Biologically Plausible Neural Networks via Evolutionary Dynamics and Dopaminergic Plasticity

Anonymous Author(s)
Affiliation
Address
email

Abstract

Artificial neural networks (ANNs) lack in biological plausibility, chiefly because backpropagation requires a variant of plasticity (precise changes of the synaptic weights informed by neural events that occur downstream in the neural circuit) that is profoundly incompatible with the current understanding of the animal brain. Here we propose that backpropagation can happen in evolutionary time, instead of lifetime, in what we call neural net evolution (NNE). In NNE the weights of the links of the neural net are sparse linear functions of the animal's genes, where each gene has two alleles, 0 and 1. In each generation, a population is generated at random based on current allele frequencies, and it is tested in the learning task through minibatches. The relative performance of the two alleles of each gene is determined, and the allele frequencies are updated via the standard population genetics equations for the weak selection regime. We prove that, under assumptions, NNE succeeds in learning simple labeling functions with high probability, and with polynomially many generations and individuals per generation. NNE is also tested on MNIST with encouraging results. Finally, we explore a further version of biologically plausible ANNs (replacing backprop) inspired by the recent discovery of dopaminergic plasticity.

1 Introduction

In his Turing award lecture [4], neural networks pioneer Geoff Hinton opined that *"evolution can't get gradients because a lot of what determines the relationship between the genotype and the phenotype is outside your control"*. We beg to differ. The well-established equations of population genetics governing evolution under recombination and weak selection [2, 3] do bear a similarity to gradient descent – and, even more, to no-regret learning [3]. In this paper, we show that, in fact, quite effective training of neural nets can be carried out without backpropagation, and in evolutionary time, through a variant of gradient descent. In other words, we argue both theoretically and through experiments that brain circuits specializing in classification tasks could have evolved.

The towering empirical success of ANNs has brought into focus their profound incongruity with what we know about the brain: backpropagation requires that plasticity be informed by downstream events. Clever versions of ANNs have been proposed recently that avoid this criticism: ANNs whose backward weights are random and fixed [6], a backpropagation interpretation of STDP (a widely accepted theory of plasticity) [1], or ANNs driven by neural competition [5].

Here we take a very different approach. We believe that, while forward neural computation is coterminous with life, backpropagation (i.e., local feedback on the performance of the circuit) can be effectively carried out over evolutionary time. Suppose that the brain circuitry for a particular classification task, such as "food/not food", is encoded in the animal's genes, assuming each gene to have two alleles 0 and 1. A (haploid) genotype is a bit string, and the weight of each link of the

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

neural network (assumed to be fixed for simplicity) is a sparse linear function of the genes. Evolution proceeds in generations. At each generation, a gene is an independent binary variable with fixed probability of 1. A population is sampled from this distribution of genotypes, and it experiences a sequence of inputs to the brain circuit. Fitness of each genotype depends, to some small degree, on the animal's success over its lifetime in the specific classification task. In the next generation, the allele frequencies will change slightly, depending on how each allele of each gene fared cumulatively (over both all inputs and all genotypes containing it) in the classification task. These changes follow the standard population genetics equations for the weak selection regime, see [2, 3]; weak selection means that the classification task is only one of the many biological functions (digestion, locomotion, etc.) that affect the animal's fitness.

The question is, can competent brain circuits evolve this way? We offer both theoretical and experimental evidence that this is indeed the case.[1] Our experiments are on the MNIST data set, where we apply Neural Network Evolution (NNE) to learn to categorize digits. In fact, we use the simplest possible version corresponding to a single hidden layer of activations. We find that this already gives surprisingly good accuracy rates. We then use a simple model of arbitrary linear *target* functions and show that NNE will converge to the target. We show in Section 2 that NNE shadows the gradient of the squared error function over a mini-batch, and converges to a $0/1$ allele distribution.

We also propose a totally different alternative to backprop — biologically plausible ANNs based on dopaminergic plasticity. It was recently established experimentally [8] that weights in certain synapses (from the cortex to the striatum but not only) are increased if dopamine was released within 0.5-2 seconds after the synapse's firing. Inspired by this experiment, we define dopaminergic neural nets (DNN), in which the weight of a link that fired (that is, both nodes fired during the current minibatch) is modified by a multiple of $(\frac{1}{4} - err^2)$, where err is the error of the current minibatch. That is, links that fired are rewarded if the result was good, and punished if it was not. We show experimentally that such ANNs can also learn to classify quite well.

The Model. A genotype can be viewed as a vector $x \in \{0,1\}^n$. A probability distribution over the genotypes is given by a vector $p \in [0,1]^n$; a genotype x is sampled by setting $x(i) = 1$ with probability $p(i)$, independently for each i. The neural network corresponding to a genotype x is a *feed-forward neural network* (FFNN) whose weights are computed as follows. For a prediction network having m links, the weights of the links are given by Wx where W is an $m \times n$ sparse *weight generation matrix* W. We choose the entries of W to be random and i.i.d.: with probability β, $W(i,j)$ is chosen uniformly at random from $[-1,1]$, and is 0 with probability $1-\beta$.

The input to the network is a vector y drawn from a distribution $\mathcal{D}$ and has a label (possibly real-valued) $\ell(y)$. The output of the network on an input y is $\mathsf{NNE}_x(y)$. In the simplest linear case (Section 2), $y \in \mathcal{R}^m$ and $\mathsf{NNE}_x(y) = x^T W^T y$. In our experiments (Section 3), we study the case when $\mathcal{D}$ is the uniform distribution over MNIST, and $\mathsf{NNE}_x(\cdot)$ is a 1-layer neural network with a ReLU output gate (see Section 3 for formal definition).

Evolution. We fix a sufficiently large population size $\mathcal{N}$. A generation is a sample of $\mathcal{N}$ independently chosen genotypes from the allele distribution p^t and time t. The weight matrix W is fixed across individuals and generations. Each individual/genotype x in the generation maps to $\mathsf{NNE}_x(\cdot)$. We evaluate this $\mathsf{NNE}_x(\cdot)$ on sufficiently many examples sampled from $\mathcal{D}$. For each genotype x, we measure its performance by comparing the loss $\mathcal{L}(\mathsf{NNE}_x(y), l(y))$ (this could be squared loss, cross-entropy loss, etc.) against that of a random classifier. We then *reward* each allele based on the average performance of the subpopulation in which the allele is present. Let these rewards be $f^t(i)$ and $\bar{f}^t(i)$ if the allele x_i is present and absent respectively. Then the allele probabilities for the next generation are updated as $p^{t+1}(i) \propto \epsilon \cdot p^t(i) \cdot f^t(i)$. This is the standard update rule in population genetics under the weak selection assumption. The multiplier ϵ expresses the small degree to which the performance of this task by the animal confers an evolutionary advantage leading to larger progeny.

Our Contributions. In Section 2, we study a 1-layer NNE with squared loss as loss function, and give a rigorous proof for the case of learning linear functions. In Section 3, we discuss preliminary experiments showing that NNE performs reasonably well on MNIST.

[1] We note incidentally that NNE is very distinct from neuroevolution (see the recent survey [7]), which optimizes ANN architecture and hyperparameters through genetic algorithms

2 A rigorous analysis for the case of learning linear functions

In this section, D is the uniform distribution over vectors in an n-dimensional unit ball. There is a fixed vector a with $\|a\| \leq 1$, and the label of y is $\ell(y) := a^T y$. Our NNE has m input gates connected to one output gate (i.e., no hidden layers). For a genotype x, the weights of the connections are given by Wx. On input y, the NNE outputs $x^T W^T y$.

We analyze NNE using the squared-loss function, i.e. $L(a, b) = (a - b)^2$. For a probability distribution over genotypes p, we define the loss as $\mathcal{L}(p) := \mathsf{E}_{x\sim p}\mathsf{E}_{y\sim D}(\mathsf{NNE}_x(y) - \ell(y))^2$. Our first observation is that the squared loss perfomance per allele is in fact a function of the gradient of the squared loss. For a matrix W, we denote its i'th column by $W^{(i)}$.

Lemma 1 *When the network function g is linear, i.e.,* $\mathsf{NNE}_x(y) = x^T W^T y$,

1. $\frac{\partial}{\partial p^t(i)}(\mathcal{L}(p^t)) = 2\mathsf{E}_{x\sim p^t}\mathsf{E}_{y\sim D}\left((W^{(i)} \cdot y)(\mathsf{NNE}_x(y) - \ell(y))\right)$.
2. $f^t(i) = \mathcal{L}(p^t) + (1 - p^t(i))\frac{\partial}{\partial p^t(i)}(\mathcal{L}(p^t))$ *and* $\bar{f}^t(i) = \mathcal{L}(p^t) - p^t(i)\frac{\partial}{\partial p^t(i)}(\mathcal{L}(p^t))$.

We use this to prove the following theorem.

Theorem 1 *There is an $\eta > 0$ s.t. we have* $\mathsf{E}(\mathcal{L}(p^{t+1})) \leq \mathcal{L}(p^t) - \eta\|\nabla p^t\|^2$.

Thus, evolution converges to a vector p where the gradient is close to zero. In the case of a linear labeling/target function, along with the next lemma, this implies that evolution converges to a p which gives the correct linear labeling.

Lemma 2 *For any $\delta \in (0, 1]$, with $n = O(m + \log(1/\delta))$, with probability at least $1 - \delta$, there exists an allele distribution p s.t. $Wp = a$. Moreover, with probability at least $3/4$, for any $\epsilon \in (0, 1]$, with $n = \Omega(m(\log(1/\epsilon)/\epsilon^2)$, there is a $0/1$ vector x s.t. $\frac{(Wx)\cdot a}{\|Wx\|\|a\|} \geq 1 - \epsilon$.*

We remark that the above guarantee works for *every* linear target function in $\mathbb{R}^m$. To learn, with high confidence, the target function from among d unknown (arbitrary) linear functions, m above can be replaced by $\log d$.

3 Experimental Validation

NNE on MNIST. We validate our idea using the MNIST handwritten digit recognition task. We experiment with NNE on a small subset S drawn uniformly at random from the MNIST training set (we ensure that the set S is balanced). The alleles for the first generation are drawn uniformly at random from $[0, 1]$. Our network has 784 input units, one hidden layer of $|h^1| = 1000$ units with ReLU activation and an output layer of 10 units with softmax activation. To avoid redundancy, we make the network connections sparse. Any edge from input to the hidden layer is present with a probability $\gamma = 0.1$. However, hidden layer to output layer remains fully connected. For the input sample $s = (y, l(y))$ where $l(y)$ is now a one-hot encoding of the label, $\mathsf{NNE}_x(y)$ represents the softmax output. Using cross-entropy loss, $L_{x,s} = -\sum_{c\in[C]} l(y)_c \log\left(\mathsf{NNE}_x(y)_c\right)$, each genotype x's performance $\delta_{x,s}$ on a sample s is measured by comparing against a random classifier, $\delta_{x,s} = \max\{0, \alpha^2 - L_{x,s}^2\}$. For MNIST, $\alpha = -\log(1/10)$. Then $\delta_x = -\frac{1}{|S|}\sum_{s\in[S]}\delta_{x,s}$ is the average performance of genotype x across $|S| = 2000$ samples. For each allele, we calculate its performance by averaging over the performance of the population, given if its present or absent, $f^t(i) = \sum_x \delta_x x(i)/\mathcal{N}$ and $\bar{f}^t(i) = \sum_x \delta_x(1 - x(i))/\mathcal{N}$. Now the allele probabilities for the next generation will be,

$$p^{t+1} = \frac{p^t(1 + \epsilon f^t)}{1 + \epsilon\bar{f}^t + \epsilon p^t(f^t - \bar{f}^t)}$$

where ϵ is sufficiently small learning rate.

A crucial choice is the number of genes. In our experiments, we involve a few thousands of genes; this is not unreasonable, as it is estimated that about $5,000$ genes are expressed in the cells of the mammalian brain. To investigate further, we compare the performance of our algorithm with

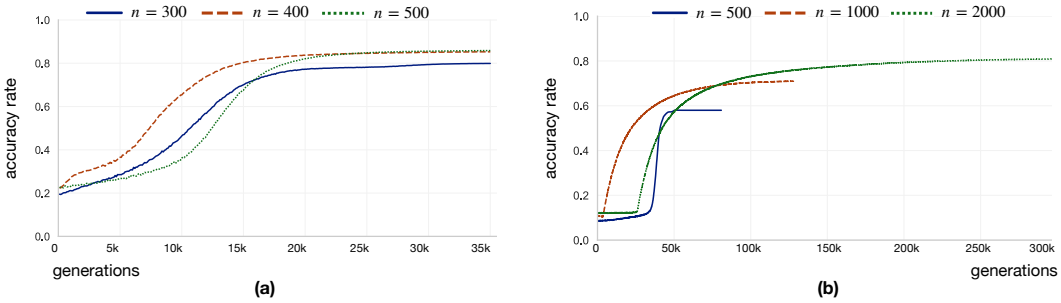

Figure 1: Number of genes (n) vs performance of NNE: **(a)** Accuracy rates of NNE classifying the first five digits [0-4] of MNIST; effect of number of genes on performance. **(b)** Similarly, we also plot the accuracy rates of NNE classifying all ten digits from MNIST dataset while varying the number of genes. The accuracy trends show that more the number of genes, better the performance of NNE, but at the cost of more training time.

increasing value of n (the number of genes). Figure 1 presents the validation accuracy trends on the same network described above for five class [0-4] classification and for full MNIST dataset. We observe that the accuracy rate of the network improves significantly with increase in the number of genes. However, it requires much longer training time to achieve a desired accuracy rate. NNE as described above achieves 83% train accuracy on S and **79%** test accuracy on full MNIST test set. Moreover, 94.65% of the allele probabilities are either less than 0.05 or greater than 0.95.

With output layer training. Another line of our experimentation includes performing sign gradient descent (signGD) on the output layer weights. The gradient of the cross-entropy loss function w.r.t. an output layer weight w_{ij} will be, $\nabla_{w_{ij}} L_{x,s} = (\mathsf{NNE}_x(y)_j - l(y)_j) \cdot h_i^{L-1}$. Clearly, *symmetric feedback weights* problem doesn't exist in the output layer, which makes the computation of the sign of gradient biologically plausible. For the same network described above, we randomly initialize the network weights using learned allele distribution. We then calculate the sign of the gradient of the output layer weights and update the weights in the opposite direction, controlled by a sufficiently small learning rate ϵ', similar to gradient descent; $w_{ij}^L := w_{ij}^L - \epsilon' \cdot sign\big((\hat{y}_j - y_j)h_i^{L-1}\big)$. This method gives a significant boost in the performance, 83% accuracy on full MNIST test set. SignGD on DNN compares well with usual gradient descent on the output layer.

References

[1] Y. Bengio, D.-H. Lee, J. Bornschein, and Z. Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[2] R. Bürger. *The mathematical theory of selection, recombination, and mutation*, volume 228. Wiley Chichester, 2000.

[3] E. Chastain, A. Livnat, C. Papadimitriou, and U. Vazirani. Algorithms, games, and evolution. *Proceedings of the National Academy of Sciences*, 111(29):10620–10623, 2014.

[4] G. E. Hinton. Turing award lecture "the deep learning revolution", 1:27:43.

[5] D. Krotov and J. J. Hopfield. Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci. U.S.A.*, 116(16):7723–7731, 2019.

[6] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. In *Nature communications*, 2016.

[7] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1:24–35, 2019.

[8] S. Yagishita, A. Hayashi-Takagi, G. C. Ellis-Davies, H. Urakubo, S. Ishii, and H. Kasai. A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620, 2014.