Computing With Chemicals: Perceptrons Using Mixtures of Small Molecules

Christopher Rose, Sherief Reda, Brenda Rubenstein and Jacob Rosenstein
Brown University
Providence, RI
Email: Christopher_Rose, Sherief_Reda, Brenda_Rubenstein and Jacob_Rosenstein@brown.edu

***Abstract*—Computation that can exploit the Avogadrian numbers of molecules in heterogeneous solutions, and the even larger number of potential interactions among these molecules, is a tantalizing dream. However, the lack of precise specificity/control of chemical interactions can be at odds with the dream. In this paper, we show how relatively simple chemistry can be used to produce a ubiquitous computational primitive (the multiply-accumulate or MAC operation) that forms the basis for a single-layer neural network called a perceptron. A chemical perceptron can be realized using distinct mixtures as inputs and different reagents as operations to produce the results of the perceptron MAC operation, that can be read out perhaps using simple indicators such as pH or fluorescence. With a moderately large chemical library, the number of potential inputs can be Avogadrian so that reagent addition implicitly performs a concomitantly large number of MAC operations in parallel.**

I. Introduction

Computation that can exploit the Avogadrian numbers of heterogeneous molecules present even in small volumes of chemical mixtures and the even larger number of potentially unique interactions among these molecules in gaseous and/or liquid states, is a tantalizing dream. In principle, molecules hold the promise of realizing levels of parallelism orders of magnitude beyond what is currently achievable *in silico*, while requiring substantially less energy [1]. They also lie at the heart of the unmistakably powerful biochemical computing that our bodies perform everyday. However, the lack of precise specificity/control over chemical interactions in a mixture can be at odds with this dream. Molecules in solution can react in a stochastic fashion largely dictated by diffusion that is highly dependent not only upon ambient conditions, but also upon what other molecules and reagents are present. It is this inherent non-linearity that has established chemistry as the challenging field that is over the past few centuries, yet makes chemical computation so tantalizing. In this paper, we show how relatively simple chemistry can be used to produce a ubiquitous computational primitive (the multiply-accumulate or MAC operation) that forms the basis for a single-layer neural network called a perceptron. Our perceptron is produced using distinct chemical mixtures whose inclusion/exclusion in a pool is controlled by a binary input vector, applying different reagents/processes to the pooled inputs, and reading the result through simple indicators such acidity, alkalinity, or fluorescence emission. And while not Avogadrian, the number of potential inputs can be very large and reagent addition implicitly performs a concomitantly large number of multiply accumulate operations in parallel.

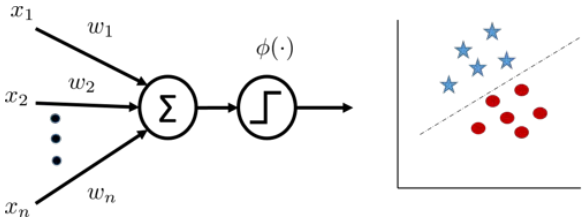

Fig. 1. Perceptron classifier with inputs x_j and weights w_j, $j = 1, 2, \cdots, J$ and output nonlinearity $\phi()$.

The organization of this paper is as as follows. We will first review perceptron structure in Section II and briefly describe the structure of the individual small molecules upon which the scheme is built in Section III. Then in Section IV, based upon the presence/absence of small molecule mixtures as inputs, we will describe how, for a given desired perceptron weight set, mixture compositions can be found such that their collective interaction with different reagents produces the desired perceptrons and the desired indicator outputs.

II. The Perceptron

Our overarching goal is to devise computational systems that can perform classification and signal processing on massive data sets by exploiting the inherent parallelism of solution phase chemical reactions. Our target architecture is the *perceptron*, which is a flexible and universal pattern classification structure, and a key element of neural network systems that handle large data sets [2]. The “perceptron," consists of a single multiply-accumulate primitive, followed by a nonlinearity, as illustrated in FIGURE 1. More complex neural networks can be built from multiple layers of perceptrons.

Mathematically, a perceptron is simply a dot product between a real “input" vector $\mathbf{x}$ and a real “weight" vector $\mathbf{w}$, both of dimension J, followed by a nonlinearity $\phi(\mathbf{w} \cdot \mathbf{x})$ where $\phi(\cdot)$ is a “sigmoidal," nondecreasing thresholding function with a rapid transition between two discrete levels. The perceptron is a binary classifier which separates inputs into two groups. For example, a perceptron can be designed to classify an MNIST handwritten digit image into one of two groups such as "zero" vs. "not a zero" [3]. By controlling the values of the weights, one can change the classification operation. One

978-1-5386-4780-6/18/$31.00©2018 IEEE

advantage of using perceptron-based computation is that the weights can be flexibly designed; another advantage is that the computations are error tolerant. For instance, the precision of the weights can be flexibly set in floating point, fixed point, or even binary ($-1|1$ or $0|1$), but nevertheless, the classification outcome can still be accurate because the training process can compensate for the reduction in precision and "heal" the perceptron [2], [4]–[7]. Such flexibility and error tolerance are crucial given the underlying discrete nature of the input-coding we propose and the stochastic nature of chemical reactions.

III. Perceptrons Through Differential Reactivity

Over billions of years, biological systems have evolved "lock and key" receptor/ligand complexes, in which a given ligand interacts with a specific receptor structure, often with extremely high specificity. Impressive examples include restriction enzymes that attach to a specific sequence of bases in DNA and ion channels that select for a single ionic species [8].

In contrast, *de novo* design of exquisitely specific reactions which will select for (each of) one and only one molecule remains an outstanding challenge for synthetic chemistry [9], [10]. Thus, the notion of selecting individual molecules from a mixture to react without affecting other molecules is untenable – which in turn means the notion of mapping individual molecules to inputs in the perceptron model is untenable. So, developing an approach to realizing perceptrons (and thence more complex neural networks) which does not depend on carefully matched ligand/receptor chemistry is necessary.

Thus, instead of using the presence or absence of distinct *molecules* as perceptron inputs, we will use distinct *mixtures* of molecules as units of input and combine these mixtures as dictated by the binary input vector $\mathbf{x}$ – resulting in a mixture of mixtures that we call a pool. We then apply some reagent (or process) to the pool to perform the necessary multiply-accumulate operations, and follow with threshold detection of a detectable reaction product to produce the perceptron output.

Our individual molecules will be composed of different "R-groups" – molecular species with different chemical properties that can be attached to some chemical backbone [11]. The number of R-groups comprising a molecule will be some fixed K, and while it is possible that the chemistry will allow multiple copies of the same R-group per molecule, we may invoke restrictions such that a given R-group can appear only once per molecule. The multiplicity of potential R-groups and the size of K implies that the number of different molecules is combinatorial and thus could be very large.

We take as a given that a specific reagent will interact differently with different R-group types [11]. For example, suppose reagent G_i reacts with R-group $\mathcal{R}_n$ and replaces $\mathcal{R}_n$ with a different detectable R-group (that may be a fluorophore or have a different acidity, polarity, or charge) with a given yield. Then, G_i is also assumed to affect other R-groups with varying degrees of specificity/affinity. If we assume that the *reagent is plentiful* and thereby accessible to all molecules in a mixture given sufficient mixing and/or diffusion, then we can define an operator $\mathcal{G}_i(\cdot)$ where $\mathcal{G}_i(\mathcal{R}_n) = \gamma_{in} \in [0,1]$ is the equilibrium proportion of R-group $\mathcal{R}_n$ replaced by indicator T_i under application of reagent G_i. Given N different R-groups, any given reagent G_i has an associated R-group replacement "equilibrium/yield vector" $\boldsymbol{\gamma}_i$ defined as

$$\boldsymbol{\gamma}_i = \begin{bmatrix} \gamma_{i1} \\ \vdots \\ \gamma_{iN} \end{bmatrix}.$$

Now, were it possible to design a reagent G_i that would react *only* with R-group $\mathcal{R}_n$, always replacing it completely by T_i, then $\boldsymbol{\gamma}_i = e_n$, the canonical unit vector in $\Re^N$. However, such a scenario is unlikely owing to the lack of precise ligand/receptor specificity between R-groups and reagents, so the replacement vector associated with reagent G_i will contain an assortment of non-negative numbers between zero and one. Further, we will we also admit the possibility that $\boldsymbol{\gamma}$ may be stochastic (with known statistics). Our $\boldsymbol{\gamma}$ is defined assuming that all of the related reactions have achieved equilibrium: while $\boldsymbol{\gamma}$ will vary with time as reactions proceed, we only consider final $\boldsymbol{\gamma}$ values here.

Now, consider molecules, $\{\boldsymbol{\mu}_m\}$, composed of K R-groups $\mathcal{R}_{m_1}, \mathcal{R}_{m_2}, \cdots, \mathcal{R}_{m_K}$. We can represent each molecule $\boldsymbol{\mu}_m$ as an integer N-vector where nonzero entries in position ℓ indicate the number of $\mathcal{R}_\ell$ contained in $\boldsymbol{\mu}_m$. For molecules constructed of K R-groups, we must have $||\boldsymbol{\mu}_m||_1 = K$, but the structure of $\boldsymbol{\mu}_m$ may be further constrained depending upon how molecules are constructed. For instance, perhaps a given R-group can appear only once per molecule so that $\boldsymbol{\mu}_m$ is always a binary vector. Further, we might also have K distinct classes of R-groups where each class may be represented only once. However, for now we simply assume the entries are non-negative integers that sum to K. We can then define an $N \times M$ molecule matrix, $\mathbf{U}$, as

$$\mathbf{U} = \begin{bmatrix} \boldsymbol{\mu}_1 & \boldsymbol{\mu}_2 & \cdots & \boldsymbol{\mu}_M \end{bmatrix} \tag{1}$$

which contains our "universe" of M distinguishable molecules, $m_1, \cdots, m_M$.

Now, let each component of a particular group of molecular *mixtures*, $\{\mathcal{M}_j\}$, $j = 1, \cdots, J$, be defined by a binary M-vector $\boldsymbol{\theta}_j$ whose components are $0|1$, corresponding to which molecules are absent|present in $\mathcal{M}_j$ (at presumed unit concentration). We can then define a matrix $\boldsymbol{\theta}$, each column of which specifies the molecules that comprise input j as

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \cdots & \boldsymbol{\theta}_J \end{bmatrix} \tag{2}$$

where $J \leq 2^M$ is the number of inputs to our perceptron. That is, mixture j is present in the pool if perceptron input $x_j = 1$ and is not present if $x_j = 0$.

Now, since reagents act on R-groups, to obtain the results of reagent application we must translate $\boldsymbol{\Theta}$ into a corresponding collection of R-group mixtures, $\{\mathbf{r}_j\}$, upon whose components the $\boldsymbol{\gamma}_{in}$ can operate. To this end, we define the $N \times J$ matrix $\mathbf{R}$ as

$$\mathbf{R} = \mathbf{U}\boldsymbol{\Theta} \tag{3}$$

We note that

$$\mathbf{R} = \left[\begin{array}{ccc} \mathbf{r}_1 & \cdots & \mathbf{r}_J \end{array} \right]$$

and

$$\mathbf{r}_j = \left[\begin{array}{c} r_{j1} \\ \vdots \\ r_{jN} \end{array} \right]$$

where the integer r_{jn} indicates the number of $\mathcal{R}_n$'s in mixture j. Since the action of reagent G_i on molecule $\boldsymbol{\mu}_m$ is $\boldsymbol{\gamma}_i^\top \boldsymbol{\mu}_m$, it is easy to see that application of reagent G_i to a given mixture $\mathcal{M}_j$ results in *non-negative* indicator "weight"

$$w_{ij} = \boldsymbol{\gamma}_i^\top \mathbf{U} \boldsymbol{\theta}_j$$

Finally, if the presence or absence of $\mathcal{M}_j$ in the pool is defined by the binary variable x_j, the indicator amount produced by applying reagent G_i to the pool is

$$||T_i|| = \sum_{j=1}^{J} w_{ij} x_j = \mathbf{w}_i^\top \mathbf{x} = \boldsymbol{\gamma}_i^\top \mathbf{U} \boldsymbol{\Theta} \mathbf{x} \tag{4}$$

where

$$\mathbf{w}_i = \left[\begin{array}{c} w_{i1} \\ \vdots \\ w_{iJ} \end{array} \right]$$

and each $w_{ij} \geq 0$. We then have

$$\mathbf{w}_i = \boldsymbol{\Theta}^\top \mathbf{U}^\top \boldsymbol{\gamma}_i \tag{5}$$

We summarize these results as a theorem:

*Theorem 1 (***Chemical Perceptron):** Assuming the presence or absence of each of an ensemble of molecular mixtures as binary inputs, x_j, the scalar $||T_i||$ as given in equation (4) represents a chemical multiply-accumulate operation on the x_j using weights $w_{ij} \geq 0$. Applying a threshold operator $\phi()$ to $||T_i||$ results in the mathematical equivalent of the perceptron structure shown in FIGURE 1.

Proof: [Theorem 1] See the mathematical development leading to the statement of Theorem 1. •

In FIGURE 2, we provide a cartoon representation of the chemical perceptron described in Theorem 1. We now consider the appropriate design of input mixtures to realize different perceptron weight-sets under the action of different reagents.

IV. Designing the Input Composition, $\boldsymbol{\Theta}$

Assuming R-groups, molecules, and reagents have already been chosen, we are then presented with at least two mathematical problems:

1) How do we choose input mixture compositions to produce some desired weight pattern $\mathbf{w}_1$?
2) Since the ensemble of input mixture compositions will represent real data, is it possible to choose a single input mixture set where application of reagent G_p, $p = 1, 2, \cdots, P$ represent P different perceptrons with weight sets $\mathbf{w}_p$, $p = 1, 2, \cdots, P$?

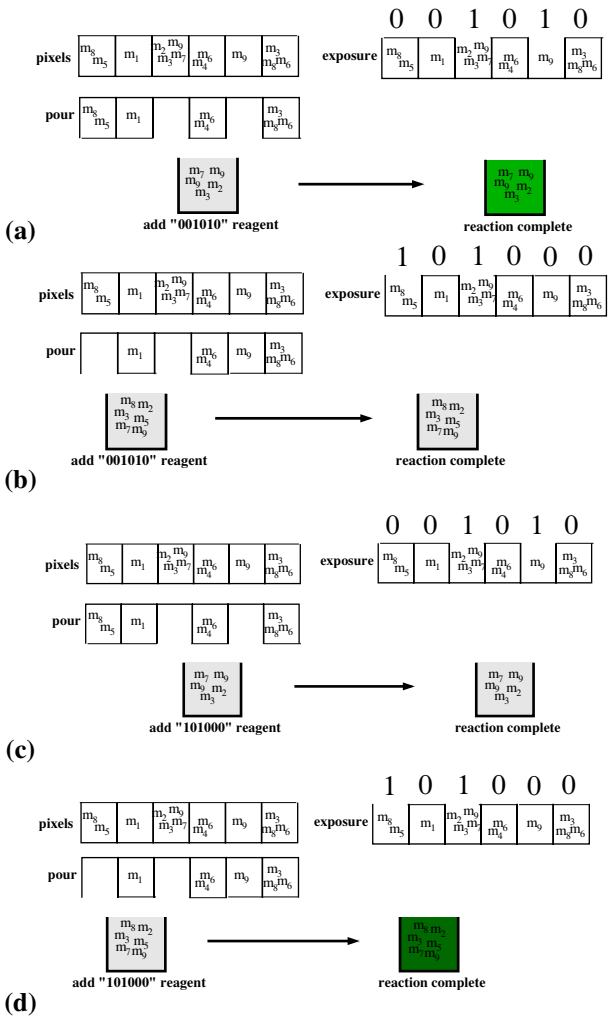

Fig. 2. Representation of a chemical perceptron that can recognize two patterns (001010 and 10100) corresponding to two different reagents. Pixel cells are mixtures composed of different molecules m_ℓ. Pixels are exposed according to the inputs, x_j, $j = 1, 2, \cdots, 6$, and then "poured" into the pool. Reagent is added to the pool and reaction proceeds. Green(darker) represents post-reaction above-threshold indicator detection. **(a)** Application of reagent 001010 with input 001010; **(b)** Application of reagent 101000 with input 001010; **(c)** Application of reagent 101000 with input 001010; **(c)** Application of reagent 101000 with input 101000;

Let weight set, $\mathbf{w}_i$, corresponding to application of reagent G_i be

$$\boldsymbol{\gamma}_i^\top \mathbf{R} = \boldsymbol{\gamma}_i^\top \mathbf{U} \boldsymbol{\Theta} = \left[\begin{array}{ccc} w_{i1} & \cdots & w_{iJ} \end{array} \right] = \mathbf{w}_i^\top$$

Now, let

$$\boldsymbol{\Gamma} = \left[\begin{array}{ccc} \boldsymbol{\gamma}_1 & \cdots & \boldsymbol{\gamma}_P \end{array} \right]$$

where P is the number of different perceptrons we require of a given data set corresponding to the set of input mixtures, $\mathcal{M}_1$, $j = 1, \cdots, J$. We then have

$$\boldsymbol{\Theta}^\top \mathbf{U}^\top \boldsymbol{\Gamma} = \mathbf{R}^\top \boldsymbol{\Gamma} = \left[\begin{array}{ccc} \mathbf{w}_1 & \cdots & \mathbf{w}_P \end{array} \right] \equiv \mathbf{W} \tag{6}$$

and we must solve for $\boldsymbol{\Theta}$.

However, before considering specific approaches, some useful general observations can be made from the structure of

equation (6). Notice that if any column of $\mathbf{\Gamma}$, $\boldsymbol{\gamma}_q$, is linearly dependent on another set of columns $\{\boldsymbol{\gamma}_\ell\}$, there exists a set of constants $\{\alpha_\ell\}$ such that

$$\boldsymbol{\gamma}_q = \sum_{\ell \neq q} \alpha_\ell \boldsymbol{\gamma}_\ell$$

By equation (5) we must then have

$$\mathbf{w}_q = \sum_{\ell \neq q} \alpha_\ell \mathbf{w}_\ell \tag{7}$$

so that at least one of the weight sets that comprises $\mathbf{W}$ cannot be chosen independently from others. We state the implication of equation (7) as a theorem:

*Theorem 2 (***Independent Perceptron Limit):** The number, P, of perceptrons that can be independently composed by choice of $\mathbf{\Theta}$ is less than or equal to N, the number of R-groups.

Proof: [Theorem 2] By equation (7), the number, P, of independent weight sets $\mathbf{w}_p$ that can be composed by choice of $\mathbf{\Theta}$ is upper-bounded by the number of independent $\boldsymbol{\gamma}_\ell$ which is identically the column rank of $\mathbf{\Gamma}$. The number of independent $\{\boldsymbol{\gamma}_\ell\}$ is in turn upper-bounded by the dimension, N, of the $\{\boldsymbol{\gamma}_\ell\}$ which completes the proof. •

$\mathbf{U}$ is fixed and $\mathbf{\Gamma}$ is either a deterministic or random (with known density) vector. In either case, solution of (or approximation to) equation (5) can be approached as an optimization over binary matrices $\{\mathbf{\Theta}\}$ using some norm such as component-wise mean square error between $\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma}$ (or $\mathbf{R}^\top \mathbf{\Gamma}$) and some target $\mathbf{W}$. It is useful to note that if the thresholding function $\phi(\cdot)$ can be adjusted or if different thresholding functions $\phi_i(\cdot)$ can be used after application of reagent G_i then we can choose a set of nonzero constants $\{a_p\}$, $p = 1, 2, \cdots P$ to relax equation (5) and obtain

$$\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} = \mathbf{R}^\top \mathbf{\Gamma} = \left[\begin{array}{ccc} a_1 \mathbf{w}_1 & \cdots & a_P \mathbf{w}_P \end{array} \right] = \mathbf{WA} \tag{8}$$

where

$$\mathbf{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_P \end{bmatrix} \tag{9}$$

If only one threshold function $\phi(\cdot)$ can be used, then $a_p = a \neq 0$, $p = 1, 2, \cdots P$. Otherwise, the non-zero $\{a_p\}$ can be chosen freely.

We rewrite equation (8) as

$$\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} = \mathbf{R}^\top \mathbf{\Gamma} = \mathbf{WA} \tag{10}$$

so that the total mean square error, e^2, between $\mathbf{WA}$ and its approximation $\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma}$ can be written as

$$e^2 = \text{Trace}[\left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \mathbf{WA}\right)^\top \left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \mathbf{WA}\right)] \tag{11}$$

which we can also write as

$$e^2 = \text{Trace}[\left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \mathbf{WA}\right) \left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \mathbf{WA}\right)^\top] \tag{12}$$

since $\text{Trace}[\mathbf{Z}^\top \mathbf{Z}] = \text{Trace}[\mathbf{Z}\mathbf{Z}^\top]$ for any matrix $\mathbf{Z}$. If we then define $\tilde{\mathbf{W}} = \mathbf{WA}$, the necessary optimization is

$$d_\theta = \min_{\mathbf{\Theta}, \mathbf{A}} \text{Trace}[\left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right) \left(\mathbf{\Theta}^\top \mathbf{U}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top] \tag{13}$$

However, one can also pursue a less constrained optimization over non-negative $\mathbf{R}$:

$$d_R = \min_{\mathbf{R}, \mathbf{A}} \text{Trace}[\left(\mathbf{R}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right) \left(\mathbf{R}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top] \tag{14}$$

It is clear that $d_\theta \geq d_R$ since the possible $\mathbf{R}$ are constrained by the $\mathbf{\Theta}$.

V. Input Composition Optimization

A. Optimization for Fixed $\mathbf{A}$

We first show that the minimizations equation (14) and equation (13) are convex in continuous $\mathbf{R}$ and $\mathbf{\Theta}$, respectively, for fixed $\mathbf{A}$.

*Theorem 3 (***d_R and d_θ convexity in $\mathbf{R}$ and $\mathbf{\Theta}$):** Assume $\mathbf{A}$ fixed. d_R and d_θ are convex optimizations in continuous $\mathbf{R}$ and $\mathbf{\Theta}$, respectively, over convex search spaces. d_R and $d_{\mathbf{\Theta}}$ are *strictly* convex with unique solutions iff there are N linearly independent yield vectors, $\boldsymbol{\gamma}_p$.

Proof: [Theorem 3] Let $\lambda \in (0, 1)$. If we set

$$\mathbf{R} = \lambda \mathbf{R}_1 + (1 - \lambda) \mathbf{R}_2$$

convexity of d_R requires

$$\begin{aligned} & \text{Trace}[\left(\mathbf{R}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right) \left(\mathbf{R}^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top] \\ \leq \; & \lambda \text{Trace}[\left(\mathbf{R}_1^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right) \left(\mathbf{R}_1^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top] \\ + \; & (1 - \lambda) \text{Trace}[\left(\mathbf{R}_2^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right) \left(\mathbf{R}_2^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top] \end{aligned} \tag{15}$$

for any two different $\mathbf{R}_1$ and $\mathbf{R}_2$.

Strict convexity further requires equality iff $\lambda = 0$ or $\lambda = 1$. Expansion and rearrangement of equation (15) produces the inequality

$$-\lambda(1 - \lambda) \text{Trace}[(\mathbf{R}_1 - \mathbf{R}_2)^\top \mathbf{\Gamma}\mathbf{\Gamma}^\top (\mathbf{R}_1 - \mathbf{R}_2)] \leq 0 \tag{16}$$

which is clearly satisfied since $\text{Trace}[\mathbf{Z}\mathbf{Z}^\top] \geq 0$ for any nonzero real matrix $\mathbf{Z}$. Therefore d_R is convex. Furthermore, d_R is *strictly* convex iff $\mathbf{\Gamma}\mathbf{\Gamma}^\top$ has rank N so as to preclude $(\mathbf{R}_1 - \mathbf{R}_2)^\top \mathbf{\Gamma} = \mathbf{0}$ for some choice of different $\mathbf{R}_1$ and $\mathbf{R}_2$. $\mathbf{\Gamma}\mathbf{\Gamma}^\top$ has rank N iff $\exists N$ linearly independent $\boldsymbol{\gamma}_p$ which comprise $\mathbf{\Gamma}$.

The structure of d_θ is identical to that of d_R. So, assuming continuous $\mathbf{\Theta}$, the same argument for convexity of d_θ applies.

Finally, the $\mathbf{r}_j$ and $\boldsymbol{\theta}_j$ which comprise $\mathbf{R}$ and $\mathbf{\Theta}$, respectively, are confined to the positive orthant – a convex search space for $\mathbf{R}$ and $\mathbf{\Theta}$. Thus, d_R and d_θ are convex optimizations over convex spaces. Strict convexity implies unique solutions, thus completing the proof. •

Since d_R and d_θ are convex, efficient numerical methods exist to find optimizing continuous $\mathbf{R}^*$ and $\mathbf{\Theta}^*$. However, feasible solutions for $\mathbf{R}^*$ and $\mathbf{\Theta}^*$ are integer matrices. Nonetheless,

once $\mathbf{R}^*$ and $\mathbf{\Theta}^*$ are identified, the continuity of d_R and d_θ in their arguments allow us to examine feasible discrete solutions within the neighborhoods of $\mathbf{R}^*$ and $\mathbf{\Theta}^*$. The quality of these discrete solutions, essentially quantizations, will depend upon their coarseness with respect to the metrics' variation in the continuous search space. Therefore, we would expect solutions could be improved through appropriate choice of $\mathbf{A}$.

B. Choosing $\mathbf{A}$

Consider $\mathbf{R}^*$ a continuous solution to equation (14) for some arbitrary $\mathbf{A}$ which produces minimum error

$$(e^*)^2 = \text{Trace}[\left((\mathbf{R}^*)^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)\left((\mathbf{R}^*)^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top]$$

Then consider the integer approximation $\tilde{\mathbf{R}}^*$ to $\mathbf{R}^*$ obtained by rounding. We define $\mathbf{\Delta}$ such that $\mathbf{R}^* = \tilde{\mathbf{R}} + \mathbf{\Delta}$ and note that each element of $\mathbf{\Delta}$ cannot have magnitude larger than 1. Then consider that if $\tilde{\mathbf{W}}$ is replaced by $\alpha\tilde{\mathbf{W}}$, the optimizing $\mathbf{R}^*$ becomes $\alpha\mathbf{R}^*$ and the integer matrix solution obeys $\alpha\mathbf{R}^* = \tilde{\mathbf{R}} + \mathbf{\Delta}$. Applying $\tilde{\mathbf{R}}$ to equation (14) produces

$$e^2 = \text{Trace}[\left((\mathbf{R}^* - \frac{\mathbf{\Delta}}{\alpha})^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)\left((\mathbf{R}^* - \frac{\mathbf{\Delta}}{\alpha})^\top \mathbf{\Gamma} - \tilde{\mathbf{W}}\right)^\top]$$

Clearly, as $\alpha \to \infty$, $e^2 \to (e^*)^2$. Thus, larger α produces better discrete approximation of $\mathbf{R}^*$. As equation (13) is structurally equivalent to equation (14), the same argument applies to the integer approximation of $\mathbf{\Theta}^*$. Therefore, we can seek continuous solutions to equation (13) and equation (14) and then scale the elements of $\mathbf{A}$ until the performance of the integer matrix approximation is sufficiently close to that of the optimal continuous solution, at least within practical limits of chemical mixture composition. The choice of the a_p in equation (9) will depend on the coarseness with which the corresponding $\tilde{\mathbf{r}}_p$ and $\tilde{\boldsymbol{\theta}}_p$ approximate $\mathbf{w}_p$.

C. Optimization with Random $\mathbf{\Gamma}$

For the case of random equilibrium vectors $\boldsymbol{\gamma}_p$, the optimizations are still convex, as stated in the following lemma:

Lemma 1: (**Random $\mathbf{\Gamma}$**) If $\mathbf{\Gamma}$ is random, the optimizations d_R and d_θ remain convex, and strictly convex iff the correlation matrix $E\left[\mathbf{\Gamma}\mathbf{\Gamma}^\top\right] = \mathbf{K}_\Gamma$ is positive definite.

Proof: [Lemma 1] When the equilibrium vectors $\boldsymbol{\gamma}_p$ are random, the optimizations d_R and d_θ are replaced with optimizations of $E[d_R]$ and $E[d_\theta]$ respectively. The optimizations are still convex since $\mathbf{\Gamma}\mathbf{\Gamma}^\top$ in Theorem 3 is replaced by the positive semi-definite correlation matrix $E\left[\mathbf{\Gamma}\mathbf{\Gamma}^\top\right] = \mathbf{K}_\Gamma$ so that equation (16) is still satisfied. If $\mathbf{K}_\Gamma$ is positive definite (full rank) then the optimizations are strictly convex. •

VI. Discussion & Conclusion

We have shown how to implement chemical perceptrons using small molecules composed of reactive groups (R-groups), and reagents which act differentially (but without impractical exquisite specificity) upon them. Furthermore, given multiple reagents with differing R-group reactivity, we can realize multiple independent perceptrons (one per reagent) assuming there are no more reagents than R-groups. The number of possible inputs to our perceptron is combinatorially huge – our ongoing Ugi synthesis work will produce a library of $\approx 3 \times 10^4$ different molecules, implying $2^{30000} - 1$ non-empty mixtures. And even larger libraries (2×10^6) are possible [12]. Thus, the number of simultaneous multiply-accumulate operations (inputs $\times$ weights to produce an indicator product) implemented by reagent addition can also be extremely large even after selecting only for mixtures which implement a given set of weights. In addition, since small molecules are used and operation relies upon the natural promiscuity of liquid phase chemical reactions, the physical size of these perceptrons is limited only by the amount of indicator product that can be reliably detected. Finally, we note that chemical training [13], [14], layering into larger networks and producing negative weights while not considered here, are the subjects of ongoing work.

Acknowledgment

This research was supported by funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Finally, we thank the anonymous reviewers for their thoughtful comments.

References

[1] M. P. Frank. Throwing computing into reverse. *IEEE Spectrum*, 54(9):32–37, September 2017.
[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *CoRR*, abs/1511.00363, 2015.
[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.
[4] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016.
[5] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. 2016.
[6] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *CoRR*, abs/1510.03009, 2015.
[7] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. *CoRR*, abs/1502.02551, 2015.
[8] J.B. Reece. *Campbell Biology*. Benjamin Cummings / Pearson, 2011.
[9] Carl Poree and Franziska Schoenebeck. A holy grail in chemistry: Computational catalyst design: Feasible or fiction? *Accounts of Chemical Research*, 50(3):605–608, 2017. PMID: 28945392.
[10] Louis Marchetti and Mindy Levine. Biomimetic catalysis. *ACS Catalysis*, 1(9):1090–1118, 2011.
[11] R.T. Morrison and R.N. Boyd. *Organic Chemistry*. Allyn and Bacon, 1966.
[12] Derek S. Tan, Michael A. Foley, Matthew D. Shair, and Stuart L. Schreiber. Stereoselective synthesis of over two million compounds having structural features both reminiscent of natural products and compatible with miniaturized cell-based assays. *Journal of the American Chemical Society*, 120(33):8565–8566, 1998.
[13] P. Banda, C. Teuscher, and M.R. Lakin. Online Learning in a Chemical Perceptron. *Artificial Life*, 19(2):195–219, Spring 2013.
[14] P. Banda, C. Teuscher, and D. Stefanovic. Training an Asymmetric Signal Perceptron Through Reinforcement in an Artificial Chemistry. *Journal of the Royal Society Interface*, 11(93), April 2014.