Distributed Representations Accelerate Evolution of Adaptive Behaviours

James V. Stone*

Psychology Department, Sheffield University, Sheffield, United Kingdom

Animals with rudimentary innate abilities require substantial learning to transform those abilities into useful skills, where a skill can be considered as a set of sensory–motor associations. Using linear neural network models, it is proved that if skills are stored as distributed representations, then within-lifetime learning of part of a skill can induce automatic learning of the remaining parts of that skill. More importantly, it is shown that this "free-lunch" learning (FLL) is responsible for accelerated evolution of skills, when compared with networks which either 1) cannot benefit from FLL or 2) cannot learn. Specifically, it is shown that FLL accelerates the appearance of adaptive behaviour, both in its innate form and as FLL-induced behaviour, and that FLL can accelerate the rate at which learned behaviours become innate.

Citation: Stone JV (2007) Distributed representations accelerate evolution of adaptive behaviours. PLoS Comput Biol 3(7): e147. doi:10.1371/journal.pcbi.0030147

Introduction

Both evolution and learning may be considered as different types of adaptation. Learning occurs within a lifetime, whereas genetic change occurs across lifetimes [1]. Whereas genetic change ensures that a task can be executed innately, learning permits even the most rudimentary innate ability to be honed into a useful skill.

In an environment that fluctuates from generation to generation, learning permits an innate ability to be adapted to the particular physical environment into which each generation is born. If the environment ceases to fluctuate, then genetic assimilation [2] can transform a rudimentary innate ability, which requires much learning, into an innate skill, which requires minimal learning. This transformation is more likely to occur if the cost of learning is high [3,4], and, in this case, computer simulations suggest that learning can accelerate the rate of genetic assimilation [5] via the Baldwin effect [6]. However, if learning is sufficiently inexpensive, then genetic change may not occur at all [7,8]. Overall, there appears to be a tradeoff between learning and genetic assimilation, such that learning can subsidize genetic change, especially if learning is inexpensive.

All but the most primitive organisms learn in order to survive, and organisms which learn quickly are at a selective advantage relative to those that learn slowly. Therefore, a mechanism which reduces the time required to learn a given behaviour confers a selective advantage. One candidate for such a mechanism is FLL [9,10].

As explained below, FLL ensures that in the process of learning one set of associations or behaviours another set of associations is usually learned. These associations could comprise either perceptual skills (such as face recognition, predator recognition [11], or prey recognition), or motor skills (such as catching prey, flying, seed pecking, or nest building).

Free-Lunch Learning

Before considering how FLL can accelerate evolution of certain types of behaviours, FLL will be described in its original context of spontaneous recovery of memory in humans [9] and in neural network models [10]. Note that FLL is not unique to a specific class of network architectures, although it does assume that associations are learned using a form of supervised learning.

In humans, FLL has been demonstrated using a task in which participants learned the positions of letters on a nonstandard computer keyboard [9]. After a period of forgetting, participants relearned a proportion of these letter positions. Crucially, it was found that this relearning induced recovery of the non-relearned letter positions.

More recently, a set of theorems provided a formal characterization of FLL in linear neural network models [10]. In essence, FLL occurs in neural network models because each association is distributed amongst all connection weights (synapses) between units (model neurons). After partial forgetting, relearning some of the associations forces all of the weights closer to pre-forgetting values, resulting in improved performance even on non-relearned associations; a general proof is provided in [10]. A geometric demonstration of FLL for a network with two connection weights is given in Figure 1. Networks with multiple input and output units can be considered without loss of generality [10].

The protocol used to examine FLL in neural networks is as follows (see Figure 2). A network with n input units and one output unit has n connection weights. This network learns a set A of $m \leq n$ associations, where $A = A_1 \cup A_2$ comprises two subsets A_1 and A_2 of n_1 and n_2 associations, respectively (note that $m = n_1 + n_2$). After the associations A have been learned and then partially forgotten, performance error on subset A_1 is measured (forgetting is induced by adding isotropic noise

Editor: Karl J. Friston, University College London, United Kingdom

Received January 15, 2007; **Accepted** June 11, 2007; **Published** August 3, 2007

A previous version of this article appeared as an Early Online Release on June 11, 2007 (doi:10.1371/journal.pcbi.0030147.eor).

Abbreviations: FLL, free-lunch learning

* To whom correspondence should be addressed. E-mail: j.v.stone@shef.ac.uk

Author Summary

Some behaviours are purely innate (e.g., blinking), whereas other, "apparently innate," behaviours require a degree of learning to refine them into a useful skill (e.g., nest building). In terms of biological fitness, it matters how quickly such learning occurs, because time spent learning is time spent not eating, or time spent being eaten, both of which reduce fitness. Using artificial neural networks as model organisms, it is proven that it is possible for an organism to be born with a set of "primed" connections which guarantee that learning part of a skill induces automatic learning of other skill components, an effect known as free-lunch learning (FLL). Critically, this effect depends on the assumption that associations are stored as distributed representations. Using a genetic algorithm, it is shown that primed organisms can evolve within 30 generations. This has three important consequences. First, primed organisms learn quickly, which increases their fitness. Second, the presence of FLL effectively accelerates the rate of evolution, for both learned and innate skill components. Third, FLL can accelerate the rate at which learned behaviours become innate. These findings suggest that species may depend on the presence of distributed representations to ensure rapid evolution of adaptive behaviours.

to weights). Finally, *only* A_2 is relearned, and then performance error on A_1 is remeasured. FLL occurs if relearning A_2 improves performance on A_1. It has been proven that the probability of FLL approaches unity as the number of weights increases [10]. For the sake of brevity, this is reflected in phrases such as "learning A_2 *usually* improves performance on A_2" in this paper.

FLL and Evolution

Now consider an organism b_2 which is born with a genetically specified set of neuronal connections [12]. These connections are organised such that, if b_2 learns one subset A_2 of associations then another subset A_1 is usually learned. In other words, *the organism b_2 happens to be born with neuronal connections similar to the connections of an organism b_1 which had once learned and then forgotten subsets A_1 and A_2* (e.g., isotropically distributed around w_0 in Figure 1). Just as FLL ensures that if organism b_1 relearns A_2 then subset A_1 is usually relearned (see Figure 2), so *if b_2 learns A_2 then A_1 is usually learned*. In both cases, FLL ensures that learning one subset of associations induces learning of the other subset. Critically, whereas the FLL exhibited by organism b_1 depends on previous learning and forgetting, FLL in organism b_2 depends on being born in a state such that the first time A_2 is learned, the associations A_1 are also usually acquired. Such a network can be evolved using a genetic algorithm, as shown below.

The use of two distinct subsets in this paper is clearly unrealistic when considered in the context of skill learning. However, the use of two subsets lies at one extreme along a continuum of tasks. At one extreme, associations are learned one by one in a strict order, and at the other extreme, all associations are learned simultaneously. In a biological context, the components of a skill which are learned first act as "scaffold" for others, and this effectively imposes a temporal order to the acquisition of different skill components. This is the type of scenario assumed for the simulations reported in this paper. Essentially, learning A_2 is assumed to consist of a subset of skill components which provide a scaffold for the skill components in A_1.

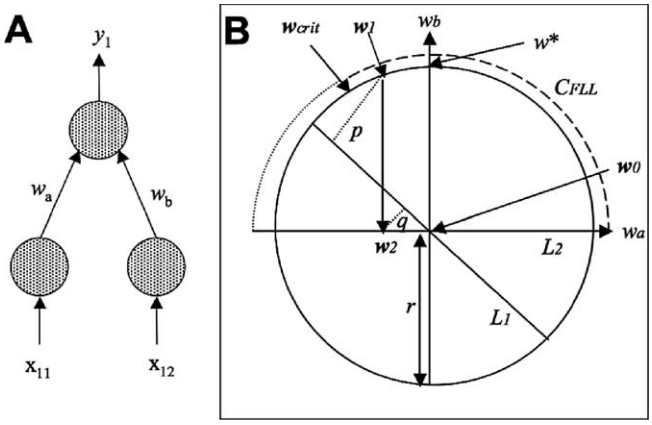

Figure 1. Geometry of Free-Lunch Learning

(A) Given a network with two input units and one output unit, its connection weights w_a and w_b define a weight vector $\mathbf{w} = (w_a, w_b)$. The network learns two subsets A_1 and A_2. In this example, each subset is a single association, where A_1 (for example) is the mapping from input vector $\mathbf{x}_1 = (x_{11}, x_{12})$ to desired output value d_1, and learning A_1 consists of adjusting $\mathbf{w}$ until the network output $y_1 = \mathbf{w} \cdot \mathbf{x}_1$ equals d_1.

(B) Each association A_1 and A_2 defines a constraint line L_1 and L_2, respectively. The intersection of L_1 and L_1 defines a point $\mathbf{w}_0$ which satisfies both constraints, so that zero performance error on $A = A_1 \cup A_2$ is obtained if $\mathbf{w} = \mathbf{w}_0$. After partial forgetting, the weight vector is a randomly chosen point $\mathbf{w}_1$ on the circle C of radius r, and the performance error on A_1 is given by the squared distance p^2. After relearning A_2, the weight vector $\mathbf{w}_2$ lies in L_2, and performance error on A_1 is the squared distance q^2. FLL occurs if performance error on A_1 is decreased by relearning A_2, or equivalently if $p^2 > q^2$. Relearning A_2 has three possible effects, depending on the position of $\mathbf{w}_1$ on C: 1) if $\mathbf{w}_1$ is under the larger (dashed) arc C_{FLL} (as shown here), then $p^2 > q^2$; therefore, FLL is observed, 2) if $\mathbf{w}_1$ is under the smaller (dotted) arc, then $p^2 < q^2$; therefore, negative FLL is observed, and 3) if $\mathbf{w}_1$ is at the critical point $\mathbf{w}_{crit}$, then $p^2 = q^2$; therefore, no FLL is observed. Given that $\mathbf{w}_1$ is a randomly chosen point on the circle C, the probability of FLL is equal to the proportion of the upper semicircle under the (dashed) arc C_{FLL}. Symmetry implies the above analysis also applies to the lower half of C. In terms of evolution, a network "born" with $\mathbf{w} = \mathbf{w}^*$ has zero error on both A_1 and A_2 after learning only A_2 (see text).

doi:10.1371/journal.pcbi.0030147.g001

The Geometry of FLL

This section is a brief account of the basic geometry underlying FLL, in the absence of its interactions with evolution. For the present, and without loss of generality (see [10]), we assume that the network has one output unit and two input units, which implies $n = 2$ connection weights, and that A_1 and A_2 each consist of $n_1 = n_2 = 1$ association, as in Figure 1. Input units are connected to the output unit via weights w_a and w_b, which define a weight vector $\mathbf{w} = (w_a, w_b)$. Associations A_1 and A_2 consist of different mappings from the input vectors $\mathbf{x}_1 = (x_{11}, x_{12})$ and $\mathbf{x}_2 = (x_{21}, x_{22})$ to desired output values d_1 and d_2, respectively. If a network is presented with input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, then its output values are $y_1 = \mathbf{w} \cdot \mathbf{x}_1 = w_a x_{11} + w_b x_{12}$ and $y_2 = \mathbf{w} \cdot \mathbf{x}_2 = w_a x_{21} + w_b x_{22}$, respectively. More generally, network performance error for k associations is defined as

$$E(\mathbf{w}, A) = \sum_{i=1}^{k} (d_i - y_i)^2. \quad (1)$$

The weight vector $\mathbf{w}$ defines a point in the (w_a, w_b) plane. For an input vector $\mathbf{x}_1$, there are many different combinations of weight values w_a and w_b that give the desired output d_1. These combinations lie on a straight line L_1, because the network

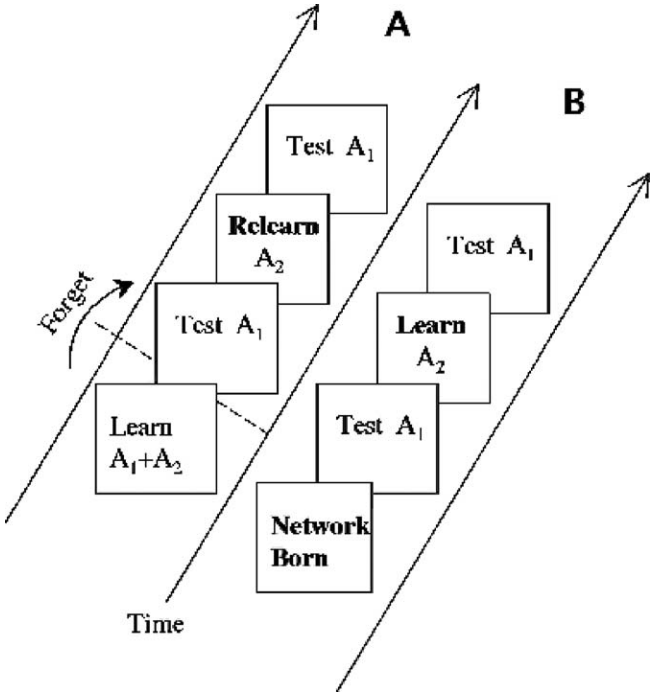

Figure 2. Free-Lunch Learning within and across Generations

(A) FLL within the single lifetime of a neural network model. Two subsets of associations A_1 and A_2 are learned. After partial forgetting (see text), performance error on subset A_1 is measured. Subset A_2 is then relearned, and performance error on subset A_1 is measured again. If performance error on A_1 decreases as a result of learning A_2, then FLL has occurred. (B) FLL across generations. Using a genetic algorithm, a network is born with connections similar to those of the network in **a** after it has learned and then partially forgotten subsets A_1 and A_2. Consequently, innate performance error on A_1 is similar to that of the network in **a** after it has partially forgotten both subsets. After measuring performance error on A_1 at birth, subset A_2 is learned for the first time, and performance error on subset A_1 is measured again. After many generations, the innate connection values of each network ensure that if subset A_2 is learned *for the first time,* then this induces automatic learning of subset A_1.

doi:10.1371/journal.pcbi.0030147.g002

output is a linear weighted sum of input values. A corresponding constraint line L_2 exists for A_2. The intersection of L_1 and L_2 therefore defines the only point $\mathbf{w}_0$ that satisfies both constraints, so that zero error on A_1 and A_2 is obtained if and only if $\mathbf{w} = \mathbf{w}_0$. Without loss of generality, we define the origin $\mathbf{w}_0$ to be the intersection of L_1 and L_2. A general prerequisite for FLL is that L_1 is not orthogonal to L_2.

We now consider the geometric effect of partial forgetting of both associations, followed by relearning A_2. This geometric account applies to a network with two weights (see Figure 1), and depends on the following observation: if the length of the input vector $\mathbf{x}_1$ is unity, then the performance error $E(\mathbf{w},A_1) = (d_1 - y_1)^2$ of a network with weight vector $\mathbf{w}$ when tested on association A_1 is equal to the squared distance between $\mathbf{w}$ and the constraint line L_1 [10]. For example, if $\mathbf{w}$ is in L_1, then $E(\mathbf{w},A_1) = 0$, but as the distance between $\mathbf{w}$ and L_1 increases so $E(\mathbf{w},A_1)$ must increase. For the purposes of this geometric account, we assume the length of the input vectors is unity.

If partial forgetting is induced by adding isotropic noise $\mathbf{g}$ to the weight vector $\mathbf{w} = \mathbf{w}_0$, then this effectively moves $\mathbf{w}$ to a randomly chosen point $\mathbf{w}_1 = \mathbf{w}_0 + \mathbf{g}$ on the circle C of radius r, where r is the length of $\mathbf{g}$, and represents the amount of forgetting. For a network with $\mathbf{w} = \mathbf{w}_1$, learning A_2 moves $\mathbf{w}$ to the nearest point $\mathbf{w}_2$ on L_1 [10], so that $\mathbf{w}_2$ is the orthogonal projection of $\mathbf{w}$ on L_2. Before relearning A_2, the performance error $E(\mathbf{w},A_1)$ on A_1 is the squared distance p^2 between $\mathbf{w}_1$ and its orthogonal projection on L_1. After relearning A_2, the performance error E_{post} is the squared distance q^2 between $\mathbf{w}_2$ and its orthogonal projection on L_1. The amount of FLL is $\delta = E(\mathbf{w}_1, A_1) - E(\mathbf{w}_2, A_1)$, and (for a network with two weights) is also given by $Q = p^2 - q^2$. The probability $P(\delta > 0)$ of FLL given L_1 and L_2 is equal to the proportion of points on C for which $\delta > 0$ (or equivalently, for which $Q > 0$). For example, it can be shown that the mean value of this proportion is $P(\delta > 0)$ for a two-weight network like the one shown in Figure 1A. Given the particular configuration shown in Figure 1A, the critical point $\mathbf{w}_{crit}$ is defined such that the performance error before and after learning is the same (i.e., $\delta = 0$).

FLL Induces Perfect Performance

Given a network with n weights $\mathbf{w}$ and two subsets A_1 and A_2 of n_1 and n_2 associations, respectively, it is shown that weights $\mathbf{w}^*$ exist such that learning associations A_2 is guaranteed to yield zero performance error on A_1, provided $n \geq n_1 + n_2$.

Consider a network with $n = 2$ weights and subsets A_1 and A_2, each of which comprises a single association. Each association defines a constraint line L_1 and L_2, respectively (see Figure 1). If the weight vector $\mathbf{w}$ is in L_1, then performance error on A_1 is zero, and if $\mathbf{w}$ is in L_2, then performance error on A_2 is zero. Clearly, if and only if $\mathbf{w}$ is at the intersection $\mathbf{w}_0$ of L_1 and L_2, then performance error on both A_1 and A_2 is zero. If $\mathbf{w}$ is not in L_2, then learning A_2 moves $\mathbf{w}$ from its current position to its orthogonal projection onto L_2 [10]. Crucially, if $\mathbf{w} = \mathbf{w}^*$ in Figure 1, then learning A_2 moves $\mathbf{w}$ to the optimal weight vector $\mathbf{w}_0$. In this case, learning A_2 reduces performance error to zero on both A_1 and A_2, and therefore learning A_2 implies perfect performance on A_1.

This line of reasoning generalises to networks with more than two weights, as follows. If a network has more than two input units, then subsets A_1 and A_2 can have $n_1 > 1$ and $n_2 > 1$ associations. If $n \geq n_1 + n_2$, then A_1 and A_2 define an $(n - n_1)$-dimensional subspace L_1 and an $(n - n_2)$-dimensional subspace L_2, respectively. The intersection L_{12} of L_1 and L_2 corresponds to weight vectors which generate zero error on $A = A_1 \cup A_2$. In this case, the circle in Figure 1 corresponds to an n-dimensional hypersphere, with its centre $\mathbf{w}_0$ in L_{12}. Given that learning A_2 provides an orthogonal projection of $\mathbf{w}$ onto L_2, and that there exists a $\mathbf{w} = \mathbf{w}^*$ such that its orthogonal projection onto L_2 is $\mathbf{w}_0$, it follows that learning A_2 in a network with $\mathbf{w} = \mathbf{w}^*$ yields zero performance error on both A_2 and A_1.

Given that the weight vector $\mathbf{w}$ is genetically specified with finite precision, a network is necessarily born with its weight vector $\mathbf{w} = \mathbf{w}_1$ at a non-zero distance r from the optimal weight vector $\mathbf{w}_0$. This finite precision defines a hypersphere C around $\mathbf{w}_0$, and the location of $\mathbf{w}_1$ on C determines the amount of FLL. If a network is born with $\mathbf{w}_1 = \mathbf{w}^*$, then learning A_2 induces perfect performance on A_1. If fitness depends on performance on both A_1 and A_2 after learning only A_2, then there is selective pressure for networks to be born with weight vectors close to $\mathbf{w}^*$, given a specific degree of genetic precision r. More generally, there is pressure for

networks to be born with weight vectors close to the subspace with contains $\mathbf{w}_0$ and $\mathbf{w}^*$.

Terminology: Evolution, Innateness, and FLL

As this paper deals with subtle combinations of evolution and learning, involving two distinct subsets of associations (A_1 and A_2), it is important to be clear about terminology. Specifically, we need to be careful about which subset is being referred to, and whether we are referring to innate performance or not. Accordingly, performance error on A_1 before learning A_2 is called just that, "innate performance error on A_1." Behaviours that are induced by learning A_2 are called *FLL-induced* behaviours, because they are not innate, nor are they learned, so that performance error on A_1 after learning A_2 is called "FLL-induced performance error on A_1." If learning A_2 does not affect performance on A_1 (as in condition NoFLL, below), then this is referred to as "post-learning performance." More generally, performance will be quoted with a specified context (e.g., performance on A_1 after learning A_2).

Methods

The effect of FLL on evolution was tested by measuring performance on A_1 after learning A_2 across generations. To eliminate the possibility that the observed results are artefacts, the effects of FLL were compared with two control conditions (described below).

Each generation consisted of 1,000 neural networks, each of which consisted of 20 input units and one output unit. The genome of each network was defined by a one-to-one mapping of the $n = 20$ weight values in the network to a single string of n genes, where the value of each gene was set to the value of a corresponding network weight. The number of offspring generated by each network was proportional to its fitness, which depended only on its ability to provide the correct desired output value for each of 20, n-element input vectors. The mapping from each input vector to its output value defines one association (see Figure 1).

A network's output y_i is a weighted sum of input values $y_i = \mathbf{w} \cdot \mathbf{x}_i = \Sigma_{j=1}^{n_1+n_2} w_j x_{ij}$, where x_{ij} is the jth value of the ith input vector $\mathbf{x}_i$, and each weight w_i is one input–output connection.

The fitness of each network was assessed with respect to its performance error on a single common set $A = A_1 \cup A_2$ of $m = 20$ associations, where A_1 and A_2 are two disjoint subsets of $n_1 = 10$ and $n_2 = 10$ associations, respectively. The m associations in A were allocated randomly to the two subsets, A_1 and A_2. The subsets A_1 and A_2 were intended to represent different components of a task, and were therefore the same *for all networks,* and *across all generations*. In the first generation, each network's weight values were chosen from a Gaussian distribution (see below for details).

The desired output value d_i for each input vector $\mathbf{x}_i$ was drawn from a Gaussian distribution with variance $1/n$. An analytic method was used to solve for the optimal weight vector $\mathbf{w}_0$ which maps inputs to outputs: $d_i = \mathbf{w}_0 \cdot \mathbf{x}_i$ (see below). Given the variances of the inputs and outputs, the expected length of $\mathbf{w}_0$ is unity.

Each new generation was formed from 1,000 matings between 1,000 pairs of networks. The K^{th} network was chosen for mating according to its fitness $F(K)$ with probability $p(K)$. Networks were chosen with replacement to ensure that the number of offspring from a given network was proportional to $p(K)$, which is defined as

$$p(K) = \frac{F(K)}{\sum_{k}^{1000} F(k)}, \tag{2}$$

where the denominator ensures $\sum_k p(k) = 1$. Half the weights of each offspring were copied from (randomly chosen) corresponding weight locations in one parent network, and half from the other parent. Aside from mutations, weight values inherited by an offspring were the same as those inherited by its parents (i.e., inheritance was Darwinian, not Lamarckian).

Mutation was applied to each weight with a probability of 0.05, using a uniform probability density function. Then Gaussian noise with a standard deviation of 0.05 was added to the value of those weights that had been chosen for mutation.

There were three conditions: *FLL, NoFLL,* and *NoLearn*, with corresponding fitness functions F_{FLL}, F_{NoFLL}, and $F_{NoLearn}$. The initial randomly chosen weight values (see Network Learning Algorithm) of the population of networks were the same in all conditions. Networks were selected for mating according to their performance on the combined set $A = A_1 \cup A_2$, according to Equation 2 for all fitness functions, as described next.

Condition FLL

Networks that exhibited high levels of FLL were preferentially selected for mating. Only associations A_2 were learned, but the fitness of each network depended on its performance on both the learned associations A_2 and on the unlearned associations A_1. The fitness $F_{FLL}(K)$ of the K^{th} network is defined in terms of its innate performance error E_{pre} on $A = A_1 \cup A_2$, and on its performance error E_{post} on A after learning A_2:

$$F_{FLL}(K) = \frac{1}{cE_{pre} + (1-c)E_{post}}, \tag{3}$$

where E_{pre} and E_{post} are:

$$E_{pre} = \sum_{i}^{n_1+n_2} D_i^{pre} \tag{4}$$

$$E_{post} = \sum_{i}^{n_1+n_2} D_i^{post} \approx \sum_{i}^{n_1} D_i^{post}, \tag{5}$$

where D_i^{pre} and D_i^{post} are the network's output errors in response to the ith input vector before and after learning A_2 (respectively). The parameter $c = 0.05$ defines the balance between performance error on innate versus post-learning (e.g., FLL-induced) behaviours, and is interpreted as a cost-of-learning parameter (see below). The network's fitness error D_i is a function of the difference $e_i = y_i - d_i$ between the network's response y_i to the ith input vector and the desired output value d_i:

$$D_i = \begin{cases} e_i^2 \text{ if } e_i^2 \le D_{thresh}, \\ 1 \text{ if } e_i^2 > D_{thresh}. \end{cases} \tag{6}$$

This ensures that output errors above D_{thresh} have a disproportionately large and detrimental effect on fitness,

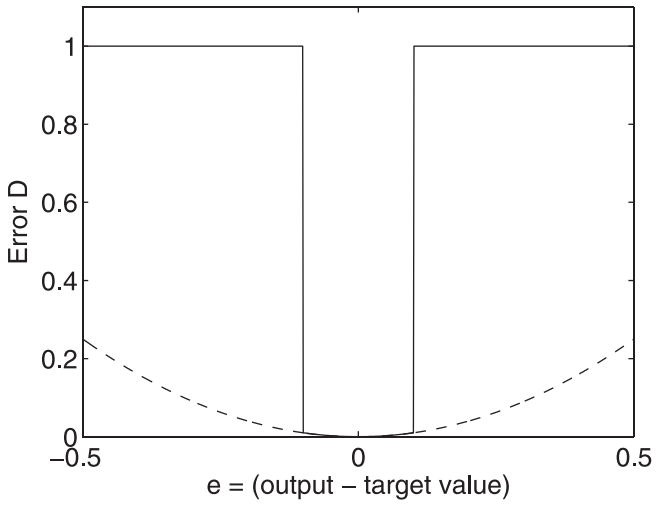

Figure 3. Network Fitness Error Function

The response of the network to a given input vector **x** is $y = \mathbf{w} \cdot \mathbf{x}$. Given a desired (target) output *d*, the solid curve shows how the fitness penalty *D* for an incorrect response increases sharply (to unity) if the magnitude of the difference $e = y - d$ is greater than 0.1 (i.e., if $e^2 > 0.01$). For comparison, the quadratic error function $e^2 = (y - d)^2$, which is minimised during learning, is shown as a dashed curve. The range of *e*-values shown are typical for the simulations reported here. See Methods section for details.

doi:10.1371/journal.pcbi.0030147.g003

as shown in Figure 3. This, in turn, ensures that only those networks with "good" performance are likely to be selected for reproduction. The value of D_{thresh} was set to 0.01.

For later use, we also define the fitness errors $E_{pre} = E_{pre}(A_1) + E_{pre}(A_2)$, and $E_{post} = E_{post}(A_1) + E_{post}(A_2) \approx E_{post}(A_1)$. The approximation here and in Equation 5 emphasises the fact that the total fitness error is attributable almost exclusively to A_1 after learning A_2 (because error on A_2 is then almost zero).

The inclusion of innate performance error E_{pre} in F_{FLL} ensures that the cost of learning is taken into account when assessing fitness. If c is small, then E_{pre} tends to be large, so that much learning is required to increase fitness. Conversely, if c is large, then E_{pre} tends to be small, so that little learning is required to increase fitness. Thus, E_{pre} is an implicit measure of the cost of learning (where c multiplies E_{pre}), and ensures that networks which require minimal learning have high innate fitness (although the small value of c used in Equation 3 defines a relatively low cost of learning).

Condition NoFLL

This was identical to condition FLL, except that the effects of FLL were precluded by making all input vectors in A mutually orthogonal, whilst retaining the length of each vector as in condition FLL, using Gram-Schmidt orthogonalisation. This makes the input vectors in A_1 orthogonal to those in A_2, which ensures L_1 and L_2 are orthogonal. This, in turn, ensures that learning A_2 cannot affect performance on A_1. The fitness function F_{NoFLL} was the same as in condition FLL (i.e., $F_{NoFLL} = F_{FLL}$).

Condition NoLearn

No learning occurred, so that improvement in performance over successive generations was due only to selection of innate performance on A_1 and A_2. Fitness was defined as $F_{NoLearn} = 1/E_{pre}$, where E_{pre} is defined in Equation 4. This is equivalent to setting $c = 1$ in Equation 3.

Network Learning Algorithm

The network learning algorithm used here involves a type of supervised learning. Note that Equation 1 defines the network error used for learning, whereas Equation 3 defines the fitness of a network.

Each network was initialised with n weight values drawn randomly from a Gaussian distribution with unit variance. This was then divided by $n^{1/2}$, which ensures that the expected length of weight vectors in the population is unity.

Given a network with n input units and one output unit, the set A of m, n-element input vectors $\mathbf{x}_i$: $i = 1 \ldots m$ and m desired scalar output target values d_i were chosen randomly from a Gaussian distribution with unit variance. Each input vector was then divided by $n^{1/2}$ so that the expected length of input vectors was unity (i.e., the variance of input values was $1/n$).

In conditions FLL and NoFLL, each network learned n_2 associations. Rather than using the iterative weight update normally associated with the delta rule, an analytic solution was obtained. Learning n_2 associations consists of finding the orthogonal projection operator which projects the initial weight vector $\mathbf{w}_1$ to its nearest point in the subspace (e.g., L_2) defined by the n_2 input vectors being learned. The end result $\mathbf{w}_2$ is the same as that obtained using the standard delta rule for infinitesimal learning rates [10]. As with the standard delta rule, this yielded a value of approximately zero for post-learning performance error on the learned associations A_1. This type of learning is most plausibly associated with motor learning in the cerebellum and basal ganglia [10].

Results

The results are based on ten computer simulation runs for each of the three conditions, FLL, NoFLL, and NoLearn, described above, and graphs show the mean of these ten runs. Each run involved a different fixed set A of 20 associations. As a reminder, the two free parameters are: 1) the cost-of-learning parameter, which was set to $c = 0.05$, and 2) the threshold of the fitness error function, which was set to $D_{\text{thresh}} = 0.01$.

The main results are shown in Figure 4, which is a summary of more detailed results in Figure 5. Condition FLL yields a FLL-induced error (i.e., error on A_1 after learning A_2) of approximately zero after 30 generations, whereas condition NoFLL requires about 60 generations to achieve an error of less than unity. Condition NoLearn (dotted curve) yields the slowest innate learning, and is included for comparison.

The proportion of networks that exhibit FLL over generations is shown in Figure 6A, and the amount of FLL is shown in Figure 6B. The proportion and amount of FLL increases in condition FLL, as indicated by the solid line in each figure. The zero prevalence of FLL in condition NoFLL (dashed line) is associated with zero FLL as indicated in Figure 6B. More detailed results are shown in Figure 5A–5C.

Performance on A_1

Performance on A_1 (solid line in Figure 5A) after learning A_2 is better in condition FLL than in condition NoFLL (solid line, 5B). Innate performance on A_1 is also better in condition FLL (dashed line, Figure 5A) than in conditions NoFLL

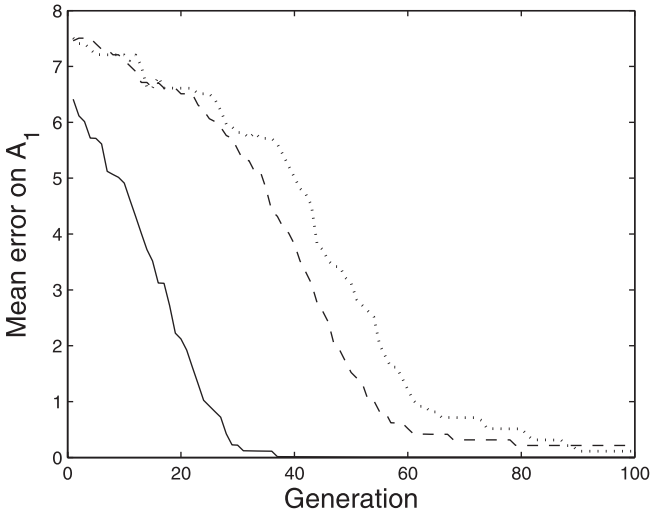

Figure 4. Effect of Free-Lunch Learning on Performance Error

The graph shows the mean results of ten computer simulation runs, in three conditions: FLL, NoFLL, and NoLearn (see text). In each run, the fitness of each of 1,000 networks was determined by performance error on a fixed set $A = A_1 \cup A_2$ of 20 associations; a different set A was used for each run. For all graphs in this paper, the median (of error, here) of 1,000 networks was obtained throughout each run, and the mean of ten medians is shown for each generation in each condition (standard errors were no greater than 0.5, and are not shown for clarity).

Condition FLL (solid line): mean performance error $E_{post}(A_1)$ on the ten associations in A_1 after learning the ten associations in subset A_2. Learning A_2 had a beneficial effect on performance on A_1 over 100 generations, corresponding to an increase in the amount and prevalence of FLL (see Figure 6). **Condition NoFLL** (dashed line): mean performance error $E_{post}(A_1)$ was evaluated as in condition FLL, except that the input vectors in A_1 were orthogonal to those in A_2, so that learning A_2 could not have any effect on performance on A_1 (see text). **Condition NoLearn** (dotted line): mean performance error E_{pre} on A_1 was evaluated "at birth" (i.e., no within-lifetime learning was allowed).

doi:10.1371/journal.pcbi.0030147.g004

(dashed line, Figure 5B) and NoLearn (dashed line, Figure 5C). Together, these results suggest that FLL accelerates both the rate at which FLL-induced behaviours (A_1) appear, as well as the rate at which FLL-induced behaviours (A_1) become genetically assimilated.

Performance on A_2

Innate performance on subset A_2 is better in condition FLL (dotted curve, Figure 5A) than in conditions NoFLL (dotted curve, Figure 5B) and NoLearn (dotted curve, Figure 5C).

Learning A_2 reduces error on subset A_1 even in the first generation in conditions FLL (Figure 5A). Additionally, the proportion of networks showing FLL is greater than 0.5 (Figure 6A), and the amount of FLL is greater than zero (Figure 6B) in the first generation. These effects are not due to any special properties of the networks nor of the associations. Indeed they are entirely expected, and are consistent with the theoretical analysis in [10]. In essence, FLL is observed in the first generation because it is very unlikely that the mainfolds L_1 and L_2 defined by A_1 and A_2 (respectively) are orthogonal, so that learning A_2 usually reduces error on A_1 (albeit by a small amount in the first generation).

Discussion

Before discussing results in detail, it is important to clarify precisely what is being claimed here. The main claim is that, given a population of organisms which can learn, the presence of FLL accelerates the rate at which a given set of advantageous behaviours evolves relative to populations which 1) can learn but which do not have FLL (e.g., condition NoFLL) and 2) cannot learn (e.g., condition NoLearn). Specifically, it is claimed that FLL accelerates the appearance of adaptive behaviour, both in its innate form and as FLL-induced behaviour, and that FLL can accelerate the rate at which learned behaviours become innate.

It is also claimed that FLL increases the rate at which a set of behaviours (e.g., A) is acquired within a lifetime. Clearly, if learning one subset (e.g., A_2) induces learning of another subset (e.g., A_1), then the amount of learning required to learn both subsets (e.g., A) is reduced.

It is worth noting that FLL is not related to generalisation (see [10]), which cannot therefore be responsible for the effects reported here.

Task Difficulty

The task was purposely made difficult, such that network outputs which were not close to desired target values were assigned an error value of unity. This heavily penalises networks that do not generate near-correct responses. This type of task may emulate tasks for which being "almost correct" provides no fitness benefit. Such tasks are exemplified by a predator which almost catches prey (e.g., a kingfisher almost catching a fish, or where each failed attempt yields a large fitness cost), or where learning is incremental and stepwise (e.g., learning to catch progressively larger prey). Such tasks give rise to "needle-in-a-haystack" search spaces [5], which have rugged or uncorrelated landscapes [13].

Is Accelerated Evolution due to Learning?

A cogent critique of research by Nolfi et al. [14] argues that accelerated evolution (specifically, assimilation) is a generic consequence of learning per se [15]. In results not shown here, replacing A_2 with a new, randomly chosen subset every generation in condition FLL yields a more gradual evolution of FLL-induced and innate behaviours than is obtained in any of the conditions used here. This effectively excludes the possibility that the accelerated evolution reported here is due to learning per se.

Reaction Norms

In terms of evolutionary theory, FLL-induced behaviours can be considered as the establishment of a new reaction norm. The specific "environment" that induces the reaction norm is learning a particular subset of behaviours (A_2), and the phenotypic reaction to this environment is another subset of behaviours (A_1).

Genetic Assimilation

FLL does not necessarily force FLL-induced behaviours to become genetically assimilated. In fact, there is a tradeoff between the amount of acceleration induced by FLL and the extent to which behaviours become innate. If the cost-of-learning parameter is set to $c = 1$, then there is no incentive for FLL to increase over generations. In contrast, if $c \approx 0$ (as in the simulations reported here), then the rapid evolution of FLL-induced behaviour shown in Figure 5A (solid line) is obtained, alongside the slower evolution of innate behaviour (dashed line in Figure 5A). Thus, even the small value of c (0.05) used here puts pressure on learned behaviours to

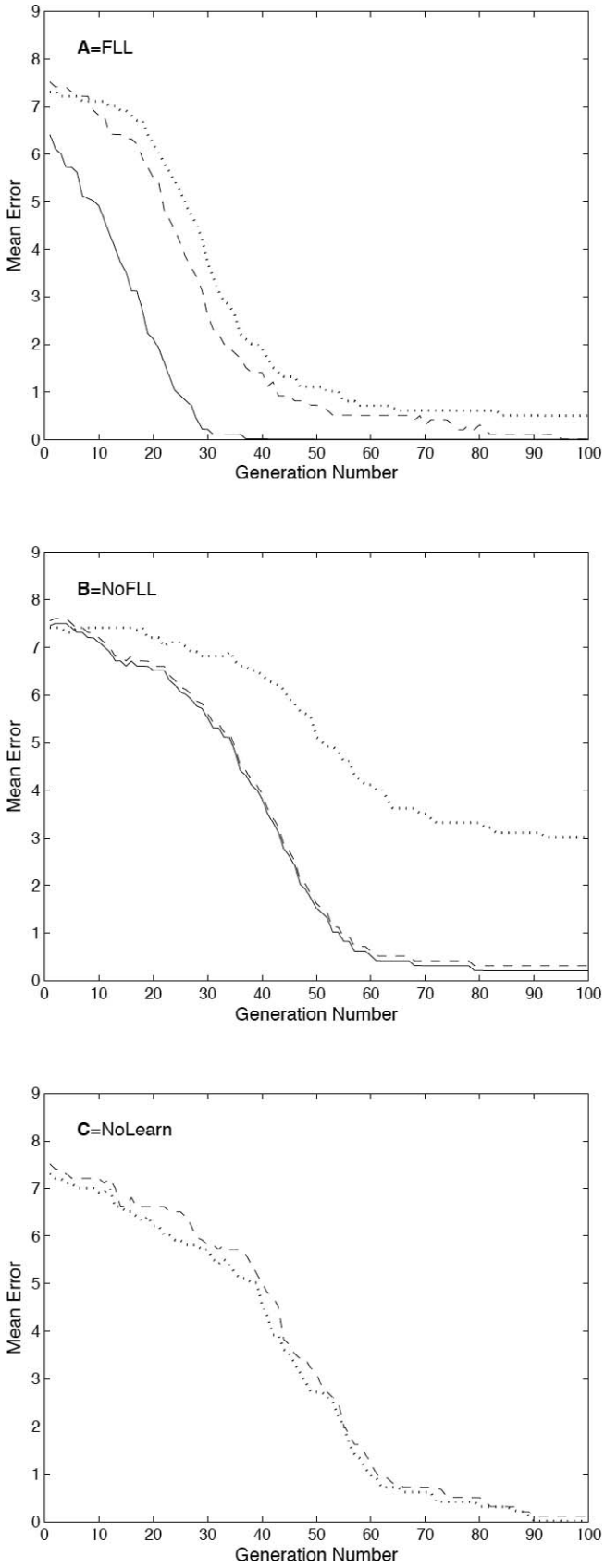

Figure 5. Effect of Free-Lunch Learning on Evolution

Performance error in three conditions (FLL, NoFLL, and NoLearn). Different performance errors are drawn as follows: **solid line**, $E_{post}(A_1)$, performance error on A_1 after learning only A_2; **dashed line**, $E_{pre}(A_1)$, innate performance error on A_1; **dotted line**, $E_{pre}(A_2)$, innate performance error on A_2.

(A) **Condition FLL**: mean performance error $E_{post}(A_1)$ on A_1 (solid line) after learning A_2 decreased over generations most rapidly in this condition. Innate error on A_1 is slightly lower than on A_2.

(B) **Condition NoFLL**: precluding FLL ensured that mean innate error on A_1 (dashed line) was essentially the same as after learning A_2 (solid line). The dashed line has been plotted 0.1 units above the solid line for clarity. Innate error on A_2 is large because the fitness cost of innate errors is low ($c = 0.05$).

(C) **Condition NoLearn**: mean innate performance error $E_{pre}(A_1)$ and $E_{pre}(A_2)$ on A_1 and A_2. For comparison, mean performance errors on A_1 in each condition (i.e., the solid lines here) are summarised in Figure 4.

doi:10.1371/journal.pcbi.0030147.g005

become innate, as indicated by the decreasing innate performance errors on A_1 (dashed line) and A_2 (dotted line) in Figure 5A.

In practice, learning always has a non-zero fitness cost, if only in terms of the time required for that learning to occur. This is because time spent learning is time spent not eating, or time spent being eaten, both of which reduce fitness. Thus, the small value of c used here represents one value along the spectrum of learning costs. It therefore seems likely that even the simplest learned behaviours have a tendency to become innate, and that this tendency increases with the cost of learning. For example, in results not shown here, increasing the cost-of-learning parameter c decreases the rate at which FLL-induced performance on A_1 improves, and increases the rate at which performance on A_1 and A_2 becomes innate (innate performance with $c = 1$ is effectively obtained in condition NoLearn (see Figure 5C)). It is therefore not easy to classify the effect reported here as a clear-cut example of the Baldwin effect [6], although these effects are almost certainly related.

General Free-Lunch Effects

The basic geometry which underpins FLL within a lifetime (as in [10]) and across lifetimes (as here) can also be applied in two other contexts: 1) evolution of innate behaviours without learning, and 2) evolution of general phenotypic traits. These two cases are considered in the next two paragraphs. Both of these effects require the presence of environmental conditions that fluctuate over successive generations (e.g., fluctuations in temperature induced by ice ages, salinity, prey numbers, or predation pressure).

1) Accelerated evolution of innate behaviours without learning can be understood by considering an organism that has no learning ability, and which relies on genetic specification of its neuronal connections [12]. Natural selection ensures that its neuronal connections at birth yield innate behaviour matched to its environment. If the environment changes, then natural selection will induce a corresponding shift to a new set of innate connections. If the environment then shifts back to its original state, then organisms' connections will tend to revert to their original values. Let us assume that some connections revert faster than others over successive generations. For example, some connections may be specified by genes linked to other innate behaviours, and this genetic linkage would tend to reduce the rate of genetic change. In fact, for simplicity, assume that half of the connections revert quickly and half revert slowly. If the required behaviours are encoded as distributed representations, then this connection reversion will induce a FLL-type

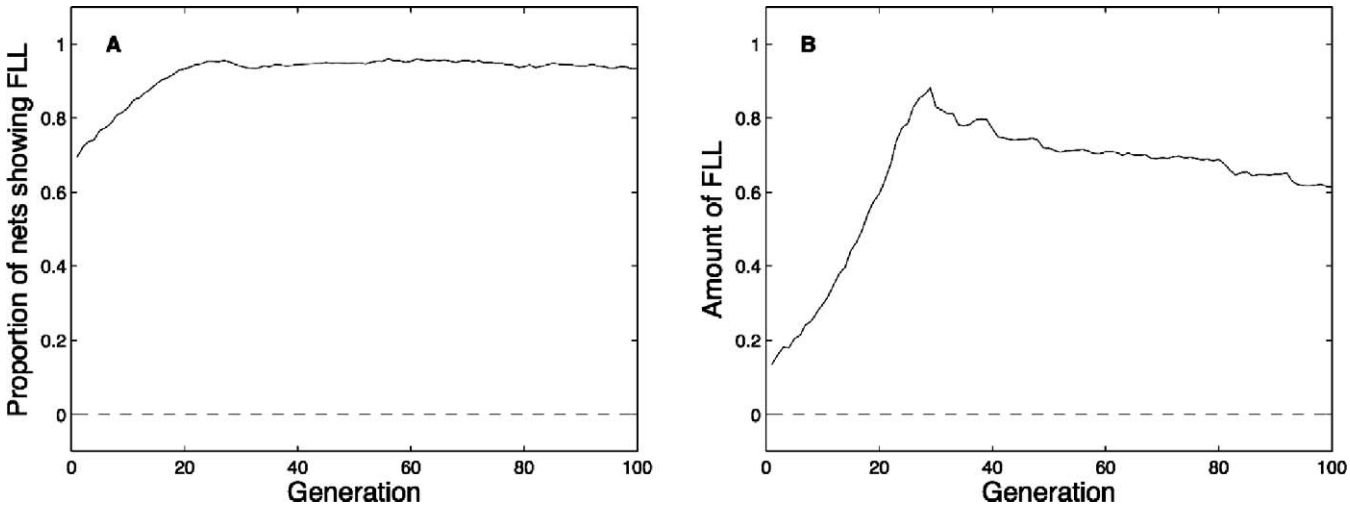

Figure 6. Prevalence and Amount of Free-Lunch Learning

The lines in each graph correspond to the conditions: FLL = **solid line**, NoFLL = **dashed line**. Performance error on A_1 was tested after learning A_2 in conditions FLL and NoFLL. Each plotted line is the mean of ten computer simulation runs (see Figure 4 for details).

(A) **Prevalence of FLL**: the proportion of networks which showed improved performance on A_1 after learning A_2. In condition FLL, the prevalence of FLL was non-zero in the first generation, as expected (see text), and increased across subsequent generations. In condition NoFLL, the prevalence remained at zero, as expected.

(B) **Amount of FLL**: in condition FLL, the amount of FLL increased dramatically over the first 30 generations. The absence of FLL in condition NoFLL is as expected (see text). The amount of FLL defined for this graph only is the difference in fitness error on A_1 before and after learning A_2, expressed as a proportion of the error on A_1 before learning A_2; or, equivalently, as $[E_{pre}(A_1) - E_{post}(A_1)]/E_{pre}(A_1)$.

doi:10.1371/journal.pcbi.0030147.g006

effect, such that *all* associations benefit from the reversion of a proportion (half here) of connection values.

2) Accelerated evolution of general phenotypic traits can be understood if we assume an extreme form of pleiotropy: that each of a given set of genes affects every phenotypic trait. This is equivalent to assuming that the genome is a *distributed representation* of the phenotype. Consider a population in which the fittest organism has a genome $\mathbf{w}_0$ which is perfectly adapted to its environment e_0. If the environment changes to e_1, then the fittest organism's genome will eventually evolve to a new state $\mathbf{w}_1$ that is suited to e_1 (this is analogous to forgetting in FLL). Now, consider what happens if the environment changes back to e_0. The fittest organism's genome will be forced back toward $\mathbf{w}_0$, but inevitably some genes will revert faster than others. For the sake of argument, assume that a subset G_2 of genes revert to their original values, while others G_1 remain as they were in $\mathbf{w}_1$ (this is analogous to relearning only A_2). Because each gene in G_2 contributes to every phenotypic trait, the reversion of genes in G_2 to their original values will push the entire phenotype back toward its state in the original environment e_0. Thus, the reappearance of an entire set of phenotypic traits (e.g., changes in size) can occur more quickly if those traits are encoded within a set of pleiotropic genes than if each trait is represented by a non-pleiotropic gene, and suggests a form of *free-lunch evolution*.

Conclusion

It has been demonstrated that FLL accelerates the evolution of behaviours in neural network models. Given that FLL appears to be a fundamental property of distributed representations, and given the reliance of neuronal systems on distributed representations, FLL-induced behaviours may constitute a significant component of apparently innate behaviours (e.g., nest-building). Results presented here suggest that any organism that did not take advantage of such a fundamental and ubiquitous effect would be at a selective disadvantage. Finally, if FLL accelerates evolution in the natural world, then it may have been involved in the Cambrian explosion, an explosion that began when brains (and therefore learning) first appeared.

Acknowledgments

Thanks to N. Hunkin and R. Lister for comments on this paper, and to P. Parpia for useful discussions. Thanks also to two anonymous reviewers for their detailed comments.

Author contributions. JVS conceived and designed the experiments, performed the experiments, analyzed the data, and wrote the paper.

Funding. The author received no specific funding for this study.

Competing interests. The author has declared that no competing interests exist.

References

1. Bateson G (1979) Mind and nature. Flamingo.
2. Waddington C (1959) Canalisation of development and genetic assimilation of acquired characters. Nature 183: 1654–1655.
3. Mery F, Kawecki T (2003) A fitness cost of learning in *Drosophila melangaster*. Proc Roy Soc London B 270: 2465–2469.
4. Price T, Qvarnstrom A, Irwin D (2003) The role of phenotypic plasticity in driving genetic evolution. Proc Roy Soc Lond B 270: 1433–1440.
5. Hinton G, Nowlan S (1987) How learning can guide evolution. Complex Systems 1: 495–502.
6. Baldwin J (1896) A new factor in evolution. Am Nat 30: 441–451; 536–553.
7. Mery F, Kawecki T (2004) The effect of learning on experimental evolution of resource preference in *Drosophila melangaster*. Evolution 58: 757–767.
8. Dopazo H, Gordon M, Perazzo R, Risau-Gusman S (2001) A model for the interaction of learning and evolution. Bull Math Biol 63: 117–134.
9. Stone J, Hunkin N, Hornby A (2001) Predicting spontaneous recovery of memory. Nature 414: 167–168.

10. Stone J, Jupp P (2007) Free-lunch learning: Modelling spontaneous recovery of memory. Neural Comput 19: 194–217.
11. Tinbergen N (1951) The study of instinct. Oxford: Oxford University Press.
12. Kaufman A, Dror G, Meilijson I, Ruppin E (2006) Gene expression of *Caenorhabditis elegans* neurons carries information on their synaptic connectivity. PLoS Comput Biol 2: 1561–1567.
13. Kauffman S, Levin S (1987) Towards a general theory of adaptive walks on rugged landscapes. J Theor Biol 128: 11–45.
14. Nolfi S, Elman J, Parisi D (1994) Learning and evolution in neural networks. Adaptive Behavior 3: 5–28.
15. Harvey I (1996) Relearning and evolution in neural networks. Adaptive Behaviour 4: 81–84.