Towards a Biomolecular Learning Machine

Matthew R. Lakin, Amanda Minnich, Terran Lane, and Darko Stefanovic

Department of Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
{mlakin,aminnich,terran,darko}@cs.unm.edu

Abstract. Learning and generalisation are fundamental behavioural traits of intelligent life. We present a synthetic biochemical circuit which can exhibit non-trivial learning and generalisation behaviours, which is a first step towards demonstrating that these behaviours may be realised at the molecular level. The aim of our system is to learn positive real-valued weights for a real-valued linear function of positive inputs. Mathematically, this can be viewed as solving a non-negative least-squares regression problem. Our design is based on deoxyribozymes, which are catalytic DNA strands. We present simulation results which demonstrate that the system can converge towards a desired set of weights after a number of training instances are provided.

1 Introduction

Learning and generalisation are fundamental capabilities of intelligent life. In biological organisms, learning takes place in the brain: a vastly complex, massively parallel biological computing device. In computer science, the field of machine learning has made great strides in designing and implementing learning algorithms on digital computers, which are themselves highly sophisticated machines. Our interest lies in the computational possibilities at the molecular scale of matter, using devices orders of magnitude smaller and simpler than a microchip or even a single neuron. Our goal is to demonstrate that complex learning behaviour is feasible at the molecular level.

In this paper we take the first steps towards designing and constructing synthetic biomolecular devices capable of learning and generalising from a series of training inputs. We choose DNA as our computational medium, because of its highly specific binding and innate programmability. In order to function as a learning device our biomolecular circuits must meet certain design criteria: *(i)* the circuit must be reusable, so multiple training instances can be presented sequentially; *(ii)* the parameters to be learned must persist in time across multiple training instances; and *(iii)* these parameters must be modulated by each use of the circuit, so learning can occur. We will present a chemically plausible design for a biomolecular learning device that meets these criteria.

Biomolecular computing is a promising field, in which biomolecules, such as nucleic acids, are rationally designed so that their interactions carry out some computational function. Our designs are based on deoxyribozymes, which are catalytically active single strands of DNA [8]. A deoxyribozyme catalyses the cleavage of a particular DNA substrate molecule, as outlined in Figure 1(a). Previous work [9] has shown that the catalytic activity of deoxyribozymes can be made conditional on the presence (or absence)

J. Durand-Lose and N. Jonoska (Eds.): UCNC 2012, LNCS 7445, pp. 152–163, 2012.

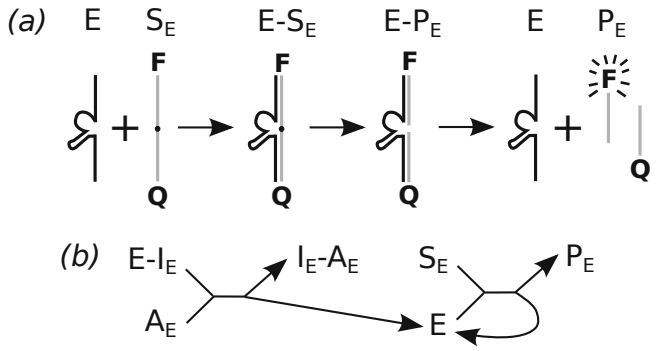

Fig. 1. Basic deoxyribozyme reactions. *(a)* Deoxyribozyme E catalyses the cleavage of a substrate molecule S_E into product molecules P_E. The substrate binding arms of the deoxyribozyme (shown in black) bind to the complementary substrate molecule (shown in grey). The catalytic action of the deoxyribozyme breaks the phosphodiester backbone of the substrate strand at an RNA base (shown as a small black circle), and the two product strands subsequently unbind from the deoxyribozyme. In the example above, the cleavage causes a rise in fluorescence since the fluorophore F and quencher Q attached to the substrate S are separated when S is cleaved. *(b)* We can inhibit a deoxyribozyme E by providing the inhibitor species I_E, so that the catalytically inactive E-I_E complex is formed. Adding an activator A_E removes the inhibitor to produce a waste complex I_E-A_E and an activated deoxyribozyme E, which catalyses the conversion of substrates S_E into products P_E.

of particular input DNA strands, thereby allowing them to function as logic gates. In this paper we employ deoxyribozyme reactions of the form shown in Figure 1(b), where an activator A_E converts the deoxyribozyme E from its inactive state into an active one [2]. Once a deoxyribozyme has been activated, it will continue to cleave substrates until either it is deactivated or it runs out of substrate. This makes deoxyribozymes ideal for use over extended periods of time or in situations where the circuit may need to be reused to process multiple sequential input signals.

We will tackle the problem of learning a linear function of the form $f(X_1,X_2) = w_1 \cdot X1 + w_2 \cdot X_2$, for positive input values X_1,X_2 and positive weights w_1,w_2. We restrict the inputs and weights to be positive as this makes it much simpler to represent them as concentrations of chemical species. The system begins with initial weights $\widehat{w_1}$ and $\widehat{w_2}$, which represent its current approximation $\widehat{f}(X_1,X_2) = \widehat{w_1} \cdot X_1 + \widehat{w_2} \cdot X_2$ to the target function f. The system is repeatedly presented with training instances of the form $(x_1,x_2,f(x_1,x_2))$ and responds by adjusting $\widehat{w_1}$ and $\widehat{w_2}$ so that $\widehat{f}$ better approximates f. Given enough training data, the system should converge on the correct weight values. Mathematically, this is equivalent to solving a non-negative least-squares regression problem [1]. This problem is simple enough that our solution could conceivably be implemented in the laboratory, while still displaying non-trivial learning behaviour. Our design is modular and the elements could, in principle, be replicated to learn similar functions of three or more inputs.

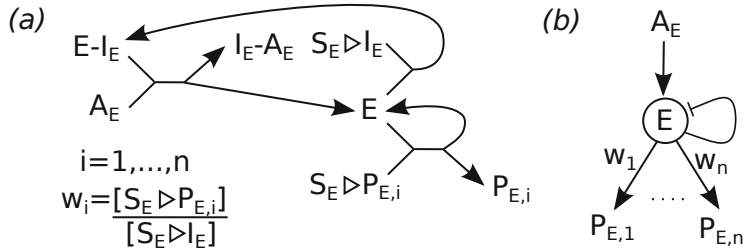

Fig. 2. Deoxyribozyme-based reusable multiplier design. *(a)* General reaction mechanism. When an activated deoxyribozyme E cleaves the self-inhibitory substrate $S_E \triangleright I_E$, the released inhibitor species I_E returns an activated deoxyribozyme to the inactive E-I_E state. Other competing substrates $S_E \triangleright P_{E,i}$ produce output products $P_{E,i}$ when they are cleaved, and the resulting concentrations of $P_{E,i}$ are multiples of the input concentration of the activator A_E. *(b)* Graphical shorthand notation for the general multiplier motif presented in *(a)*.

2 Deoxyribozyme-Based Signal Multipliers

In this section we present the basic computing motif for our design—a self-inhibiting deoxyribozyme gate that can serve as a reusable multiplier to scale up (or down) an input signal encoded as the concentration of some chemical species.

The design of our self-inhibiting deoxyribozyme gate is presented in Figure 2(a). As in Figure 1(b), we assume that the inactive deoxyriboyzme complex E-I_E can be activated by the addition of an activator A_E, producing inert waste I_E-A_E and the activated deoxyribozyme E. At this point, however, we introduce a number of additional concepts. The first of these is substrate molecules that sequester another chemical species until after they are cleaved, rather than just producing a fluorescent signal as shown in Figure 1(a). We write $S_E \triangleright X$ to denote a substrate for deoxyribozyme E which releases the chemical species X into solution after cleavage. In Figure 2(a) we use $i = 1, \ldots, n$ to represent n output signals $P_{E,1}, \ldots, P_{E,n}$ which are produced by the cleavage of n different substrate molecules $S_E \triangleright P_{E,1}, \ldots, S_E \triangleright P_{E,n}$. This allows an input signal to "fan out" and be sent to multiple different parts of the circuit. Note that these substrates must compete to bind to an activated deoxyribozyme E before they can be cleaved.

Our main technical innovation is the use of a special self-inhibiting substrate $S_E \triangleright I_E$, which sequesters the same inhibitory species I_E that was initially used to inactivate the deoxyribozyme E. When the deoxyribozyme cleaves one of these substrates, the I_E inhibitor which is released will react with an active deoxyribozyme E and return it to the inactive state E-I_E, as shown in Figure 2(a). When an input signal activates a certain number of inactive deoxyribozymes, cleavage of the $S_E \triangleright P_{E,i}$ substrates passes new signals on to the downstream computational elements, whereas the effect of the $S_E \triangleright I_E$ substrate will be eventually to return all of the deoxyribozymes to the inactive state, at which point they are ready to receive another input signal.

In order to analyse this design, we assume that all of the substrate molecules are present in excess relative to the number of activated deoxyribozymes. This means that we can neglect changes in the absolute populations of the substrate molecules due to

cleavage, allowing us to perform a simple mathematical analysis of the signal levels generated by the computational element presented in Figure 2(a). Suppose that amount X of the activator species A_E is provided, where X is less than the amount of inactive deoxyribozyme complexes E-I_E. This will result in X of the deoxyribozymes E being activated. In order to completely deactivate these again, X new inhibitors I_E must be produced, which means that X of the self-inhibiting substrates $S_E \triangleright I_E$ must be cleaved. Now, since there is competition from the other substrates $S_E \triangleright P_{E,i}$, during this time some output species $P_{E,i}$ will also be produced by cleavage events. Assuming that the rates of the cleavage reactions are all equal, the expected amount of $P_{E,i}$ produced is $w_i \cdot X$, where $w_i = \frac{[S_E \triangleright P_{E,I}]}{[S_E \triangleright I_E]}$ is the *weight* associated with output $P_{E,i}$. Thus the total concentration of each of the outputs is a multiple of the initial input concentration, and the weight (multiplicative factor) for each output is determined by the ratio of the substrate concentrations. Hence we refer to the computing motif from Figure 2(a) as a multiplier.

Since the $S_E \triangleright P_{E,i}$ molecules are consumed in proportions according to the associated weights, it is not hard to show that when all of the deoxyribozymes have been deactivated, the expected substrate ratios w_i are the same as the ratios were before the input was added. Thus computational circuits based on our self-inhibiting deoxyribozyme design are truly reusable.

Figure 2(b) shows our graphical shorthand for the network of chemical reactions presented in Figure 2(a), which we will use for the remainder of this paper. We use the flat-headed arrow to signify self-inhibition and we label the other output arcs with their weights, that is, the multiplicative factor applied to the input signal when generating that output signal.

3 Deoxyribozyme-Based Linear Classifiers

In this section we use the multiplier from Figure 2(a) to design a simple linear classifier circuit, as shown in Figure 3(a). This circuit accepts two input signals X_1 and X_2, which represent the two arguments to the function, along with a third input signal Y, which is the expected output. We assume that the weights w_1 and w_2 are already encoded as substrate concentration ratios as outlined above. Each of the deoxyribozymes E_i (for $i = 1,2$) produces an output concentration equal to $w_i \cdot [X_i]$, and since each produces the same output molecule $\widehat{Y}$ it follows that the total concentration of $\widehat{Y}$ produced is $w_1 \cdot [X_1] + w_2 \cdot [X_2]$.

For the final reaction at the bottom of Figure 3(a) we will assume that the Y and $\widehat{Y}$ species can react together very rapidly in order to annihilate, producing an inert waste complex Y-$\widehat{Y}$ (this is feasible if they are complementary DNA strands, for example). If the concentration of one of these species is greater than the other, then that species will remain after the rest has been annihilated, and the final concentration of that species will be the difference between the two values. Thus the circuit from Figure 2(a) will reach one of two steady states: if $[\widehat{Y}] > [Y]$ initially then $[Y] = 0$ and $[\widehat{Y}] = (w_1 \cdot X_1 + w_2 \cdot X_2) - [Y]$ in the steady state; or if $[Y] > [\widehat{Y}]$ initially then $[Y] = (w_1 \cdot X_1 + w_2 \cdot X_2) - [\widehat{Y}]$ and $[\widehat{Y}] = 0$ in the steady state. This is the desired behaviour for a linear classifier circuit.

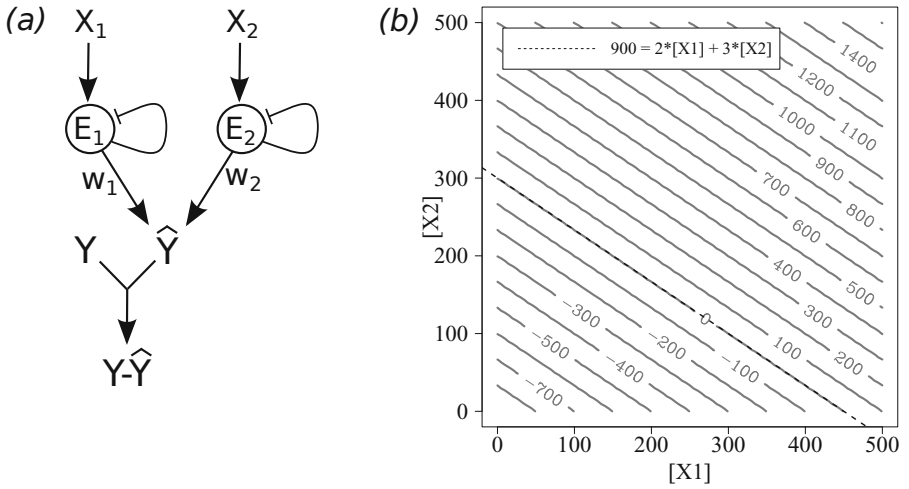

Fig. 3. Linear classifier circuit using deoxyribozymes. *(a)* Two instantiations of the reusable multiplier motif from Figure 2 can be combined to compute a weighted sum of the two input concentrations $[X_1]$ and $[X_2]$, represented as the concentration $[\widehat{Y}]$ of their shared output species $\widehat{Y}$. The concentration $[Y]$ of the third input species denotes the value with which the result of the sum should be compared. An annihilation reaction between $\widehat{Y}$ and Y results in an excess of whichever was present in the larger concentration. *(b)* Two-dimensional contour plot showing simulation results for the deoxyribozyme-based linear classifier, using the classifier line $900 = 2 \cdot [X_1] + 3 \cdot [X_2]$, for values of $[X_1]$ and $[X_2]$ ranging from 0 to 500 (arbitrary units) in increments of 5. Positive contour values denote a final excess of $\widehat{Y}$ and negative contour values denote a final excess of Y.

The contour plot in Figure 3(b) summarises simulation results from a test of our linear classifier circuit. We tested the circuit for initial concentrations $[X_1]$ and $[X_2]$ ranging from 0 to 500 in increments of 5. For each run, the initial substrate ratios were set as $w_1 = \frac{S_{E_1} \triangleright \widehat{Y}}{S_{E_1} \triangleright I_{E_1}} = 2$ and $w_2 = \frac{S_{E_2} \triangleright \widehat{Y}}{S_{E_2} \triangleright I_{E_2}} = 3$, and the initial concentration $[Y]$ was 900. Thus the system was set up to classify input pairs according to whether they plot above or below the classifier line $900 = 2 \cdot [X_1] + 3 \cdot [X_2]$. For each input, a time course was produced by integrating the ordinary differential equation model of the chemical reactions from above the dotted line in Table 1. These reactions correspond to the linear classifier design discussed here. The contour plot presents the output concentrations from the linear classifier simulations, interpreting a non-zero final concentration of $\widehat{Y}$ as a positive number and a non-zero final concentration of Y as a negative number. The plot shows that the output from the classifier circuit matches the expected behaviour—the line of zero output precisely matches the classifier line, and the other contours plot with the correct position and orientation relative to the classifier line. Thus we conclude that, for this range of inputs and these weight values, the linear classifier circuit works correctly according to its specification.

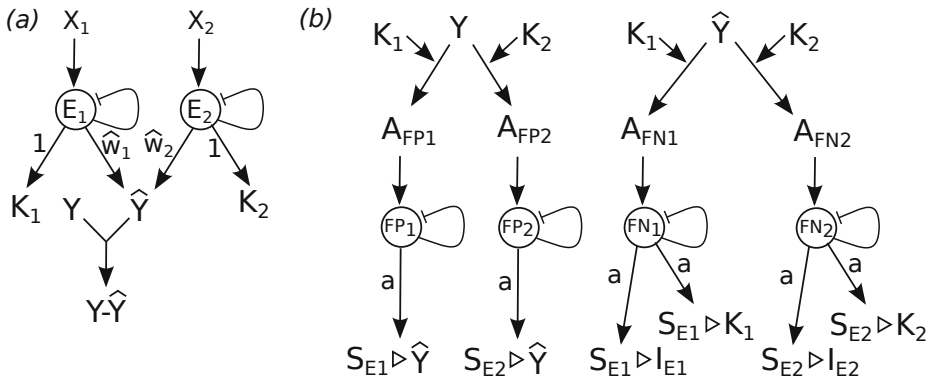

Fig. 4. Design for a biomolecular learning machine, conceptually split into two components. *(a)* The predictor component consists of the linear classifier circuit from Figure 3(a), with additional substrates and reactions which copy the input signals X_1 and X_2 into K_1 and K_2. *(b)* The feedback component uses the excess concentration of Y or $\widehat{Y}$ produced by the predictor component to modulate the weights in the predictor, by activating deoxyribozymes which generate new substrate molecules for the deoxyribozymes in the predictor. The arrows from the K_i species which point to reaction arrows denote catalysis of those reactions.

4 Feedback Reactions for Biomolecular Learning

In this section, we present chemical reactions and simulation results for our biomolecular learning machine. We build on the linear classifier design presented in the preceding section by adding a feedback phase, which modulates the weights used in the linear classifier according to the difference between the predicted and desired output values.

Figure 4 presents the full design for our biomolecular learning machine. For the sake of clarity, we divide the chemical reactions into conceptually distinct prediction (Figure 4(a)) and feedback (Figure 4(b)) phases. In practice we allow all the reactions to run simultaneously in simulations, and our initial results suggest that the results are very similar to those obtained from the system split into two temporally distinct phases.

Suppose that we wish to teach the machine weights w_1 and w_2 for the linear function $Y = w_1 \cdot X_1 + w_2 \cdot X_2$, using training instances $(x_1^1, x_2^1), \ldots, (x_1^n, x_2^n)$. We begin by setting up the initial concentrations of the substrates, inhibited deoxyribozyme complexes and other chemical species. The initial weights $\widehat{w_1}$ and $\widehat{w_2}$ are encoded in the ratios of the $S_{E_i} \triangleright \widehat{Y}$ and $S_{E_i} \triangleright I_{E_i}$ substrates, as described above. The first training instance (x_1^1, x_2^1) is presented by adding quantities of species X_1, X_2, and Y to the mixture such that $[X_1] = x_1^1$, $[X_2] = x_2^1$, and $[Y] = w_1 \cdot x_1^1 + w_2 \cdot x_2^1$. The X_1 and X_2 species activate the deoxyribozymes E_1 and E_2, and the reactions in Figure 4(a) proceed to compute the difference between the prediction $\widehat{Y}$ and the desired value Y supplied by the experimenter. The only difference between Figure 4(a) and Figure 3(a) is the addition of substrates $S_{E_1} \triangleright K_1$ and $S_{E_2} \triangleright K_2$, which each have weight 1 and therefore serve to take a copy of the input concentrations $[X_1]$ and $[X_2]$. This allows us to consume the inputs in order to compute $\widehat{Y}$, while remembering their original concentrations for later use.

The output signal from the reactions shown in Figure 4(a) is the difference between the prediction $\widehat{Y}$ and the desired value Y, expressed as a non-zero concentration of either Y (if $Y > \widehat{Y}$) or $\widehat{Y}$ (if $\widehat{Y} > Y$). The concentrations of these two species, together with the copied input values represented as concentrations of the K_i species, can be thought of as the inputs to the feedback phase of the learning machine, whose design is shown in Figure 4(b). The effect of the feedback phase is to use the output signal from the predictor in order to adjust the predictor's weights towards the target values. Since the weights in the predictor are encoded as substrate ratios, it follows that in order to adjust the weights we must modify those substrate concentrations. Thus we introduce the notion of a *presubstrate*, that is, a species which serves as the substrate for one deoxyribozyme and which, upon cleavage, releases a species which serves as the substrate for another deoxyribozyme. Using our notation from above, we write $S_A \triangleright S_B \triangleright C$ for a presubstrate molecule that serves as a substrate for deoxyribozyme A and, when cleaved, releases a substrate for deoxyribozyme B that produces species C on cleavage. For example, in Figure 4(b) the presubstrate for FN_1, which generates a self-inhibitory substrate $S_{E_1} \triangleright I_{E_1}$ for E_1, is $S_{FN_1} \triangleright S_{E_1} \triangleright I_{E_1}$. Now, there are two cases to consider.

Excess of Y. If Y is left over from the linear classifier phase then $Y > \widehat{Y}$, so we must increase the weight estimates $\widehat{w_1}$ and $\widehat{w_2}$ to reflect this. This is achieved using the two deoxyribozymes FP_1 and FP_2 on the left-hand side of Figure 4(b). In order to take a learning step in (approximately) the right direction in weight vector space, we must adjust weight w_i by an amount proportional to the input value X_i, in the correct direction. To achieve this, we assume that the K_i species produced during the predictor phase catalyse the transformation of Y into the activator species A_{FP_1} and A_{FP_2}, at the same rate. Since the concentrations of the K_i species are a copy of the original concentrations of the X_i inputs, the amount of A_{FP_i} produced will be $\frac{[K_i]}{[K_1]+[K_2]} \cdot [Y]$. Since $[Y]$ at this point is actually the difference $Y - \widehat{Y}$, the amount of A_{FP_i} generated is proportional to the excess of $\widehat{Y}$ and the original input concentration of X_i. The activators then activate their target deoxyribozymes FP_i, which multiply this signal by the learning rate a and generate additional substrate molecules $S_{E_i} \triangleright \widehat{Y}$ for the E_i deoxyribozymes in the predictor phase. Since the weights in the predictor are encoded as substrate concentrations $\widehat{w_i} = \frac{[S_{E_i} \triangleright \widehat{Y}]}{[S_{E_i} \triangleright I_{E_i}]}$, this serves to increase the weights w_i additively, by increasing the numerator. The new weight value $\widehat{w_i}'$ is related to the previous weight value $\widehat{w_i}$ by

$$\widehat{w_i}' = \frac{[S_{E_i} \triangleright \widehat{Y}] + a \cdot ([Y] - [\widehat{Y}]) \cdot \frac{X_i}{X_1+X_2}}{[S_{E_i} \triangleright I_{E_i}]} = w_i + \frac{a \cdot ([Y] - [\widehat{Y}]) \cdot \frac{X_1}{X_1+X_2}}{[S_{E_i} \triangleright I_{E_i}]}.$$

Excess of $\widehat{Y}$. If $\widehat{Y}$ is left over then $\widehat{Y} > Y$, so we must decrease $\widehat{w_1}$ and $\widehat{w_2}$. For this we use the FN_1 and FN_2 deoxyribozymes on the right-hand side of Figure 4(b). The mechanism here is similar to that described above for increasing the weights: the main difference is that the FN_i deoxyribozymes generate self-inhibitory substrates $S_{E_i} \triangleright I_{E_i}$ for the predictor deoxyribozymes, and since the weights in the predictor are encoded as substrate concentrations $\widehat{w_i} = \frac{[S_{E_i} \triangleright \widehat{Y}]}{[S_{E_i} \triangleright I_{E_i}]}$, this causes the weights w_i to decrease. There is an additional subtlety in this case: since we are generating additional $S_{E_i} \triangleright I_{E_i}$ substrates

for the predictor deoxyribozymes we must also generate the same amount of additional $S_{E_i} \triangleright K_i$ substrates. This is necessary because the concentration of K_i generated is intended to be the same as the input concentration $[X_i]$, which requires that the weight be 1 on the arrow to K_i from E_i in Figure 3(a). Adding extra $S_{E_i} \triangleright I_{E_i}$ substrates for E_i would decrease this weight, so we must also generate the same amount of additional $S_{E_i} \triangleright K_i$ substrates to compensate. Note that, in this case, the weight update operation is not additive, since generating more $S_{E_i} \triangleright I_{E_i}$ substrate molecules increases the denominator of the substrate ratio. This reflects the fact that our weight values are restricted to be positive, so they can only be decreased asymptotically towards zero. The new weight value $\widehat{w_i}'$ is related to the previous weight value $\widehat{w_i}$ by

$$\widehat{w_i}' = \frac{[S_{E_i} \triangleright \widehat{Y}]}{[S_{E_i} \triangleright I_{E_i}] + a \cdot ([\widehat{Y}] - [Y]) \cdot \frac{X_i}{X_1 + X_2}}.$$

Once the feedback reactions reach steady state, the predictor weights will have been modified and we can begin preparations for the next training phase. Since the K_i species only act as catalysts in the feedback phase, they must somehow be filtered out of the mixture at the end of the feedback phase, which we model by resetting $[K_1]$ and $[K_2]$ to zero. Finally, the learning rates a of the feedback deoxyribozymes must be reduced, so the learning mechanism takes smaller steps over time. For simplicity we assume that the experimenter performs this step by adding a constant amount of the self-inhibitory substrates $S_{FP_1} \triangleright I_{FP_1}$, $S_{FP_2} \triangleright I_{FP_2}$, $S_{FN_1} \triangleright I_{FN_1}$ and $S_{FN_2} \triangleright I_{FN_2}$ at the end of each cycle.

When these steps have been taken, the system is ready to receive the next training instance. Moving on to the second training instance (x_1^2, x_2^2), the experimenter adds more of species X_1, X_2, and Y in quantities such that $[X_1] = x_1^2$, $[X_2] = x_2^2$, and $[Y] = w_1 \cdot x_1^2 + w_2 \cdot x_2^2$ (the action of the previous cycle will have reduced these concentrations back to zero) and the reactions proceed as before. This procedure can be iterated for many training instances, provided that the deoxyribozymes do not run out of substrate. In our examples we use large initial substrate concentrations to avoid this problem.

5 Results of Learning Simulations

The intention is that, as more training instances are presented, the weight values $\widehat{w_1}$ and $\widehat{w_2}$ stored in the predictor should approach the target weight values w_1 and w_2. Since the mathematics of our weight update mechanism is slightly different from the classic perceptron learning rule [4], we do not have a formal proof that the system is guaranteed to converge on the correct result for all learnable functions. However, results from our simulations indicate that convergence to approximately correct weights is possible.

We modelled the learning process using the full set of chemical reactions listed in Table 1. We used weights from the interval $[0, 10]$ and inputs X_1 and X_2 from the interval $[0, 500]$. The initial concentrations of self-inhibitory substrates were set at 100,000, with the initial concentrations of signal-generating substrates chosen to encode the initial weights $\widehat{w_1}$ and $\widehat{w_2}$ and the initial learning rate a, which was set to 250. The initial concentration of each inhibited deoxyribozyme was set to 20,000 to cover the full range of possible output values. At the beginning of each simulated training cycle, the system

Table 1. Full chemical reactions for the biomolecular learning machine. In our simulations we use rate constants $vfast = 100.0$, $fast = 1.0$ and $slow = 0.01$, with arbitary units. These rates were chosen as they broadly reflect the expected separation of timescales for the intended deoxyribozyme reactions. The reactions above the dotted line correspond to the linear classifier design from Figure 3(a). The reactions below the dotted line implement the feedback mechanism, and together they yield the full learning circuit design presented in Figure 4.

Description	Reactants		Products
E_1 activation	$E_1\text{-}I_{E_1} + X_1$	$\xrightarrow{fast}$	$E_1 + I_{E_1}\text{-}X_1$
E_1 self-inhibition	$E_1 + S_{E_1} \triangleright I_{E_1}$	$\xrightarrow{slow}$	$E_1\text{-}I_{E_1}$
E_1 producing $\widehat{Y}$	$E_1 + S_{E_1} \triangleright \widehat{Y}$	$\xrightarrow{slow}$	$E_1 + \widehat{Y}$
E_2 activation	$E_2\text{-}I_{E_2} + X_2$	$\xrightarrow{fast}$	$E_2 + I_{E_2}\text{-}X_2$
E_2 self-inhibition	$E_2 + S_{E_2} \triangleright I_{E_2}$	$\xrightarrow{slow}$	$E_2\text{-}I_{E_2}$
E_2 producing $\widehat{Y}$	$E_2 + S_{E_2} \triangleright \widehat{Y}$	$\xrightarrow{slow}$	$E_2 + \widehat{Y}$
Annihilation of Y and $\widehat{Y}$	$Y + \widehat{Y}$	$\xrightarrow{vfast}$	$Y\text{-}\widehat{Y}$
E_1 producing K_1	$E_1 + S_{E_1} \triangleright K_1$	$\xrightarrow{slow}$	$E_1 + K_1$
E_2 producing K_2	$E_2 + S_{E_2} \triangleright K_2$	$\xrightarrow{slow}$	$E_2 + K_2$
A_{FP_1} production	$K_1 + Y$	$\xrightarrow{slow}$	$K_1 + A_{FP_1}$
A_{FP_2} production	$K_2 + Y$	$\xrightarrow{slow}$	$K_2 + A_{FP_2}$
A_{FN_1} production	$K_1 + \widehat{Y}$	$\xrightarrow{slow}$	$K_1 + A_{FN_1}$
A_{FN_2} production	$K_2 + \widehat{Y}$	$\xrightarrow{slow}$	$K_2 + A_{FN_2}$
FP_1 activation	$FP_1\text{-}I_{FP_1} + A_{FP_1}$	$\xrightarrow{fast}$	$FP_1 + I_{FP_1}\text{-}A_{FP_1}$
FP_1 self-inhibition	$FP_1 + S_{FP_1} \triangleright I_{FP_1}$	$\xrightarrow{slow}$	$FP_1\text{-}I_{FP_1}$
FP_1 producing $S_{E_1} \triangleright \widehat{Y}$	$FP_1 + S_{FP_1} \triangleright S_{E_1} \triangleright \widehat{Y}$	$\xrightarrow{slow}$	$FP_1 + S_{E_1} \triangleright \widehat{Y}$
FP_2 activation	$FP_2\text{-}I_{FP_2} + A_{FP_2}$	$\xrightarrow{fast}$	$FP_2 + I_{FP_2}\text{-}A_{FP_2}$
FP_2 self-inhibition	$FP_2 + S_{FP_2} \triangleright I_{FP_2}$	$\xrightarrow{slow}$	$FP_2\text{-}I_{FP_2}$
FP_2 producing $S_{E_2} \triangleright \widehat{Y}$	$FP_2 + S_{FP_2} \triangleright S_{E_2} \triangleright \widehat{Y}$	$\xrightarrow{slow}$	$FP_2 + S_{E_2} \triangleright \widehat{Y}$
FN_1 activation	$FN_1\text{-}I_{FN_1} + A_{FN_1}$	$\xrightarrow{fast}$	$FN_1 + I_{FN_1}\text{-}A_{FN_1}$
FN_1 self-inhibition	$FN_1 + S_{FN_1} \triangleright I_{FN_1}$	$\xrightarrow{slow}$	$FN_1\text{-}I_{FN_1}$
FN_1 producing $S_{E_1} \triangleright I_{E_1}$	$FN_1 + S_{FN_1} \triangleright S_{E_1} \triangleright I_{E_1}$	$\xrightarrow{slow}$	$FN_1 + S_{E_1} \triangleright I_{E_1}$
FN_1 producing $S_{E_1} \triangleright K_1$	$FN_1 + S_{FN_1} \triangleright S_{E_1} \triangleright K_1$	$\xrightarrow{slow}$	$FN_1 + S_{E_1} \triangleright K_1$
FN_2 activation	$FN_2\text{-}I_{FN_2} + A_{FN_2}$	$\xrightarrow{fast}$	$FN_2 + I_{FN_2}\text{-}A_{FN_2}$
FN_2 self-inhibition	$FN_2 + S_{FN_2} \triangleright I_{FN_2}$	$\xrightarrow{slow}$	$FN_2\text{-}I_{FN_2}$
FN_2 producing $S_{E_2} \triangleright I_{E_2}$	$FN_2 + S_{FN_2} \triangleright S_{E_2} \triangleright I_{E_2}$	$\xrightarrow{slow}$	$FN_2 + S_{E_2} \triangleright I_{E_2}$
FN_2 producing $S_{E_2} \triangleright K_2$	$FN_2 + S_{FN_2} \triangleright S_{E_2} \triangleright K_2$	$\xrightarrow{slow}$	$FN_2 + S_{E_2} \triangleright K_2$

was perturbed by introducing appropriate amounts of the input species X_1, X_2, and Y. After waiting sufficient time for the system to reach steady state (we chose 10 time units) the learning rate was annealed by adding 2,500 units of self-inhibitory substrate for each of the four deoxyriboymes in the feedback phase, as described above. The input

species for the next training cycle were then added, and so on. Between perturbations, the evolution of the species concentrations was computed by integrating the ordinary differential equation model of the full set of reactions from Table 1.

Figure 5 presents results from our learning simulations. The left-hand plot shows the evolution of the machine's predicted weight values over a number of training sequences. The target weight values were fixed at $w_1 = w_2 = 5$ (the intersection of the dotted lines) and the starting weight values are denoted by filled circles. Each line corresponds to the presentation of 25 randomly-chosen training instances, normally distributed across the weight interval described above. As more training instances are presented, in each case the machine appears to converge towards the target weight values, as expected. We ran 10 training sequences in total (data not all shown) and all converged to the vicinity of the $w_1 = w_2 = 5$ target weights. The right-hand plot shows a learning curve for the molecular learning machine. with a logarithmic scale on the y axis, For each of 10 randomly-selected initial settings, that is, target weight vectors and initial weight values, we presented sequences of random training data for a variety of sequence lengths S, with S ranging from 1 to 100. The entire system was restarted from scratch for each training sequence. After each training sequence we computed the root mean squared error in the predictions computed by the resulting machine on a fixed, random set of 100 test instances. This performance metric was averaged over the 10 different initial settings. The plot shows that, as the amount S of training data is increased, the average predictive performance of the machine improves, although the rate of improvement does seem to flatten out over time. Hence approximately correct generalisation is taking place.

6 Discussion

We have presented a design for a biochemical learning machine using deoxyribozymes as the basic computational elements. The data presented in Figure 5 are evidence that learning at the biomolecular level may be possible.

We intend to further quantify the performance of our design, such as its convergence properties and sensitivity to variations in starting parameters and reaction rates, and to noise in the training data. We also plan to extend and refine the design presented in this paper. In particular, we hope that improvements will further reduce the error rates shown in the plot on the right-hand side of Figure 5 and lead to better convergence across the entire weight space. This might be achieved by tuning the substrate concentrations and learning parameters of the circuit. Furthermore, the class of functions that our design can learn does not include a bias term, which restricts it to functions f such that $f(0,0) = 0$. Extending the design to learn more general functions of the form $f(X_1, X_2) = w_0 + w_1 \cdot X_1 + w_2 \cdot X_2$ would require additional computational elements to deal with the third weight. It should also be straightforward to extend our design to learn functions with more than two inputs, since the components can simply be duplicated to process additional input signals.

Our use of real-valued inputs and outputs means that our learning machine does not adhere to the classic definition of a perceptron [4]. Our design could form the basis of a perceptron but we would need to use binary, as opposed to real-valued, inputs and outputs, as well as thresholding and amplification operations to introduce a non-linear

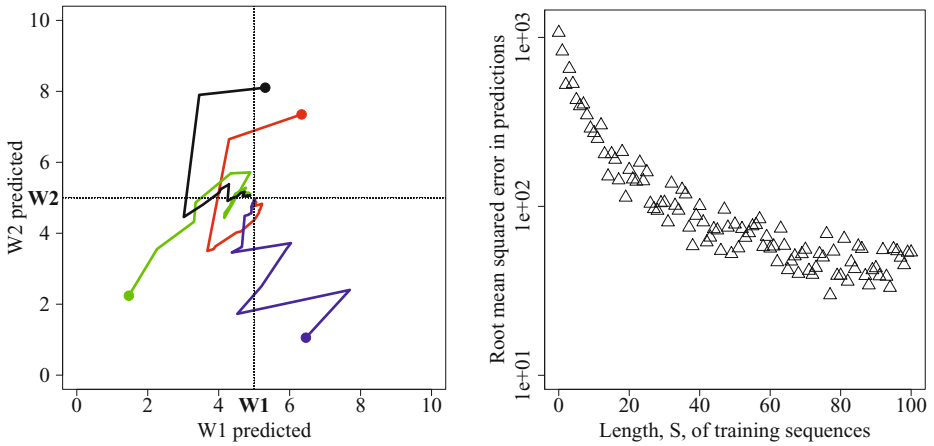

Fig. 5. Results of learning simulations. *(Left)* Weight-space plot showing modification of weight values in multiple training runs of the biomolecular learning machine. As additional training instances are supplied, the predicted weights $\widehat{w_1}$ and $\widehat{w_2}$ approach the target values $w_1 = w_2 = 5$, indicated by the intersection of the dotted lines. *(Right)* Learning curve for the biomolecular learning machine. with a logarithmic scale on the y axis. We computed the root mean squared error in predictions after exposure to random training sequences of length S, for S between 1 and 100. The plotted errors were averaged over 10 initial settings, that is, random target weights and initial weight values. On average, the error decreases when more training data are presented, which suggests that learning is occurring.

response. This might also make the learning behaviour less sensitive to errors. Finally, our design is restricted to positive weights and positive input values. Generalising it to handle negative weights and inputs would be non-trivial, but necessary in order to achieve the full computational power of perceptrons in a feedforward neural network.

Our long-term goal is to construct a biomolecular learning device in the laboratory. The number of circuit elements required to implement our design is well within the scale of circuits that have been demonstrated experimentally, both using deoxyribozymes [3,5] and other techniques such as DNA strand displacement [11,6,7], so a biochemical implementation is not implausible. With this in mind, our design was intended to follow the kind of reactions that could potentially be run in the laboratory. The main technical challenges are likely to be the scalability of the circuit and the design of multi-cleavage presubstrate molecules. We will also need a better means of splitting the Y and $\widehat{Y}$ signals at the start of the feedback phase, to avoid filtering out the K_i species after each training cycle.

In related work, Zhang and Seelig [10] designed and simulated linear classifiers using catalytic amplifiers based on DNA strand displacement, based on mathematics similar to those presented in Section 2 above. Their design was capable of handling negative weights, but their circuits could not be reused to process multiple inputs. Qian *et al.* [7] constructed a neural network *in vitro* using strand displacement cascades, but their neural network was trained *in silico* and the weights were subsequently hard-coded into

each molecular implementation, which could not be reused. Pei *et al.* [5] used deoxyribozymes to construct an automaton that could be programmed to play any strategy in a simple two-player game. The strategies were hard-coded, but the training inputs could be removed and re-programmed to select a different strategy.

Acknowledgments. This material is based upon work supported by the National Science Foundation under grants 1027877 and 1028238.

References

1. Lawson, C.L., Hanson, B.J.: Solving least squares problems. Prentice-Hall, Englewood Cliffs (1974)
2. Lederman, H., Macdonald, J., Stefanovic, D., Stojanovic, M.N.: Deoxyribozyme-based three-input logic gates and construction of a molecular full adder. Biochemistry 45(4), 1194–1199 (2006)
3. Macdonald, J., Li, Y., Sutovic, M., Lederman, H., Pendri, K., Lu, W., Andrews, B.L., Stefanovic, D., Stojanovic, M.N.: Medium scale integration of molecular logic gates in an automaton. Nano Letters 6(11), 2598–2603 (2006)
4. Minsky, M., Papert, S.: Perceptrons: an introduction to computational geometry, 2nd edn. MIT Press, Cambridge (1972)
5. Pei, R., Matamoros, E., Liu, M., Stefanovic, D., Stojanovic, M.N.: Training a molecular automaton to play a game. Nature Nanotechnology 5, 773–777 (2010)
6. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science 332, 1196–1201 (2011)
7. Qian, L., Winfree, E., Bruck, J.: Neural network computation with DNA strand displacement cascades. Nature 475, 368–372 (2011)
8. Santoro, S.W., Joyce, G.F.: A general-purpose RNA-cleaving DNA enzyme. PNAS 94, 4262–4266 (1997)
9. Stojanovic, M.N., Mitchell, T.E., Stefanovic, D.: Deoxyribozyme-based logic gates. JACS 124, 3555–3561 (2002)
10. Zhang, D.Y., Seelig, G.: DNA-Based Fixed Gain Amplifiers and Linear Classifier Circuits. In: Sakakibara, Y., Mi, Y. (eds.) DNA16 2010. LNCS, vol. 6518, pp. 176–186. Springer, Heidelberg (2011)
11. Zhang, D.Y., Seelig, G.: Dynamic DNA nanotechnology using strand-displacement reactions. Nature Chemistry 3, 103–113 (2011)