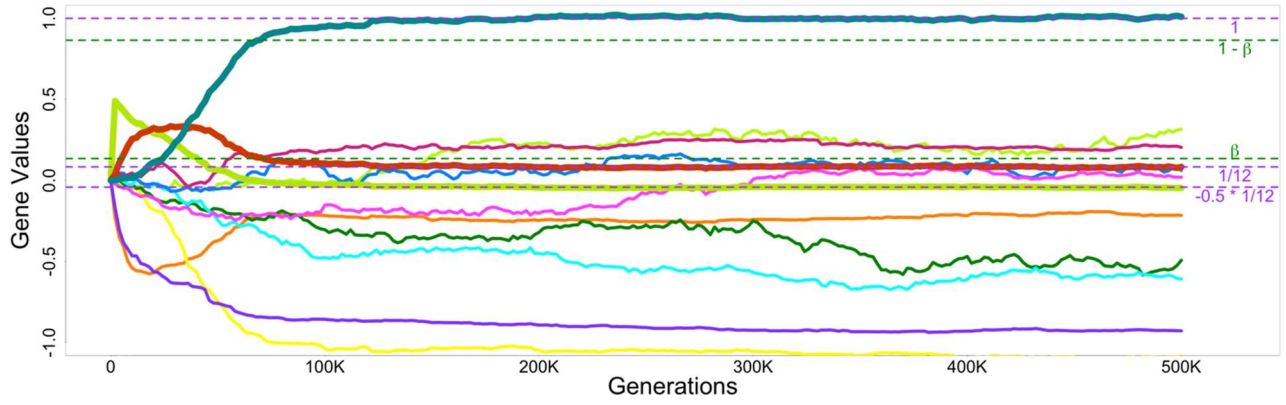


Figure A5. Evolution of weights and performance of N2 through time for the simple learning task and initial estimate E_0=0.5. Panel (a) and (b) show two independent replicates. Thin lines represent individual weights and thick lines corresponding w_E and w_R values (see Appendix A on how to calculate them) or bias (same colours as Figure A4). There is clear difference between replicates in the evolved weights values, but less difference in bias and the calculated w_E and w_R values. Note that there is also a considerable change in individual weights values over time without change in error or the respective w_E and w_R values or bias.

Appendix F. Summary plots of all results.

In addition to the three networks that were analysed in detail in the main text we also considered multiple other networks.

We studied all possible combinations of the following features:

- 1- or 2- layer network,
- Two (E_t and R) or three (E_t, R and $1/s$) input nodes,
- Activation function at nodes in last, processing, both or none of the layers of the network. For the processing layer we used a tanh transformation function as described in the main text. For the output layer we used logistic function – S-shaped transformation function that returns real values between 0 and 1 as it seems most appropriate for estimating probability.
- Bias in the last node, processing nodes or all (non-input) nodes of the network or not bias at all.

The figures in this Supplementary Material give an overview of the performance of all the networks that are possible with the afore mentioned combinations of features, considering also the variability of results across replicate simulations.

In all the figures we present 1-layer networks in top rows and 2-layer networks in bottom rows. Within each of these types, we further divide networks by number of inputs (two or three). For each of them we specify if and where activation functions and biases are applied. As a visual guideline of the structure of the networks, we present some schematic examples and their description in Figure A6.

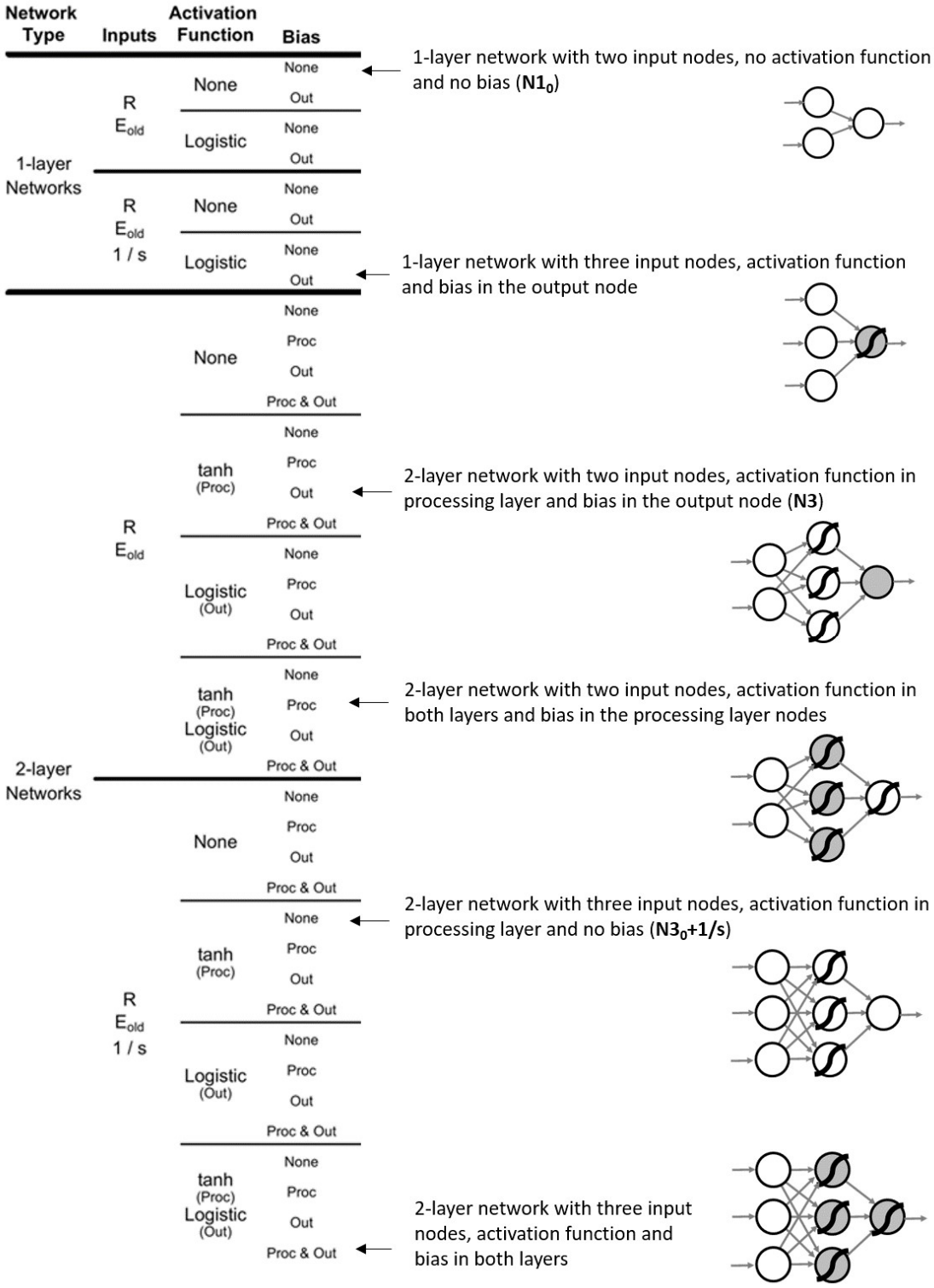


Figure A6. Overview of all networks studied by us with examples of network structure for selected networks. Gray and white circles represent nodes with and without bias (the baseline activation), respectively. Black s-shaped lines represent nodes with non-linear activation function. Other nodes have a linear identity activation function. Notice that the hierarchical classification of combination of features on the left side of the figure is the same as in the following figures.

The six figures below show the performance of different networks after 500K generations of evolution for different learning tasks and different initial estimates E_0. Each line represents the population mean performance over the last 1000 generations from an independent replicate of individual-based simulations. Red and blue lines: results of the simulations that started with weights initially set to zero or random values (between -1 and 1), respectively. For comparison the performance of the best RW rule (green) and the optimal performing rule for simple task (purple) or Bayesian rule for more complicated task (wine red) are shown. Grey fields indicate network that were addresses in the main text: dark grey fields indicate networks N1-N3 and light grey fields indicate networks N1+1/s – N3+1/s (only marked for a more complicated task). An empty network field indicates that the performance was poor and outside the limits of the figure.

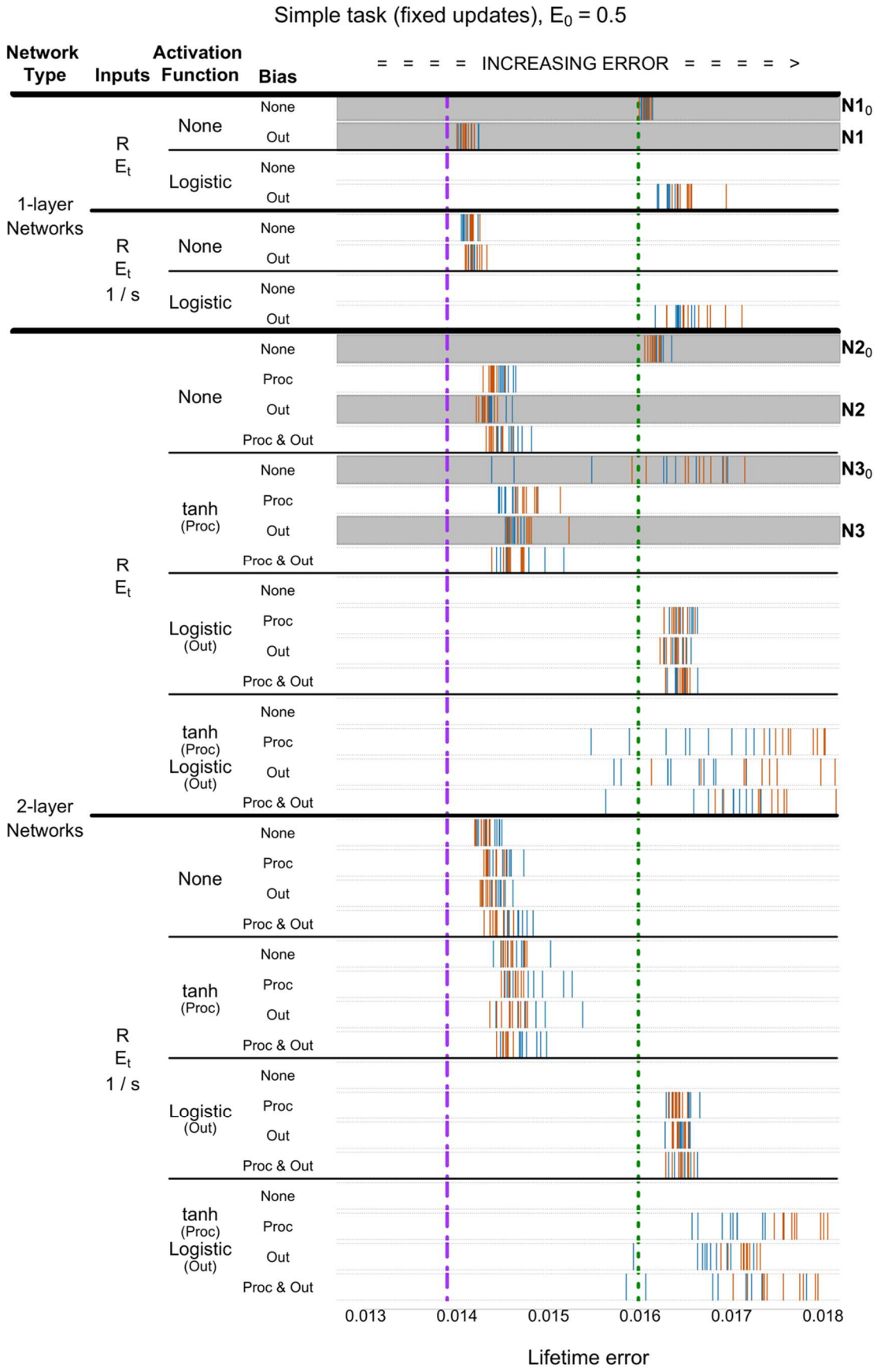


Figure A7. Performance of different networks for simple learning task (fixed number of updates *S*=10) and initial estimate E_0 = 0.5.

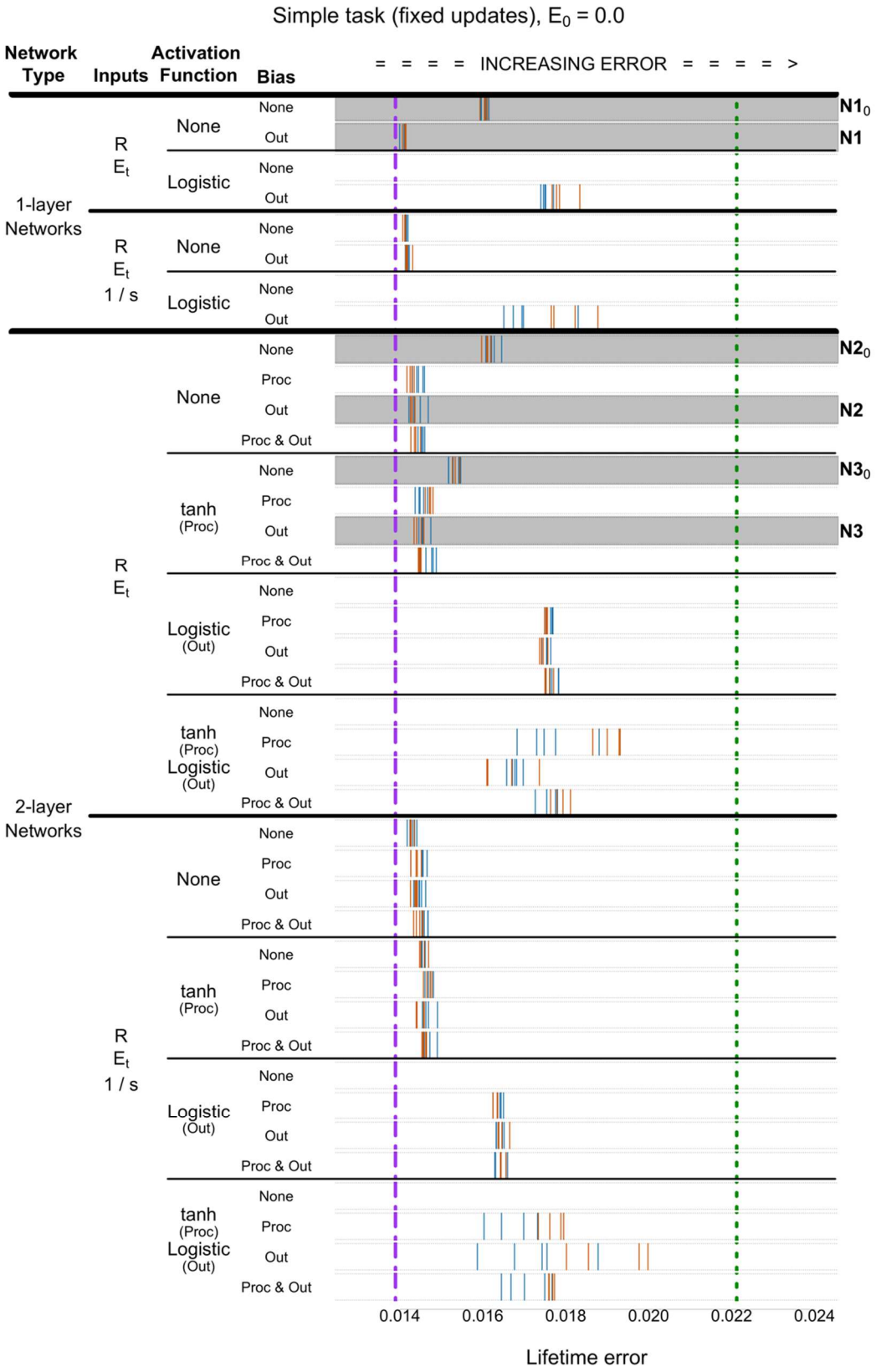


Figure A8. Performance of different networks for simple learning task (fixed number of updates *S*=10) and initial estimate E_0 = 0.0.

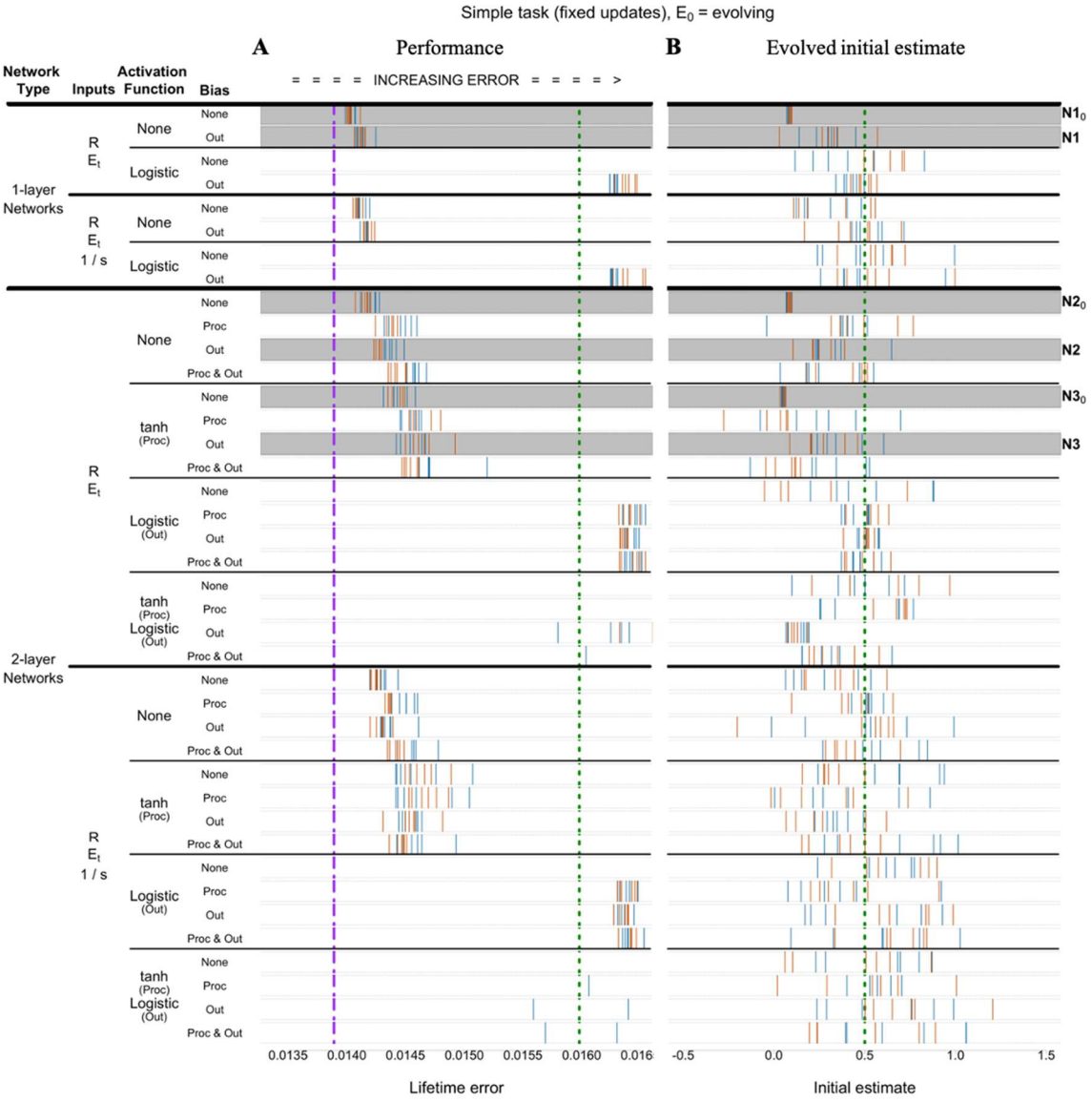


Figure A9. Performance (A) and the evolved initial estimate (B) of different networks for a simple learning task (a fixed number of updates *S*=10) and evolving initial estimate E_0.

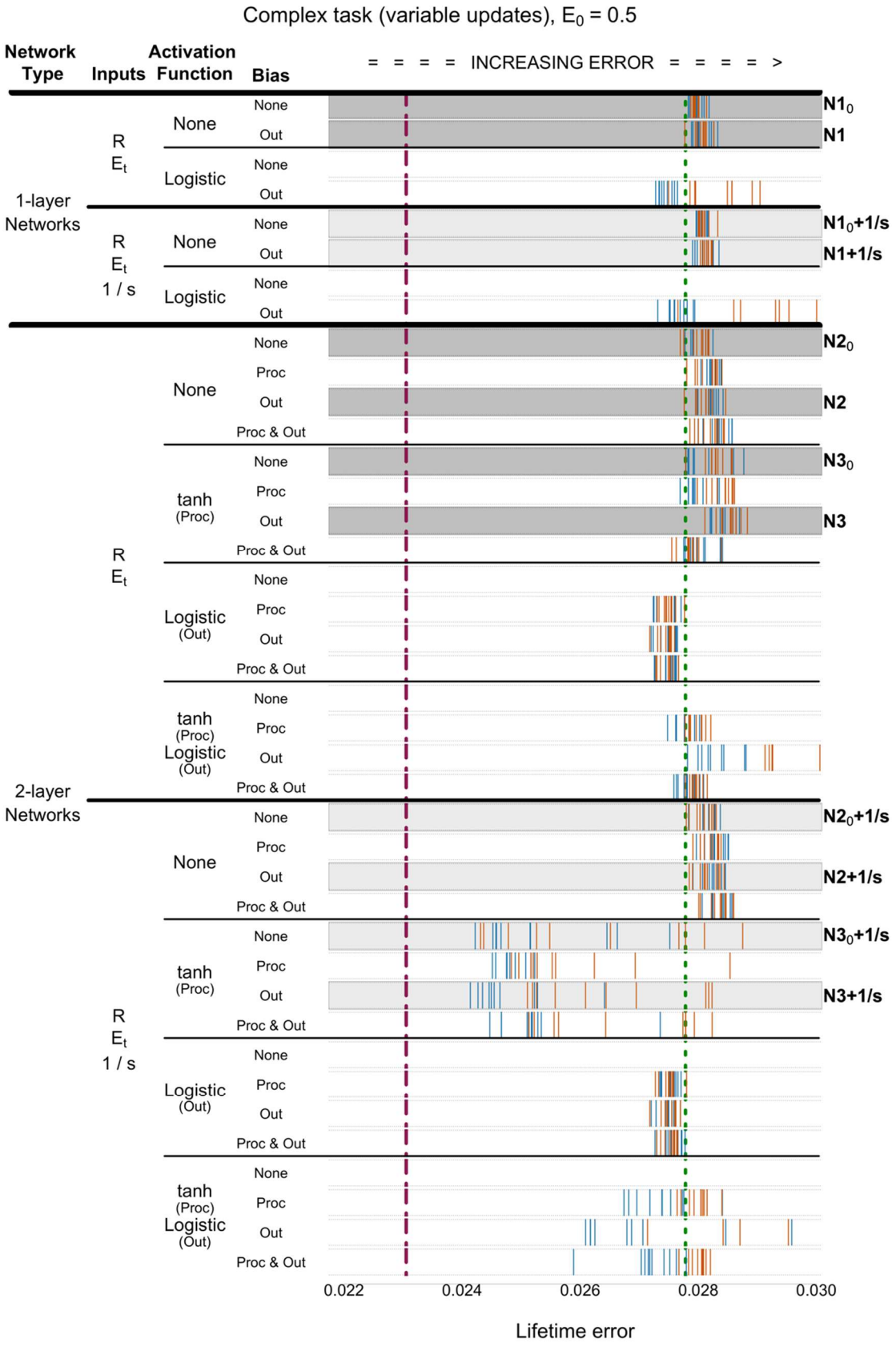


Figure A10 Performance of different networks for a more complicated task (a variable number of updates) and initial estimate $E_0 = 0.5$.

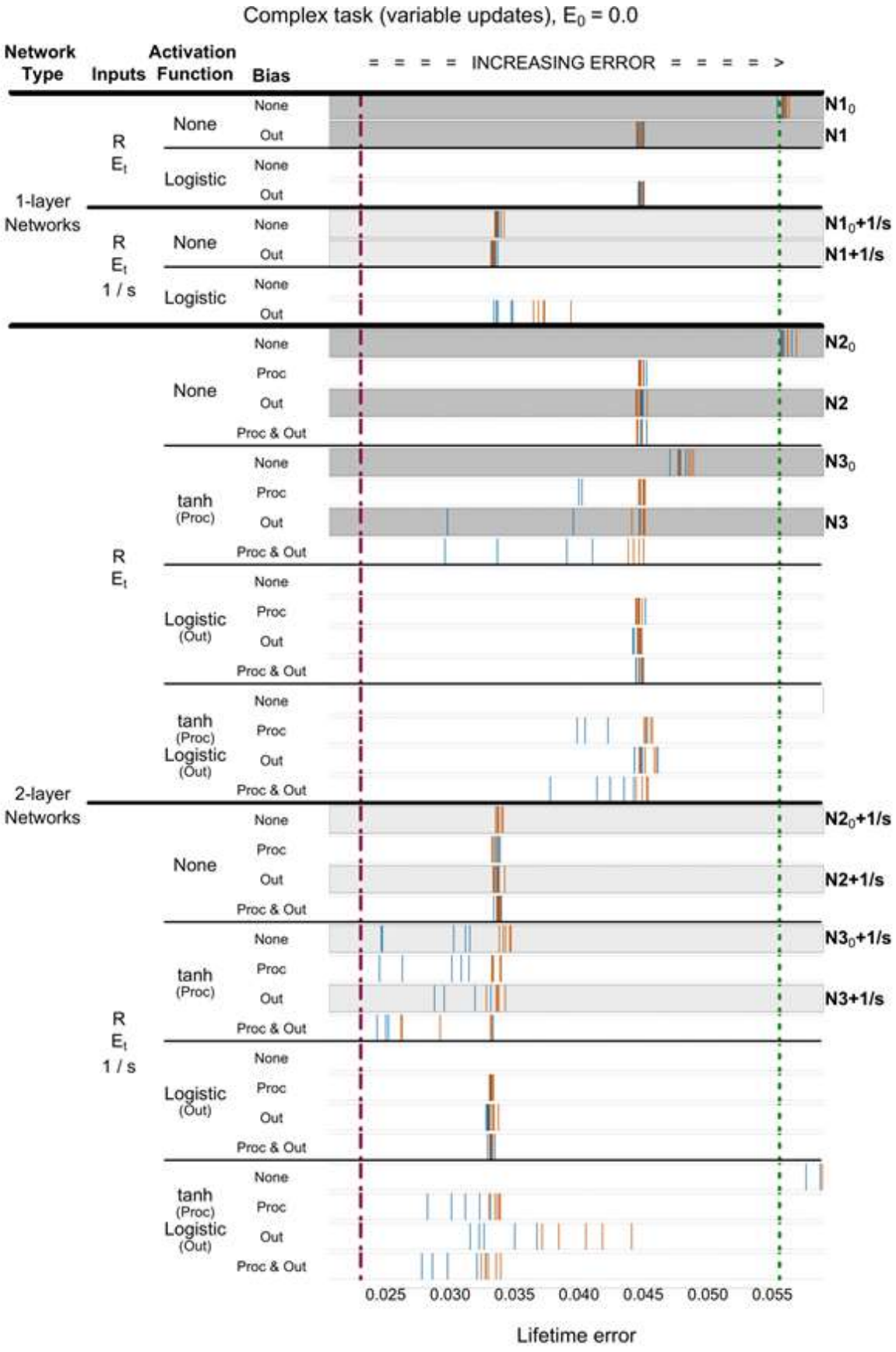


Figure A11. Performance of different networks for a more complicated task (a variable number of updates) and initial estimate E_0 = 0.0.

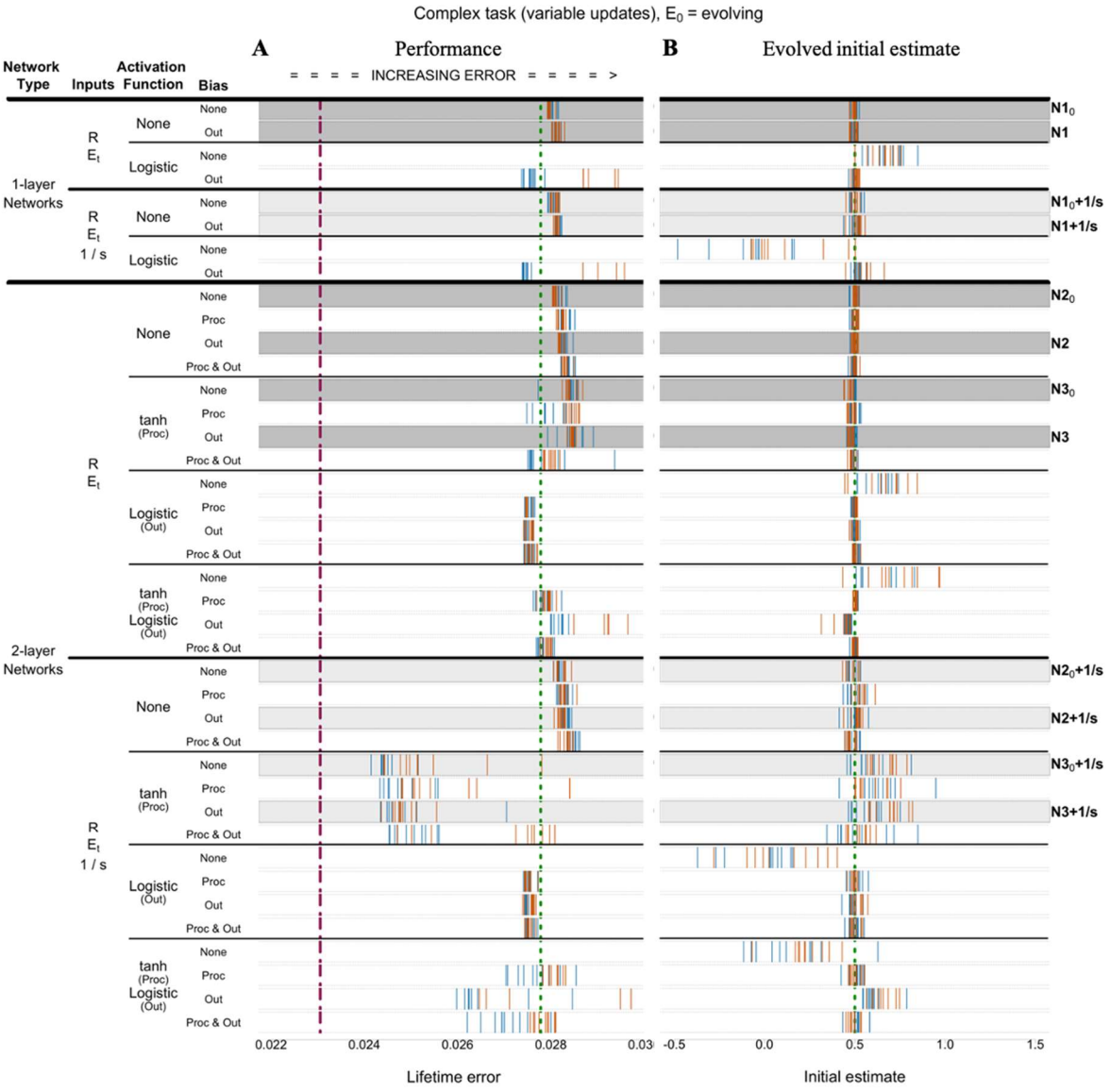


Figure A12. Performance (A) and the evolved initial estimate (B) of different networks for a more complicated learning task (a variable number of updates) and evolving initial estimate E_0.