Learning to solve the credit assignment problem

Anonymous Author(s)
Affiliation
Address
email

Abstract

Backpropagation is driving today's artificial neural networks. However, despite extensive research, it remains unclear if the brain implements this algorithm. Among neuroscientists, reinforcement learning (RL) algorithms are often seen as a realistic alternative. However, the convergence rate of such learning scales poorly with the number of involved neurons. Here we propose a hybrid learning approach, in which each neuron uses an RL-type strategy to learn how to approximate the gradients that backpropagation would provide. We show that our approach learns to approximate the gradient, and can match the performance of gradient-based learning on fully connected and convolutional networks. Learning feedback weights provides a biologically plausible mechanism of achieving good performance, without the need for precise, pre-specified learning rules.

1 Introduction

It is unknown how the brain solves the credit assignment problem when learning: how does each neuron know its role in a positive (or negative) outcome, and thus know how to change its activity to perform better next time? Biologically plausible solutions to credit assignment include those based on reinforcement learning (RL) algorithms [4]. In these approaches a globally distributed reward signal provides feedback to all neurons in a network. However these methods have not been demonstrated to operate at scale. For instance, variance in the REINFORCE estimator scales with the number of units in the network. This drives the hypothesis that learning in the brain must rely on additional structures beyond a global reward signal.

In artificial neural networks, credit assignment is performed with gradient-based methods computed through backpropagation. This is significantly more efficient than RL-based algorithms. However there are well known problems with implementing backpropagation in biologically realistic neural networks. For instance backpropagation requires a feedback structure with the same weights as the feedforward network to communicate gradients (so-called weight transport). Yet such structures are not observed in neural circuits. Despite this, backpropagation is the only method known to solve learning problems at scale. Thus modifications or approximations to backpropagation that are more plausible have been the focus of significant recent attention [8, 3]. Notably, it turns out that weight transport can be avoided by using fixed, random feedback weights, through a phenomenon called feedback alignment [8]. However feedback alignment does not work in larger, more complicated network architectures (such as convolutional networks).

Here we propose to use an RL algorithm to train a feedback system to enable learning. We propose to use a REINFORCE-style perturbation approach to train a feedback signal to approximate what would have been provided by backpropagation. We demonstrate that our model learns as well as regular backpropagation in small models, overcomes the limitations of fixed random feedback weights ("feedback alignment") on more complicated feedforward networks, and can be utilized in convolutional networks. Our method illustrates a biologically realistic way the brain could perform gradient descent-like learning.

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

2 Method

Let an N hidden-layer network be given by $\hat{\mathbf{y}} = f(\mathbf{x}) \in \mathbb{R}^p$, composed of a set of layer-wise summation and non-linear activations $\mathbf{h}^i = f^i(\mathbf{h}^{i-1}) = \sigma\left(W^i \mathbf{h}^{i-1}\right)$, for hidden layer states $\mathbf{h}^i \in \mathbb{R}^{n_i}$, non-linearity σ and with input $\mathbf{h}^0 = \mathbf{x}$ and output $\mathbf{h}^{N+1} = \hat{\mathbf{y}}$. Define $\mathcal{L}$ as the loss function $\mathcal{L}(\mathbf{x}, \mathbf{y})$, where the data $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ are drawn from a distribution ρ. Our aim is then to minimize: $\mathbb{E}_\rho\left[\mathcal{L}(\mathbf{x}, \mathbf{y})\right]$. Backpropagation computes the error signal $\tilde{\mathbf{e}}^i$ in a top-down fashion: $\mathbf{e}^i = \begin{cases} \partial\mathcal{L}/\partial\hat{\mathbf{y}} \circ \sigma'(W^i \mathbf{h}^{i-1}), & i = N+1; \\ \left((W^{i+1})^T \mathbf{e}^{i+1}\right) \circ \sigma'(W^i \mathbf{h}^{i-1}), & 1 \leq i \leq N \end{cases}$. Let the loss gradient term be denoted as $\lambda^i = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^i} = (W^{i+1})^T \mathbf{e}^{i+1}$. Here we replace λ^i with an approximation, with its own parameters to be learned: $\lambda^i \approx \mathbf{g}(\mathbf{h}^i, \tilde{\mathbf{e}}^{i+1}; B)$, for parameters B. We will use $\tilde{\mathbf{e}}^i$ to denote the gradient signal backpropagated through the synthetic gradients, and $\mathbf{e}^i$ for the true gradients. To estimate B we use stochasticity inherent to biological neural networks. For each input each unit produces a noisy response: $\mathbf{h}_t^i = \sigma\left(\sum_k W_{\cdot k}^i \mathbf{h}_t^{i-1}\right) + c_h \xi_t^i$, for independent Gaussian noise $\xi^i \sim \nu = \mathcal{N}(0, I)$ with standard deviation $c_h > 0$. This then generates a noisy loss $\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}, \xi)$ and a baseline loss $\mathcal{L}(\mathbf{x}, \mathbf{y}) = \tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}, 0)$. We will use the noisy response to estimate gradients, that then allow us to optimize the baseline $\mathcal{L}$. This is achieved by linearizing the loss: $\tilde{\mathcal{L}} \approx \mathcal{L} + \frac{\partial \mathcal{L}}{\partial h_j^i} c_h \xi_j^i$. This gives $\mathbb{E}((\tilde{\mathcal{L}} - \mathcal{L}) c_h \xi_j^i | \mathbf{x}, \mathbf{y}) \approx c_h^2 \frac{\partial \mathcal{L}}{\partial h_j^i}|_{\mathbf{x},\mathbf{y}}$, with expectation taken over the noise distribution $\nu(\xi)$. Thus we obtain an estimator of the loss gradient: $\hat{\lambda}^i := (\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}, \xi) - \mathcal{L}(\mathbf{x}, \mathbf{y})) \frac{\xi^i}{c_h}$.

3 Applications

3.1 Fully connected networks solving MNIST

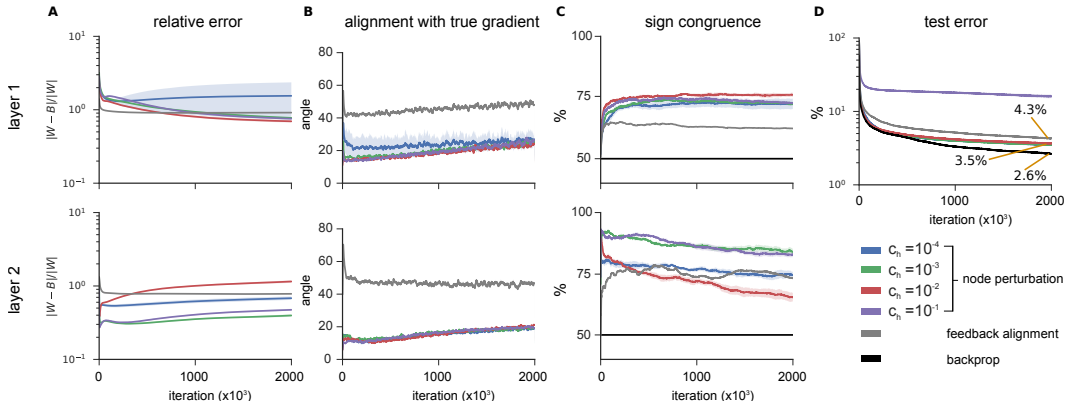

Figure 1: Node perturbation in small 4-layer network (784-50-20-10 neurons), for varying noise levels c, compared to feedback alignment and backpropagation. (A) Relative error between feedforward and feedback matrix. (B) Angle between true gradient and synthetic gradient estimate for each layer. (C) Percentage of signs in W_i and B_i that are in agreement. (D) Test error for node perturbation, backpropagation and feedback alignment. Curves show mean plus/minus standard error over 5 runs. Hyperparameters found through random search.

To demonstrate the method can be used to solve simple supervised learning problems we use node perturbation with a four-layer network and MSE loss to solve MNIST (Fig. 1). We approximate loss gradients as follows: $\mathbf{g}(\mathbf{h}^i, \tilde{\mathbf{e}}^{i+1}; B) = (B^{i+1})^T \tilde{\mathbf{e}}^{i+1}$. The feedback parameters B^{i+1} are estimated by solving the least squares problem: $\hat{B}^{i+1} = \text{argmin}_B \mathbb{E}\|B^T \tilde{\mathbf{e}}^{i+1} - \hat{\lambda}^i\|_2^2$, where $\hat{\lambda}$ is the perturbation-based estimator derived above. B is updated with each mini-batch using stochastic gradient-descent to minimize this loss.[1] Updates to W^i are made using the synthetic gradients $\Delta W^i = \eta \tilde{\mathbf{e}}^i \mathbf{h}^{i-1}$, for

[1] We can prove this converges to the true weights when applied to either non-linear shallow network or a deep linear network, though these results are omitted here for brevity.

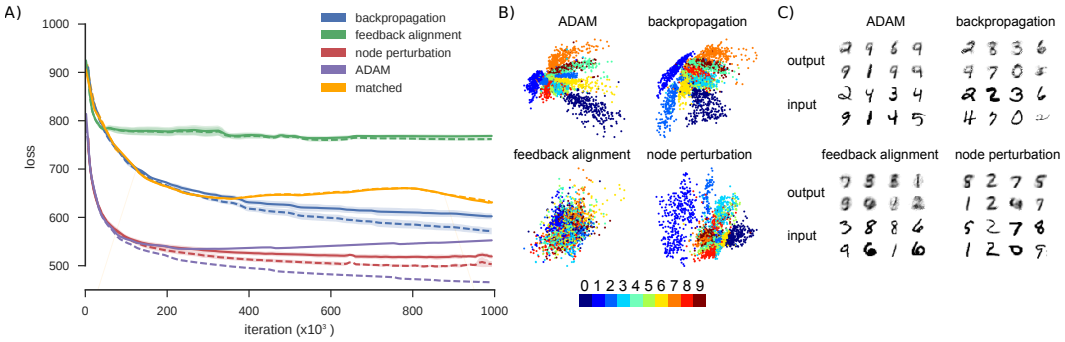

Figure 2: Results with five-layer MNIST autoencoder network. a) Mean loss plus/minus standard error over 10 runs. Dashed lines represent training loss, solid lines represent test loss. b) Latent space activations, colored by input label for each method. c) Sample outputs for each method. Hyperparameters found through random search.

learning rate η. We observed that the system is able to provide a close correspondence between the feedforward and feedback matrices in both layers of the network (Fig. 1a). The relative error between B_i and W_i is lower than what is observed for feedback alignment, suggesting that this co-adaptation of W_i and B_i is indeed beneficial. We observe that the alignment (the angle between the estimated gradient and the true gradient, proportional to $\mathbf{e}^T W B^T \tilde{\mathbf{e}}$) is lower for node perturbation than for feedback alignment (Fig. 1b). Recent studies have shown that sign congruence of the feedforward and feedback matrices is all that is required to achieve good performance [10]. Here the sign congruence is also higher in node perturbation (Fig. 1c). Finally, the learning performance of node perturbation is comparable to backpropagation (Fig. 1d) – achieving close to 3% test error. These suggest node perturbation for learning feedback weights can be used in deep networks.

3.2 Auto-encoding MNIST

A known shortcoming of feedback alignment is in auto-encoding networks with tight bottleneck layers [8]. To see if our method has the same shortcoming we examine a simple auto-encoding network with MNIST input data (size 784-200-2-200-784, MSE loss). We also compare the method to the 'matching' learning rule [9], in which updates to B match updates to W. As expected, feedback alignment performs poorly. Node perturbation actually performs better than backpropagation, and comparable to ADAM (Fig. 2a). In fact ADAM begins to overfit, while node perturbation does not. The matched learning rule performs similarly to backpropagation. These results are surprising at first glance. Perhaps, similar to feedback alignment, learning feedback weights strikes the right balance between providing a useful signal to learn, and constraining the updates to be sufficiently aligned with B, acting as a type of regularization [8]. The noise added when estimating the feedback weights may also serve to regularize the latent representation, as, indeed, the latent space learnt by node perturbation shows a more evenly distributed separation of digits. While, in contrast, the representations learnt by backprop and ADAM show more structure, and feedback alignment does not learn a useful representation at all (Fig. 2b,c). These results show that node perturbation is able to successfully communicate error signals through thin layers of a network as needed.

3.3 Convolutional neural networks solving CIFAR10

Finally we test the method on a convolutional neural network (CNN) solving CIFAR10. The CNN has the architecture Conv(3x3, 1x1, 32), MaxPool(3x3, 2x2), Conv(5x5, 1x1, 128), MaxPool(3x3, 2x2), Conv(5x5, 1x1, 256), MaxPool(3x3, 2x2), FC 2048, FC 2048, Softmax(10), with hyperparameters found through random search. For this network we learn feedback weights direct from the output layer to each earlier layer: $\mathbf{g}(\mathbf{h}^i, \tilde{\mathbf{e}}^{i+1}; \theta_i) = (B^{i+1})^T \tilde{\mathbf{e}}^N$ (similar to 'direct feedback alignment'). Here this was solved by gradient-descent. We obtain a test accuracy of 75.2%. When compared with fixed feedback weights (test accuracy of 72.5%) and backpropagation (test accuracy of 77.2%), we see it is advantageous to learn feedback weights. This shows the method can be used in a CNN, and can solve challenging computer vision problems without weight transport.

4 Discussion

Here we implement a perturbation-based synthetic gradient method to train neural networks. We show that this hybrid approach can be used in both fully connected and convolutional networks. By removing both the symmetric feedforward, feedback weight requirement imposed by backpropagation this approach is a step towards more biologically-plausible deep learning. In contrast to many perturbation-based methods, this hybrid approach can solve large-scale problems. We thus believe this approach can provide powerful and biologically plausible learning algorithms.

While previous research has provided some insight and theory for how feedback alignment works [8, 3, 2] the effect remains somewhat mysterious, and not applicable in some network architectures. Recent studies have shown that some of these weaknesses can be addressed by instead imposing sign congruent feedforward and feedback matrices [10]. Yet what mechanism may produce congruence in biological networks is unknown. Here we show that the shortcomings of feedback alignment can be addressed in another way: the system can learn to adjust weights as needed to provide a useful error signal. Our work is closely related to Akrout et al 2019 [1], which also uses perturbations to learn feedback weights. However our approach does not divide learning into two phases, and training of the feedback weights does not occur in a layer-wise fashion.

Here we tested our method in an idealized setting, however it is consistent with neurobiology in two important ways. First, it involves the separate learning of feedforward and feedback weights. This is possible in cortical networks where complex feedback connections exist between layers, and where pyramidal cells have apical and basal compartments that allow for separate integration of feedback and feedforward signals [5]. Second, noisy perturbations are common in neural learning models. There are many mechanisms by which noise can be measured or approximated [4, 7], or neurons could use a learning rule that does not require knowing the noise [6]. While our model involves the subtraction of a baseline loss to reduce the variance of the estimator, this does not affect the expected value of the estimator; technically the baseline could be removed or approximated [7]. Thus we believe our approach could be implemented in neural circuits. There is a large space of plausible learning rules that can learn feedback signals in order to more efficiently learn. These promise to inform both models of learning in the brain and learning algorithms in artificial networks. Here we take an early step in this direction.

References

[1] Mohamed Akrout, Collin Wilson, Peter C Humphreys, Timothy Lillicrap, and Douglas Tweed. Deep Learning without Weight Transport. *ArXiv e-prints*, 2019.

[2] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the Machine: Random Backpropagation and the Deep Learning Channel. *Artificial Intelligence*, 260:1–35, 2018.

[3] Sergey Bartunov, Adam Santoro, Blake Richard, Geoffrey Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *ArXiv e-prints*, 2018.

[4] Ila R Fiete, Michale S Fee, and H Sebastian Seung. Model of Birdsong Learning Based on Gradient Estimation by Dynamic Perturbation of Neural Conductances. *Journal of neurophysiology*, 98:2038–2057, 2007.

[5] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *Elife*, 6, December 2017.

[6] Benjamin James Lansdell and Konrad Paul Kording. Spiking allows neurons to estimate their causal effect. *bioRxiv*, pages 1–19, 2018.

[7] Robert Legenstein, Steven M. Chase, Andrew B. Schwartz, Wolfgang Maas, and W. Maass. A Reward-Modulated Hebbian Learning Rule Can Explain Experimentally Observed Network Reorganization in a Brain Control Task. *Journal of Neuroscience*, 30(25):8400–8410, 2010.

[8] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *Nature Communications*, 7:13276, 2016.

[9] Marco Martinolli, Wulfram Gerstner, and Aditya Gilra. Multi-Timescale Memory Dynamics Extend Task Repertoire in a Reinforcement Learning Network With Attention-Gated Memory. *Front. Comput. Neurosci.* . . . , 12(July):1–15, 2018.

[10] Will Xiao, Honglin Chen, Qianli Liao, and Tomaso Poggio. Biologically-Plausible Learning Algorithms Can Scale to Large Datasets. *ArXiv e-prints*, (92), 2018.