Design of a biochemical circuit motif for learning linear functions

Matthew R. Lakin1, Amanda Minnich1, Terran Lane1 and Darko Stefanovic1,2

1Department of Computer Science, and 2Center for Biomedical Engineering, University of New Mexico, Albuquerque, NM, USA

Learning and adaptive behaviour are fundamental biological processes. A key goal in the field of bioengineering is to develop biochemical circuit architectures with the ability to adapt to dynamic chemical environments. Here, we present a novel design for a biomolecular circuit capable of supervised learning of linear functions, using a model based on chemical reactions catalysed by DNAzymes. To achieve this, we propose a novel mechanism of maintaining and modifying internal state in biochemical systems, thereby advancing the state of the art in biomolecular circuit architecture. We use simulations to demonstrate that the circuit is capable of learning behaviour and assess its asymptotic learning performance, scalability and robustness to noise. Such circuits show great potential for building autonomous in vivo nanomedical devices. While such a biochemical system can tell us a great deal about the fundamentals of learning in living systems and may have broad applications in biomedicine (e.g. autonomous and adaptive drugs), it also offers some intriguing challenges and surprising behaviours from a machine learning perspective.

1. Introduction

Learning is a fundamental process of life. To survive, organisms must respond to dynamic environments in the face of incomplete information. Simple adaptation by desensitization of chemical sensors plays a key role in many cellular processes, for example bacterial chemotaxis [1], and associative learning has been demonstrated in paramecia [2] and sea slugs [3]. Furthermore, theoretical work has shown that gene regulatory networks may be capable of acquiring predictive behaviour and associative learning [4–6]. However, despite significant recent advances in molecular programming [7,8], DNA self-assembly [9,10] and synthetic biology [11–14], an engineered biochemical system with learning capabilities has not yet been implemented in the laboratory. Solving this problem would enable significant advances in autonomous, adaptive in vivo nanomedicine. For example, the genomes of viruses are continually evolving, and a biochemical learning device could adapt to previously unseen threats, dynamically augmenting the immune system in order to fight the viruses. Therefore, the development of biomolecular circuits that exhibit even conceptually simple learning algorithms would be of significant practical interest. In this paper, we consider how learning might be implemented in an engineered system constructed from biochemical components.

Recent advances in nanotechnology and biotechnology have given us unprecedented control over matter at the nanoscale [15,16]. Previous work on protein-based biocomputing shows promise [17–22]; however, the predictable folding and sequence-specific interactions of DNA make it an attractive material for implementing computation at the nanoscale. Since Adleman’s seminal paper [23], many researchers have exploited these properties, leading to many recent advances in the fields of DNA computing [7,8,24–27], DNA self-assembly [9,28–33] and synthetic biology [11,12,34–36]. To exhibit learning behaviour, a biochemical system must be able to: (i) store its current approximation to the parameters it is trying to learn, (ii) receive stimuli from the environment and (iii) respond by updating its stored parameter values appropriately. For supervised learning, the system must be able to remain operational...
for an extended period of time so that multiple training rounds can be conducted. These are not challenging requirements for a machine learning algorithm implemented using a digital electronic computer, but we are restricting ourselves to the limited computing components available in a biochemical setting. Most published work in molecular computing has been on one-shot computational devices that can compute a single function a single time without any ability to store or update the values of variables [7,8,25,37,38]. Recent work has also looked at reversible computational elements, which modify their state when their inputs change [39], and devices that can be reprogrammed [40], though these systems still cannot store or retrieve values from memory. Synthetic biologists have demonstrated rewritable binary memory in living cells using recombine enzymes [13,14], though these exploit the complex metabolic system of the cell to absorb chemical energy from the environment to keep the system operating out of equilibrium.

Here, we present a design for a modular biochemical circuit that implements a simple machine learning algorithm, which is tailored towards our novel biochemical implementation. Our learning circuit can remember and update its current approximation to the parameters it is trying to learn. It also remains operational for an extended period of time, so that multiple rounds of training can be conducted, and multiple predictions can be made using the learned weights. We consider supervised learning, in which the system is presented with a number of sequential training inputs, which cause its internal state to be modified, followed by a number of testing inputs, which we use to assess the performance of the machine. In our biochemical model, each of these events corresponds to a mixing event, in which new chemical species are added to the system. The newly added amounts of each chemical species correspond to the values of input variables at each step. As is standard in biochemical modelling, we use mass-action chemical simulations to analyse the performance of our learning circuit. Our learning circuit design is tailored towards a direct experimental implementation using DNAzymes [41]. Therefore, we only use reaction schemes that have either already been experimentally demonstrated or which are chemically plausible.

2. Related work

Previous work applying molecular logic, including DNAzymes, to the problem of classifying and eliminating cancer cells [35,42,43] highlights the potential impact of the DNAzyme-based approach. It is likely that the application of simple machine learning techniques will enable significant advances in these fields by allowing devices to learn appropriate responses to their surroundings rather than relying on a fixed, pre-programmed pattern of behaviour.

Zhang and Seelig have published a design for a linear classifier circuit [44], which is based on DNA strand displacement reactions [24] and, in particular, on cooperative hybridization [45]. That design uses damper species to absorb fuel strands in a catalytic cycle in order to achieve a specified gain, and is therefore similar in spirit to our system, with the crucial distinction that our circuit motif can be reused to process multiple input signals, since it is designed using a reusable DNAzyme-based multiplier motif. However, the linear classifier from [44] is capable of handling negative values, which our design will not consider.

An alternative approach to linear classifier implementation has been demonstrated using competitive toehold activation [46], though this design is restricted to weights between zero and one due to the lack of amplification in the circuit design [47].

Banda et al. [48,49] have simulated several kinds of trainable perceptron using artificial chemistry, which is conceptually similar to our work but was designed using more abstract chemical reactions. It has been shown that such arbitrary systems of chemical reactions can be faithfully implemented in a biochemical setting by compiling them into an enlarged system of DNA strand displacement reactions [50]. By contrast, our design is influenced by more practical considerations and is targeted to a specific implementation framework using DNAzymes. This makes it potentially more amenable to a direct experimental implementation, because we can simply design the appropriate DNAzyme components and the required interactions will follow, as opposed to designing multiple strand displacement gates and reactions to implement every single reaction from a high-level chemical reaction network.

In other related theoretical work, Kim et al. proposed implementations of neural networks using in vitro transcriptional circuits [51], including constructs for Hopfield networks and winner-takes-all learning systems, and circuits have been previously proposed for Hebbian learning in synthetic biological transcription networks [5,6]. In some impressive experimental work, Qian et al. [8] demonstrated a biochemical implementation of a Hopfield neural network [52] using strand displacement-based ‘seesaw’ gates [37]. In that work, the neural network was trained in silico and the final weights were hard-coded as a collection of DNA molecules, which could only process a single input before being exhausted.

Although we will present our circuit motif using a DNAzyme-based motif, we believe that similar behaviour could be implemented experimentally using other molecular computing techniques [53], such as DNA strand displacement [8,24,54] or enzyme-based techniques such as genelets [55,56] or the DNA toolbox approach [57,58]. Thus, the circuit motif presented here is of general interest for the development of biomolecular learning systems.

3. Learning problem and algorithm

In this paper, we study a biochemical circuit capable of supervised learning of optimal non-negative weight constants for linear functions. The hypothesis space for this problem is given by

\[ f(X) = W^T X, \]

where \( W, X \in \mathbb{R}_+^d \) are vectors of \( d > 0 \) non-negative input values and non-negative weights, respectively. The state of the system comprises a vector of weights \( W \), which represent the current approximation \( f(X) = W^T X \) to the target function \( f \). The system is repeatedly presented with training instances of the form \( (X,f(X)) \) and responds by adjusting \( W \), with the intention that the new \( f \) will be a better approximation to \( f \) than the old \( f \). This is a well-known problem, and there are a large number of algorithms that can solve it to arbitrary precision given sufficient training data [59].
To readers familiar with machine learning, this may seem a straightforward problem—the problem dates back at least to Gauss, and solutions based on least-squares regression [60,61] have been well established for decades. However, it is important to keep in mind that implementing those solutions requires sophisticated machines with, comparatively, enormous amounts of supporting hardware and software. Here we implement the solution to the non-negative subproblem purely in biomolecular computation, without even the luxury of a cell or metabolic supporting system. In that context, this seemingly simple problem is quite challenging indeed.

Figure 1 presents the primitive operations needed to implement a supervised machine learning algorithm. The most fundamental issue is how to represent the stored values of the weight approximations. In a biochemical system, the most obvious solution is to encode certain values as the concentrations of particular chemical species. However, species concentrations must be non-negative, so representing negative values in this way is non-trivial. Therefore, for simplicity we restrict ourselves to non-negative parameter values. To take advantage of the competitive binding reactions that will occur in our biochemical model, we will encode certain parameters as the ratio of two species concentrations. This allows us to increase the parameter value or decrease it asymptotically towards zero, by purely additive updates to the numerator or the denominator values as the concentrations of particular chemical species.

Algorithm 1. Pseudocode for our learning algorithm for supervised learning of functions of the form shown in equation (3.1). We assume constants $\delta$ for the denominators of concentration ratios, $\chi$ for the annealing constant and $W^{\text{init}}$ for the initial vector of weight approximations. Each weight approximation $\hat{w}_i$ is encoded by the ratio $\hat{w}_i^N/\hat{w}_i^D$, and the learning rate $\alpha$ is represented as $\alpha^N/\alpha^D$.

```
1: $\alpha^D \leftarrow \delta$
2: $\alpha^N = \alpha^{\text{init}} \alpha^D$ //Set-up learning rates.
3: for $j = 1 \rightarrow d$ do
4:   $\hat{w}_j^D \leftarrow \delta$
5:   $\hat{w}_j^N \leftarrow \hat{w}_j^{\text{init}} \hat{w}_j^D$ //Set-up weights.
6: end for
7: for $i = 1 \rightarrow n$ do
8:   $\hat{y} = \sum_{j=1}^d ((\hat{w}_j^N/\hat{w}_j^D)x_i) \hat{y}$ //Current predicted result.
9:   for $j = 1 \rightarrow d$ do
10:  $z_i = \sum_{j=1}^d \abs(y_i - \hat{y})$
11: end for
12: if $y_i > \hat{y}$ then
13:   $W^N \leftarrow W^N + (\alpha^N/\alpha^D)Z$ //Increase weights.
14: else if $y_i < \hat{y}$ then
15:   $W^D \leftarrow W^D + (\alpha^N/\alpha^D)Z$ //Decrease weights.
16: end if
17: $\alpha^D \leftarrow \alpha^D + \chi$ //Anneal learning rate.
18: end for
```

The main loop of algorithm 1 (lines 7–18) accepts the values of the input species $X$ associated with the next training instance to be provided to the system. It then computes the current approximation $\hat{y}$ (line 8) and compares it with the true answer $y$ (lines 12 and 16). The use of the $\abs$ function when computing the adjustments applied to the weight vector (line 10) is necessary to prevent non-negative intermediate values: the two cases are handled separately (lines 12 and 13 and lines 14 and 15). If $y > \hat{y}$, then the weights are increased by adding to the numerators (lines 12 and 13), and if $y < \hat{y}$, then the weights are decreased by adding to the denominators (lines 14 and 15) (these lines are expressed in vector notation for clarity). The algorithm adjusts each approximated weight $\hat{w}_i$ by an amount proportional to the learning rate $\alpha$ and the ratio of the corresponding input $x_i$ to the sum of all inputs.
After each training iteration, we anneal the learning rate \((\alpha = a^\gamma/a^\beta)\) towards zero by adding a constant value to the denominator \(a^\beta\) (line 17). This decreases the learning rate over time, allowing the system to converge asymptotically towards the correct weight vector rather than oscillating around it. The exact value of the quantity added at each step should not affect the end result, just the size of the next step taken in the weight space.

It is worth noting that our decision to use only non-negative quantities means that all of the elements of the weight vector must be adjusted in the same direction, either all increasing or all decreasing. Furthermore, learning weight values near zero may be problematic as the stored weight approximations must approach zero asymptotically. These are important constraints on the update procedure and the weight space, to which we will return below when discussing simulations of our biochemical circuit design in §5. Standard algorithms to learn linear functions based on gradient descent [60] do not have this restriction, which means that our algorithm may be less efficient than the standard solutions. However, we made these decisions so that our learning design is more amenable to a future implementation in the laboratory using biochemical components, where simplicity is a virtue. In electronic supplementary material, sections S8 and S9, we present results from a direct implementation of algorithm 1, which demonstrate that, in spite of these limitations, this learning algorithm is capable of learning as intended.

4. Biochemical circuit design

In this section, we present our design for a biochemical learning circuit. We begin with some preliminaries on the assumed biochemical reactions, and then present our circuit design, which carries out the primitive operations described in figure 1, to give a biochemical implementation of algorithm 1.

4.1. Preliminaries

We model our biochemical learning machine as a dilute, well-mixed solution of chemical species whose reactions follow the law of mass-action chemical kinetics. In our case, the chemical species in question are assumed to represent either single strands of DNA, or complexes of multiple strands that are held together by sequence-specific binding interactions. While we do not model our system at this level of detail, the key point is that DNA binding can be controlled by designing circuit components with suitable DNA sequences. Therefore, we can transmit information between DNA logic components by designing those components using different DNA sequences that are carefully chosen to interact as desired while preventing unwanted cross-reactivity.

The fundamental building blocks for our biomolecular circuit design are DNAzymes (also known as deoxyribozymes) [41], which are single strands of DNA capable of catalysing a wide variety of chemical reactions [62–64]. We use DNAzymes to form the basis of our proposed chemical implementation because their action is inherently catalytic, so a particular DNAzyme molecule may be used to process multiple input signals without being itself consumed in the process. Furthermore, DNAzymes are made of DNA, which allows a system designer to program which other species they will interact with. These properties make DNAzymes ideal for use in a learning system, where we require a programmable circuit that can process a number of sequential training inputs. We refer the reader to electronic supplementary material, section S1, for more details on DNAzyme chemistry. Our biochemical circuit design builds on our previous experimental work on DNAzyme circuits [65,66] and comprises a number of different kinds of reactions, which are listed below.

4.1.1. DNAzymes cleaving substrates

A DNAzyme Dz can recognize and cleave a specific substrate Siz, producing a product Piz, in the following chemical reaction:

\[
Dz + Siz \rightarrow Dz + Piz.
\]

In practice, the substrate is cleaved into two products, though we will assume that one is an unreactive waste product and just consider a single product Piz, from each cleavage reaction. Since the DNAzyme Dz is unaffected by this reaction, it follows that a single DNAzyme can repeatedly convert substrates to products in a catalytic manner. This is an important property that enables our circuit to process many inputs without itself being consumed. In general, there may be multiple substrates that can be cleaved by a particular DNAzyme Dz. If this is the case, the substrates will compete to bind to, and be cleaved by, the DNAzyme. Assuming that the various substrates are of similar design, it is reasonable to assume that the substrate binding and cleavage rates are the same for all DNAzymes and substrates. This simplifies the model since, by the law of mass-action kinetics, the rates at which the various cleavage products are produced will be proportional to the concentrations of the corresponding substrates.

4.1.2. Conditional activation of DNAzymes

Crucially, the catalytic activity of a DNAzyme can be made conditional on the presence (or absence) of particular DNA strands [25,38]. Here and henceforth, we write \(A \cdot B\) for the complex formed by the binding of species \(A\) and \(B\). We assume that a particular DNAzyme Dz can be inhibited by the addition of an inhibitor strand Iz, which binds to the DNAzyme and produces the catalytically inactive DNAzyme–inhibitor complex Dz · Iz, in the following chemical reaction:

\[
Dz + Iz \rightarrow Dz \cdot Iz.
\]

If an activator strand Az is subsequently added, it can remove the inhibitor strand, producing a waste complex Dz · Iz, and releasing an active DNAzyme Dz back into solution, in the following chemical reaction:

\[
Dz \cdot Iz + Az \rightarrow Dz + Iz \cdot Az.
\]

In previous experimental work, we have implemented this method of control for DNAzymes using DNA strand displacement reactions [65].

4.1.3. Release of effector strands from cleavage reactions

We write \(Siz \triangleright Z\) for a substrate molecule that sequesters a particular species \(Z\), releasing it into solution only after the substrate has been cleaved by the DNAzyme Dz:

\[
Dz + Siz \triangleright Z \rightarrow Dz + Z.
\]

Following the cleavage reaction, the \(Z\) species may interact with other circuit components in solution. This design motif is required to enable information transmission between DNAzymes in our learning circuit. We have demonstrated the implementation of substrate molecules that release a
downstream activator upon cleavage of an upstream DNAzyme, and used this design to implement multi-layer signaling cascades and logic circuits [66], and other groups have implemented substrate molecules with similar behaviour [27,67,68]. These are first steps towards a practical implementation of our learning circuit.

4.1.4. Self-inhibition of DNAzymes
We write $S_{Dz} \triangleright I_{Dz}$ for a substrate in which the cleavage product is an inhibitor $I_{Dz}$, which immediately binds to, and deactivates, a DNAzyme molecule:

$$Dz + S_{Dz} \triangleright I_{Dz} \rightarrow Dz \cdot I_{Dz}.$$ 

This sequesters inhibitors within the substrate structure until the cleavage reaction, protecting them from interacting with free activators in solution. We refer to $S_{Dz} \triangleright I_{Dz}$ as the self-inhibitory substrate for $Dz$, and to $Dz$ as a self-inhibiting DNAzyme. This is a crucial element of our design, as it allows the system to regulate its own catalytic activity over time by returning activated DNAzymes to a sequestered state. Although self-inhibition has not yet been engineered using DNAzymes, it is well-known as a control mechanism within cellular regulatory networks [69].

4.1.5. Multi-stage cleavage reactions
We write $S_{Dz} \triangleright S_{Dz}' \triangleright Z'$ for a substrate molecule that requires two correctly ordered cleavage reactions to occur, involving the $Dz$ and $Dz'$ DNAzymes, for the output species $Z'$ to be released:

$$Dz + S_{Dz} \triangleright S_{Dz}' \triangleright Z' \rightarrow Dz + S_{Dz}' \triangleright Z'.$$

and

$$Dz + S_{Dz} \triangleright Z' \rightarrow Dz' + Z'.$$

We refer to $S_{Dz} \triangleright S_{Dz}' \triangleright Z'$ as a presubstrate for $Dz$, and a substrate for $Dz'$. This design motif is also critical as it will allow us to modify the amount of substrate available to certain DNAzymes by triggering the catalytic activity of other DNAzymes. These reactions have also not yet been engineered using DNAzymes, but similar behaviour has been reported in the context of restriction enzymes [70].

4.2. Circuit design
Assuming the DNAzyme reactions outlined above, we must provide biochemical implementations of the primitive steps needed for the learning algorithm outlined in figure 1. Figure 2 presents an outline of our biochemical learning circuit, which implements the learning primitives described in figure 1. Conceptually, the circuit may be divided into a predictor subcircuit and the feedback subcircuit, connected in a cycle as shown in figure 1 (although we do not enforce this separation in our simulations, in which any of the chemical reactions may occur at any time). The cycle is driven forward by the addition of new training instances.

4.2.1. Multiplier component
Our circuit design relies on a novel, reusable DNAzyme-based multiplier circuit motif, which is shown in figure 2a. The multiplier circuit is based on a DNAzyme $Dz$, which we assume is initially in the inactive state in the $Dz \cdot I_{Dz}$

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Learning circuit design and components. (a) Chemical reaction network for the multiplier component, which consists of three reactions activation, signal generation and self-inhibition. (b) Graphical shorthand for the multiplier component. (c) Learning circuit design for the two-input case, divided into predictor and feedback subcircuits which implement the corresponding primitive operations from figure 1. Circles represent DNAzymes, which are activated by the species shown above them. Blunt-ended arrows denote self-inhibition, and labelled output arrows from DNAzymes show the products with their corresponding weights. The resulting $y$ and $\hat{y}$ species from the predictor subcircuit serve as inputs which trigger the feedback subcircuit. The substrate molecules generated by the feedback subcircuit adjust the stored weights in the predictor subcircuit, thereby completing the feedback cycle.}
\end{figure}
DNAzyme–inhibitor complex. The input to the multiplier is some concentration of the activator species $A_{Dz_i}$, each of which switches a DNAzyme from the catalytically inactive state to the active state. There are two kinds of substrate molecule available for the active $Dz$ DNAzymes to cleave: self-inhibitory substrate molecules $S_{Dz_i} \triangleright I_{Dz_i}$ and signal-producing substrate molecules $S_{Dz_i} \triangleright Z$. The two substrate populations compete to bind to, and be cleaved by, the active DNAzymes. When a substrate $S_{Dz_i} \triangleright Z$ is cleaved, a signal molecule $Z$ is produced, and when a self-inhibitory $S_{Dz_i} \triangleright I_{Dz_i}$ is cleaved, one of the catalytically active $Dz$ DNAzymes is returned to the inactive $Dz - I_{Dz_i}$ state.

We write $[X]$ for the concentration of species $X$. Furthermore, we write $[X]_0$ for the initial concentration of $X$ and $[X]_{ss}$ for the steady-state concentration reached by $X$ as simulation time tends to infinity (assuming it exists). Now, if we consider the initial concentration $[A_{Dz_i}]_0$ of $A_{Dz_i}$ as one input to the multiplier, the total amount of output $Z$ produced before all DNAzymes self-inhibit and the system returns to a quiescent state will be

$$[Z]_{ss} = [A_{Dz_i}]_0 \left[ \frac{S_{Dz_i} \triangleright Z}{S_{Dz_i} \triangleright I_{Dz_i}} \right]_{ss}.$$  

We consider the substrate ratio $w = [S_{Dz_i} \triangleright Z]_{ss}/[S_{Dz_i} \triangleright I_{Dz_i}]_{ss}$ as the second input to the multiplier, which we refer to as the weight, using machine learning terminology (this parameter may also be thought of as the scale factor or gain of the multiplier). Thus, the multiplier may amplify the incoming signal if $w > 1$, copy it if $w = 1$, or attenuate it if $0 \leq w < 1$. In this paper, we do not consider the case where $w < 0$. We have shown (electronic supplementary material, section S4) that substrate ratios are not altered by execution of the multiplier. This is important because the multiplicative factor is encoded as a ratio between the concentrations of substrates that are consumed by circuit operation, and shows that our design can reliably perform a sequence of multiplications. However, the absolute population of substrate molecules is depleted, a point to which we will return in §6.2.

Finally, there is no reason to restrict ourselves to a single signal-producing substrate. With multiple signal-producing substrates competing to be cleaved by a self-inhibiting DNAzyme, we obtain a more general multiplication circuit motif which allows a given input signal to fan out to multiple outputs, each with a different weight applied. Indeed, our learning circuit design will rely on this fact in a number of places. Figure 2b illustrates our graphical shorthand for a general self-inhibiting DNAzyme $Dz$ with multiple output signals $P_{Dz_i}$, each with its own weight $w_i$.

Note that the various output pathways from such a DNAzyme are (partially) coupled because the DNAzyme still only has a single kind of self-inhibitory substrate. The analysis from electronic supplementary material, section S4, holds for multiple signal-producing substrates, because having additional signal-producing substrates simply allows more signal-producing substrates to be cleaved in the time that it takes for the DNAzyme to re-inhibit itself. Therefore, it follows that normal activation of the self-inhibiting DNAzyme does not modify any of the output weights. Furthermore, adding more of a given signal-producing substrate only affects the weight corresponding to that particular output, as the substrate ratios for the other outputs are unaffected. However, adding more of the self-inhibitory substrate changes the weights for all the outputs, because the self-inhibitory substrate concentration is the denominator for all weight ratios—see §4.2.3 for an example of how this issue arises in the feedback subcircuit, and how it can be handled.

4.2.2. Predictor subcircuit

The next stage of our circuit design is to combine several multiplier motifs to produce a linear predictor subcircuit. Figure 2c shows a schematic for the predictor subcircuit component of the system. Training inputs are presented by introducing additional quantities of certain species into the system, such that newly added concentration $[x]$ equals the intended value of the corresponding input $x$, and such that $[y] = f(x)$. For each input signal $x_i$ there is a self-inhibiting DNAzyme $Dz_i$. The DNAzyme $Dz_i$ accepts the corresponding input signal $x_i$ and multiplies it by the current weight approximation $\hat{w}_i$, which our circuit stores as the substrate ratio $\hat{w}_i = [S_{Dz_i} \triangleright I_{Dz_i}]_{ss}/[S_{Dz_i} \triangleright I_{Dz_i}]_{ss}$. The DNAzymes all produce the same output signal $y$; so the overall concentration of $y$ produced is $\sum_{i \in \{1, \ldots, n\}} ([x_i]_{ss} \cdot \hat{w}_i)$. This implements the arithmetic from algorithm 1, line 8. There is also a second output substrate for each DNAzyme, which produces output $K_i$ with weight 1. This has the effect of copying the original concentration of $x_i$ by generating an equal concentration of $K_i$, for reasons which will be discussed below.

The second input required by our predictor design is the $y$ species, whose initial concentration $[y]_0$ represents a threshold against which the computed value of $\hat{y}$ is to be measured. We assume that the $y$ and $\hat{y}$ species react in an annihilation reaction to produce an inert $y \cdot \hat{y}$ complex. After the annihilation reactions finish, we will be left with an excess of $y$ if $y > \hat{y}$ and an excess of $\hat{y}$ if $\hat{y} > y$, and the concentration of the leftover species at that point will equal the difference between $y$ and $\hat{y}$. Hence, these reactions can compute a weighted sum of multiple inputs and classify them against a supplied threshold value, determining which is larger and by how much. This corresponds to the tests made on algorithm 1, lines 12 and 14. The excess of $y$ or $\hat{y}$ will activate the feedback subcircuit, which generates additional substrate molecules for the predictor subcircuit, thereby updating the stored approximations to the weights.

4.2.3. Feedback subcircuit

We have already shown that the weight parameter to our multiplier component can be encoded as a ratio of substrate concentrations, and that this ratio persists across every use of the multiplication component. The final step in the development of our biochemical learning system is to provide a feedback mechanism to adjust the weight parameters stored in the predictor subcircuit according to the output from the predictor. Our reactions will adjust the weights following the approach from algorithm 1, by adding more of the numerator substrate $S_{Dz_i} \triangleright y$ to increase the weight value $\tilde{w}_i$, and by adding more of the denominator substrate $S_{Dz_i} \triangleright I_{Dz_i}$ to decrease the weight value $\tilde{w}_i$ asymptotically towards zero. We achieve this using presubstrates, which were introduced above.

Figure 2d shows a schematic for the feedback subcircuit. The inputs to the feedback subcircuit are the $y$ and $\hat{y}$ signals produced by the predictor subcircuit, as well as the $K_i$ signals which
contain a copy of the initial concentrations of the \( x_i \) input signals. Two distinct feedback routes are shown in figure 2:

1. When \( y > \hat{y} \), the weight parameters need to be increased, and the excess of the \( y \) species from the predictor subcircuit activates the left-hand feedback route, involving the FP, DNAzymes. The self-inhibiting FP DNAzymes cleave the \( S_{FP} > S_{Dy} > \hat{y} \) presubstrates, each of which produces the corresponding \( S_{Dy} > \hat{y} \) signal-producing substrate for the Dz DNAzyme in the predictor subcircuit. The concentrations of these substrates are the numerator of the substrate ratios, causing the weight values to decrease asymptotically towards zero. These reactions correspond to algorithm 1, line 13.

2. The right-hand feedback route is activated by an excess of the \( \hat{y} \) species from the predictor subcircuit, which means that \( \hat{y} > y \) and hence that the stored weight parameters need to be reduced. The \( \hat{y} \) species activate a feedback path using the FN, DNAzymes, which cleave \( S_{FN} > S_{Dy} > I_{Dy} \) presubstrates to produce additional self-inhibiting \( S_{Dy} > I_{Dy} \) substrates for the DNAzymes from the predictor subcircuit. These increase the denominators of the substrate ratios, causing the weight values to decrease asymptotically towards zero. These reactions correspond to algorithm 1, line 15. There is an additional subtlety in this feedback route: when additional self-inhibiting substrates are released for the DNAzymes in the predictor subcircuit, the same amounts of the substrates \( S_{Dy} > K_i \) must also be released. This is necessary to ensure that the weights of the reactions in the predictor subcircuit which produce the \( K_i \) species remain at 1, so that the concentrations of the input species can still be copied accurately. We achieve this by using the same initial concentrations of the presubstrates \( S_{FN} > S_{Dy} > I_{Dy} \) and \( S_{FN} > S_{Dy} > I_{Dy} \) and by maintaining the same weights on the corresponding arcs.

In both feedback routes, each of the feedback DNAzymes (FP, and FN) is activated in proportion to the magnitude of the corresponding input signal (\( x_i \)). This is in accordance with the learning algorithm presented in algorithm 1 and is intended to activate the feedback DNAzymes such that the weight vector is modified in approximately the correct direction. In the biochemical implementation, this takes place using the \( K_i \) species, which take on the same concentration as the initial concentration of the \( x_i \) species by the execution of the reactions in the predictor subcircuit. In the initial reactions of the feedback subcircuit, the \( K_i \) species catalyse the conversion of the \( y \) and \( \hat{y} \) species into the activator species for the FP, and FN, feedback DNAzymes, respectively. This is achieved in the model by naming the \( K_i \) species as both reactants and products of these reactions. The competition between the \( K_i \) species to catalyse the conversions of these species ensures that the feedback DNAzymes are activated in the desired proportions. The weights associated with the feedback DNAzymes can be interpreted as the learning rate of the system, as they control how much feedback is produced per unit error in the prediction from the predictor subcircuit.

4.2.4. Training protocol
In our simulations, training inputs are modelled by mixing events. As described above, each mixing event increases the concentrations of species \( x_i \) and \( y \) in the system such that the newly added concentration \([x_i]\) equals the intended value of the corresponding input \( x_i \), and such that \([y] = f(x)\). When the simulated system has reached equilibrium (i.e. when the predictor subcircuit has processed the inputs and the feedback subcircuit has updated the weight approximations) further quantities of the \( x_i \) and \( y \) species which encode the next training instance are added, and so on. The addition of reagents from outside drives the system forward round the feedback cycle.

When all training inputs have been added, we enter a testing phase where the performance of the machine is evaluated. The feedback subcircuit is disabled by setting the concentration of the \( S_{Dy} > K_i \) and \( S_{FN} > S_{Dy} > K_i \) species to zero, which prevents the stored weight approximations being modified further. As in the training phase, testing instances are also provided as concentrations of the \( x_i \) and \( y \) species, except we observe the excess of \( y \) or \( \hat{y} \) to assess the accuracy of the predictions. Since the feedback subcircuit is disabled in the testing phase, the \( y \), \( \hat{y} \) and \( K_i \) species, which are the outputs of the predictor subcircuit, are no longer used by the feedback circuit, so we assume that these are set to zero before each addition of subsequent testing inputs.

5. Results from learning simulations
We simulated the behaviour of our learning circuit by numerically integrating an ODE model of our biochemical learning system, instantiated for the appropriate number of inputs in each case. The ODE model is presented in the electronic supplementary material, table S1, which includes rate constants that correspond to the separations of timescales assumed for different reactions in the biochemical model. The numerical integration process produced time courses of the species concentrations between successive mixing events, which were modelled by simply adjusting the relevant species concentrations at the appropriate point in time. Numerical integration was performed using a stiff ODE solver from COPASI [71], in conjunction with an external Python wrapper that allowed us to define multiple mixing events and to call the solver repeatedly in an iterative simulation loop, to obtain a trace of species concentrations across an entire training and testing run.

5.1. Asymptotic learning performance
We began by studying the two-input case, where the weight vector \( W \) is just a pair \((w_x, w_y)\) of weights drawn from [0, 10], and the input vector is also a pair of values drawn from [0, 500]. We analysed the effect of increasing the amount of training data on the learning capabilities of our biochemical circuit, by taking 100 uniformly distributed starting and target weight vectors in the \([0, 10]\) weight space and training the circuit, by taking 100 uniformly distributed starting and target weight vectors in the \([0, 500]\). We analysed the effect of increasing the amount of training data on the learning capabilities of our biochemical circuit, by taking 100 uniformly distributed starting and target weight vectors in the \([0, 10]\) weight space and training the system using 0 and 1000 random training instances. We evaluated each trained system using 100 testing instances and plotted the average RMSE across the 100 experiments, and the results are shown in figure 3 (shaded area denotes standard deviation). We deduce that the system does improve its current approximation to the true weight vector given additional training data, however, we note that the system does not reach zero RMSE even after 1000 training rounds. This suggests that there may be some bias in the system, making some weight vectors harder to learn than others. Therefore, we undertook a systematic investigation of the learning performance of our system across a range of weight values.
5.2. Weight-space performance profiles

We ran a learning simulation for all possible starting and target weight pairs lying on a square grid of points with spacing 0.5. In each case, 1000 training instances were provided to the system, followed by 100 testing instances, and the root mean square error (RMSE) from these 100 tests was plotted as the colour of each point on the scatter plot. In the following discussion, we will use $(-, +)$, $(+, +)$, $(-, -)$ and $(+, -)$ to indicate the four directions that point towards the top-left, top-right, bottom-left and bottom-right corners of a standard two-dimensional plot, respectively.

Figure 4 presents results from simulations which attempt to learn every target weight pair on the grid starting from the initial weights $(2, 2)$, $(8, 8)$, $(2, 2)$ and $(8, 2)$ (see the electronic supplementary material for more data). We observe that the system can learn many weight vectors on the grid from the four specified starting weight vectors well, as evidenced by the areas of low RMSE. However, there are anomalies towards the $(-, +)$ and $(+, -)$ corners of the plot where the RMSE is significantly higher.

We hypothesized that these areas of higher RMSE were related to certain aspects of our circuit design. Since weights cannot go below zero, they must decrease asymptotically towards zero, which causes slower convergence towards small weight values. Furthermore, since the feedback subcircuit must either activate all the FP, DNAzymes or all the FN, DNAzymes, and not a mixture of the two, it follows that in a single learning step the weights must either all increase or all decrease. Thus, it is easier for the learning machine to move towards $(+, +)$ or $(-, -)$ in the two-dimensional weight space than it is to move towards $(+, -)$ or $(-, +)$. To move towards $(+, +)$ or $(+, -)$, the system must use zigzagging steps in the other two directions, which is a less efficient trajectory. A combination of these factors would account for the high RMSE values in the $(-, +)$ and $(+, -)$ corners and the low RMSE values in the vicinity of the origin ($w_1 = w_2 = 0$).

To confirm these hypotheses, we observed the behaviour of the simulated learning machine moving in different directions through the weight space. Figure 4b(i) shows selected weight-space trajectories from 1000-step learning simulations in the $[0, 10]$ weight space (the ideal trajectory would be a straight line from the start to the target in each case). As predicted, the trajectory moving towards the $(-, +)$ corner is a zigzag pattern of short steps in the $(+, +)$ and $(-, -)$ directions, which peter out before reaching the target weight values. This contrasts with the trajectory moving towards the $(+, +)$ corner of the weight space, which is far more direct and is therefore able to home in on the target weight values within the allotted 1000 training instances.

5.3. Experiments with alternative weights

We addressed these issues by considering an alternative set of possible weight values: we chose the weight values in the target function from the interval $[10, 20]$ as opposed to $[0, 10]$. Apart from this change, the rest of the learning circuit is unchanged. Figure 4b(ii) shows sample weight trajectory moving in the $(-, +)$ and $(+, +)$ directions in the alternative $[10, 20]$ weight space. In both cases, the system converged very close to the target weight values within 1000 training cycles. Even when the weights had to be adjusted in the $(-, +)$ direction, the larger absolute values allowed larger steps to be taken, which made the zigzag trajectory more efficient than the corresponding trajectory in figure 4b(i).

Figure 5 presents results from learning simulations across the two-dimensional $[10, 20]$ weight space, this time using a square grid of resolution 1 (the higher resolution 0.5 grid spacing was not necessary here). We attempted to train the learning machine starting from the initial weight pairs...
(12, 12), (18, 18), (12, 12) and (18, 12) to learn every weight pair on the grid and observed a far more uniform RMSE distribution with far lower individual RMSE values. (Additional data are presented in electronic supplementary material, section S6, including additional plots for the [10, 20] weight space using a smaller scale on the colour bar.) Thus, we conclude that switching from the [0, 10] weight space to the [10, 20] weight space offers better, and more uniform, learning performance, in two ways: (i) by enabling more efficient trajectories towards the (−, +) and (+, −) corners of the weight space and (ii) by enabling faster convergence to smaller weight values as they are no longer close to zero.

Figure 6 illustrates the different error profiles between the [0, 10] and [10, 20] weight spaces by plotting the histogram of all RMSE values obtained from our weight-space scanning simulations. The RMSE histogram for the [0, 10] weight space shows a large peak at low RMSE values but also a long tail which comprises a significant number of instances with larger errors, up to a maximum between 350 and 400, whereas the histogram for the [10, 20] weight space has all RMSE values distributed between 0 and the maximum RMSE value of around 7.5. This confirms that the [10, 20] weight space is a more preferable operating regime for our biochemical learning circuit than the [0, 10] weight space, since it allows far more uniform learning performance.

Finally, figure 5b directly compares the asymptotic learning performance between the weight spaces by plotting a learning curve for the [10, 20] weight space and comparing it with the learning curve for [0, 10] from figure 3. After 1000 training instances, the system working in the [10, 20] weight space has an average RMSE about one order of magnitude lower than for the [0, 10] weight space, and with a much lower variance. We deduce that the asymptotic learning performance in the [10, 20] weight space is superior to that in the [0, 10] weight space. This suggests that learning functions in which one or more weights are close to zero using our learning circuit may be challenging. However, we note that weights close to zero will contribute relatively little to the overall result—therefore, it may be possible to omit those terms entirely without significantly altering the target function.

5.4. Learning functions with more than two inputs
To assess the performance of the system when learning more than two weights, we ran experiments similar to those from figure 5b for systems with five and 10 inputs, using the more favourable [10, 20] weight space. This was achieved by replicating our learning circuit design to process additional inputs. The resulting five-input and 10-input learning curves are shown in figure 7a, with the two-input learning curve from figure 5b included for comparison. We observe that the five-input and 10-input variants of the system are capable of learning, as evidenced by the decrease in RMSE with extra training data. The rate of learning for the higher dimensional systems is lower than that we observed in the two-input case, but this may be because we used the same learning rate and other parameters for all of the experiments, which one might expect to produce slower convergence in the systems with more inputs. It is possible that adjusting the learning rate could lead to faster convergence in the higher dimensional cases, but such optimization is beyond the scope of the current paper. Furthermore, the issue with zigzag learning trajectories discussed above is likely to be even more problematic in higher dimensions: we are still restricted to either increase all weights or decrease all weights in a single step, meaning that a greater fraction of the potential step directions are excluded than in the two-dimensional case. We conclude that the design of our system can be extended to learn linear functions with more than two inputs, with a linear increase in the numbers of species and reactions (see the electronic supplementary material, table S1).

5.5. Robustness to noise
Noise is a crucial issue in all biomolecular systems. In an experimental implementation of our learning circuit, the addition of training inputs would be a physical procedure conducted by a human experimenter and therefore prone to experimental error. We studied the effect of increasing amounts of noise in simulations of the two-input learning machine by modifying the training instances so that the concentration $[y]$ of the species added to represent the true output value is given by $[y] = f\left([X]\right) + N$ where the noise term $N$ was drawn from a Gaussian distribution with mean 0 and standard deviation $\sigma$. We retain our previous ODE to study noise in the input signals, because we assume that the
Figure 6. (a,b) RMSE histograms from the full [0, 10] (original) and [10, 20] (alternative) weight space scans. Note that there are more samples in the [0, 10] histogram, as we ran a finer weight-space scan with resolution 0.5 over the [0, 10] weight space compared with a coarser resolution of 1.0 in the [10, 20] weight space. Furthermore, the x-axis for the [10, 20] histogram uses a more fine-grained bin size to highlight the far narrower distribution of RMSE values in the [10, 20] weight space.

Figure 7. Additional learning curves. (a) Comparison of learning curves for simulated two-input, five-input and 10-input learning circuit, all in the [10, 20] weight space. The lines denote average RMSE values from 100 trials with random different starting and target weight vectors, and the shaded areas denote 1 standard deviation (s.d.) above and below the mean. The y-axis plots the average RMSE value scaled to the size of the spaces of possible output values, by dividing the average RMSE by the square root of the maximum prediction size for d dimensions, $P_{\text{max}}(d) = W_{\text{max}} \times \chi_{\text{max}} \times d$, where $W_{\text{max}}$ is the upper bound of the given weight space and $\chi_{\text{max}}$ is the largest possible input value used in the experiments. (b) Learning curves with noise, represented by the standard deviation $\sigma$ for a Gaussian noise distribution with mean 0 which was superimposed on the value of $y$ in the training instances supplied to the simulation. Average RMSE values for 100 traces are plotted, with predictive performance evaluated over a common set of 100 testing instances. The noise-free ($\sigma = 0$) learning curve from figure 5b for the [10, 20] weight space is included for comparison.

species concentrations in the biochemical model are sufficiently high that stochasticity in the individual species reactions can be neglected.

Learning curves for different values of $\sigma$, with weights drawn from the [10, 20] weight space, are presented in figure 7b (see the electronic supplementary material for additional data). We see that the circuit is capable of learning in the face of noise, with a limit on the predictive accuracy related to the standard deviation $\sigma$ of the noise term, and for $\sigma < 100$ we conclude that useful learning can indeed take place. In reality, all noise would not manifest itself in a single parameter in this way, but we believe that this noise model is still of practical relevance because the linear combination of multiple independent, normally distributed random variables also has a normal distribution. Therefore, any Gaussian noise in our input signals can be consolidated into a Gaussian noise term on the predicted output, which can then be combined with any Gaussian noise on the supplied true output value to produce a single Gaussian noise term on the true output.

6. Discussion

In summary, we have presented a simple algorithm for learning certain linear functions and demonstrated a biochemical learning circuit which implements this algorithm, based on the chemical reactions of DNAzymes. We have demonstrated that our circuit can indeed learn target functions, given enough training data, and presented a detailed investigation of the learning performance of our system for different ranges of weight values, with increasing numbers of input signals, and in the face of noise in the training data. Our work suggests that classical machine learning algorithms may not necessarily be the best fit in a biochemical implementation framework, which has implications beyond the specific system studied.
here. As a further example, it is known that competitive learning in winner-takes-all neural networks can be implemented very concisely using molecular computing architectures, because direct competition for catalytic resources can replace mutually inhibitory connections between all pairs of neurons [51, 72]. Thus, the study of biochemical learning devices may be of philosophical, as well as practical, interest.

6.1. Correctness of the learning algorithm and its biochemical implementation

To verify that the learning procedure from algorithm 1 is correctly implemented by our ODE simulations of the biochemical learning circuit, we directly implemented algorithm 1 and compared the results of the algorithm with the biochemical implementation. These data are presented in electronic supplementary material, section S9 and show good agreement between the direct implementation of the algorithm and the biochemical implementation. This suggests that our biochemical system is a reasonable implementation of our intended learning algorithm, and that the learning behaviour derives from the algorithm itself as opposed to some quirk of the biochemical implementation. The latter point is further supported by the data presented in electronic supplementary material, section S8.

While a detailed theoretical analysis of algorithm 1 is outside the scope of the current paper, this may be of interest for future work, because we have shown that a desire for simplicity in the biochemical implementation can impose constraints on the capabilities of the learning algorithm. Furthermore, it is worth noting that formally proving correctness of the biochemical implementation itself would be extremely challenging due to the inherent massive parallelism. So far, formal verification has only been attempted for biochemical systems with relatively small numbers of each molecular species [73,74], owing to the combinatorial explosion in the size of the state space as the numbers of copies increase. However, recent work on modular verification of DNA strand displacement devices [75] may offer a solution to this problem.

6.2. Practical considerations

The learning circuit simulated in this paper is designed around plausible DNAzyme reactions, offering a route to a future experimental implementation of our design. In previous experimental work, we have implemented DNAzyme activation using toehold-mediated strand displacement reactions [65] and DNAzyme signalling cascades using structured substrate molecules that release a downstream activator when cleaved by an upstream DNAzyme [66]. These important steps are directly relevant to the experimental realization of our learning circuit design. However, further advances in DNAzyme chemistry are needed before the entire circuit design can be implemented in the laboratory, in particular, the design of presubstrate molecules that require multiple cleavage reactions to release the sequestered sequence.

An experimental implementation would also need to work with realistic rate constants. The biochemical model presented in the electronic supplementary material, table S1 includes rate constants that were chosen to correspond with the expected separation of timescales in the specified reactions: we assume that annihilation reactions are the fastest reactions, that activation reactions (e.g. by DNA strand displacement) are the next fastest and that DNAzyme-catalysed cleavage reactions are the slowest. The assumption that annihilation is fast is required for correct computation of the difference operation, and this is chemically plausible if $y$ and $\dot{y}$ are fully complementary single strands of DNA, for example. Previous work [76,77] has shown that the kinetics of activation via toehold-mediated strand displacement can be controlled by controlling parameters such as toehold length and internal spacer length. The kinetics of DNAzyme-based reactions can also be controlled by adjusting the lengths of the substrate binding arms of the DNAzyme, or by using a different catalytic motif, e.g. the E6 motif [62] has slower cleavage kinetics than the 8–17 or 10–23 motifs [63]. Hence, there are possible methods to optimize the performance of real-world implementations to satisfy our assumptions regarding the kinetics of different reactions in the learning circuit.

Furthermore, the system would be restricted to small numbers of weight parameters and short training schedules due to the practicalities of experimental work. In an experimental setting, the ‘resetting’ of the system between the testing and training phases described in §4.2.4 might be achieved by a mechanical separation technique such as washing or centrifugation or by chemical means, e.g. light-triggered cleavage of strands that incorporate photo-cleavable backbone modifications. A practical implementation would encounter additional sources of uncertainty beyond those investigated in figure 7b, such as pipetting errors, non-specific activation of circuit components, and DNA synthesis errors, which would reduce the accuracy of the learning computations. Nonetheless, we believe that the results in figure 7b show promise for the learning circuit to overcome these more realistic noise scenarios. Furthermore, our learning circuit can only function correctly with an adequate supply of substrates, which places an upper bound on the amount of time it can spend learning. Although our simulations used up to 1000 training instances to enable rigorous characterization of the learning performance, the learning curves from figure 5b show that good predictive performance can be achieved using far fewer training instances. We refer the reader to electronic supplementary material, section S5 for an extended discussion on the effects of substrate depletion on circuit operation.

6.3. Future research directions

The design of our biochemical learning circuit could be extended in a number of directions, e.g. to learn nonlinear functions such as a classical perceptron [78], to handle negative values as in the artificial chemistry approach of Banda et al. [48], and to connect multiple learners to implement a trainable artificial neural network that could learn more complex functions. Another fruitful direction would be to explore concurrent biochemical implementations of parallel learning algorithms, as this could take full advantage of the massively parallel nature of biochemical interactions in solution.

In a future theoretical paper, we will fully explore the nature of the learning algorithm that we have realized in artificial chemistry in this paper. We have demonstrated a working combination of reaction rates in this paper; in future work, we will explore the parameter space more fully to ascertain how robust the system is to variance in these parameters. This analysis may be made more challenging by the presence of nonlinearities inherited from the biochemical substrate of our learning circuit.
On a practical level, experimental implementations of reusable, self-inhibitory DNAzyme circuit components would be a major step forward in the development of biomolecular computing systems. This would open up new research possibilities in a number of areas, such as adaptive DNA-based pharmaceuticals and continuous monitoring and control of industrial processes.

Funding statement. This material is based upon work supported by the National Science Foundation under grant nos. CDA-1027877, CDA-1028238 and CCF-1318833. A.M. is partially supported by a National Science Foundation Graduate Research Fellowship under grant no. DGE-0237002. M.R.L. is partially supported by a New Mexico Cancer Nanoscience and Microsystems Training Center Postdoctoral Fellowship (NIH/NCI grant no. R25CA153825).

References


