ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/333600980

Learning to solve the credit assignment problem

Preprint · June 2019

CITATIONS
0

READS
905

3 authors, including:

Prashanth Ravi Prakash
University of Pennsylvania
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Neuroscience, Neural network View project

Project RSC related circuit View project

All content following this page was uploaded by Prashanth Ravi Prakash on 24 September 2019.

The user has requested enhancement of the downloaded file.

arXiv:1906.00889v2 [q-bio.NC] 5 Jun 2019

Learning to solve the credit assignment problem

Benjamin James Lansdell
Department of Bioengineering
University of Pennsylvania
Pennsylvania, PA 19104
lansdell@seas.upenn.edu

Prashanth Ravi Prakash
Department of Bioengineering
University of Pennsylvania
Pennsylvania, PA 19104

Konrad Paul Kording
Department of Bioengineering
University of Pennsylvania
Pennsylvania, PA 19104

Abstract

Backpropagation is driving today's artificial neural networks (ANNs). 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: neurons can randomly introduce change, and use unspecific feedback signals to observe their effect on the cost and thus approximate their gradient. However, the convergence rate of such learning scales poorly with the number of involved neurons (e.g. $\mathcal{O}(N)$). Here we propose a hybrid learning approach. Each neuron uses an RL-type strategy to learn how to approximate the gradients that backpropagation would provide – in this way it *learns to learn*. We provide proof that our approach converges to the true gradient for certain classes of networks. In both feed-forward and recurrent networks, we empirically show that our approach learns to approximate the gradient, and can match the performance of gradient-based learning. Learning to learn 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? Actions are rarely immediately rewarded (or punished), so each neuron must further determine *which* of a potential series of its actions is responsible for ultimate reward. This is a challenge for models of learning in the brain.

Biologically plausible solutions to credit assignment include those based on reinforcement learning (RL) and reward-modulated STDP [7, 11, 12, 32, 40]. In these approaches a globally distributed reward signal provides feedback to all neurons in a network. Essentially, changes in rewards from a baseline, or expected, level are correlated with noise in neural activity, allowing a stochastic approximation of the gradient to be computed. However these methods have not been demonstrated to operate at scale. For instance, variance in the REINFORCE estimator [57] scales with the number of units in the network [46]. This drives the hypothesis that learning in the brain must rely on additional structures beyond a global reward signal.

In artificial neural networks (ANNs), credit assignment is performed with gradient-based methods computed through backpropagation [49]. This is significantly more efficient than RL-based algo-

Preprint. Under review.

rithms, with ANNs now matching or surpassing human-level performance in a number of domains [42, 53, 30, 19, 16, 50]. However there are well known problems with implementing backpropagation in biologically realistic neural networks. One problem is known as weight transport: an exact implementation of backpropagation requires a feedback structure with the same weights as the feedforward network to communicate gradients. Such a symmetric feedback structure has not been observed in neural circuits. A further problem, particularly in recurrent neural networks (RNNs), is that the temporal trace of each neuron's activity must be somehow stored by the network until the backward pass occurs (though eligibility traces may be able to address this issue to some extent [13, 33]). Despite these issues, backpropagation is the only method known to solve supervised and reinforcement learning problems at scale. Thus modifications or approximations to backpropagation that are more plausible have been the focus of significant recent attention [51, 35, 31, 28].

These efforts do show some ways forward. Synthetic gradients demonstrate that learning can be based on approximate gradients, and need not be temporally locked [24, 10]. In small feedforward networks, somewhat surprisingly, fixed random feedback matrices in fact suffice for learning [35] (a phenomenon known as feedback alignment). But still issues remain: feedback alignment does not work in RNNs, very deep networks, networks with tight bottleneck layers. Regardless, these results show that rough approximations of a gradient signal can be used to learn, and suggest that even relatively inefficient methods of approximating the gradient may be good enough.

On this basis, here we propose an RL algorithm to train a feedback system to enable learning. Recent work has explored similar ideas, but not with the explicit goal of approximating backpropagation [40, 41, 54]. RL-based methods like REINFORCE may be inefficient when used as a base learner, but they may be sufficient when used to train a system that itself instructs a base learner. We propose to use REINFORCE-style perturbation approach to train a feedback signal to approximate what would have been provided by backpropagation. Our system *learns to learn*.

Learning to learn is often framed as a two-learner system: one system that updates a network's weights, and another system that modifies the learner to update weights more efficiently [29]. A two learner system may in fact align well with cortical neuron physiology. For instance, the dendritic trees of pyramidal neurons consist of an apical and basal component [14, 26]. Similarly, climbing fibers and Purkinje cells may define a learner/teacher system in the cerebellum [38]. These components allow for independent integration of two different signals. Indeed such a setup has been shown to support supervised learning in feedforward networks [14, 26]. Learning to learn may thus provide a realistic solution to the credit assignment problem.

Here we implement a system that learns to use feedback signals trained with reinforcement learning via a global reward signal. This provides a plausible account of how the brain may perform deep learning. We mathematically analyze the model, and compare its capabilities to other biologically plausible accounts of learning in ANNs. We prove consistency of the estimator in particular cases, extending the few theoretical results available in synthetic gradients [24, 10]. We demonstrate that our synthetic gradient model learns as well as regular backpropagation in small models, overcomes the limitations of feedback alignment on more complicated feedforward networks, and can be utilized in recurrent networks. Thus our method may provide an account of how the brain performs gradient descent learning.

2 Learning to learn through perturbations

We use the following notation. Let $\mathbf{x} \in \mathbb{R}^m$ represent an input vector. Let an N hidden-layer network be given by $\hat{\mathbf{y}} = f(\mathbf{x}) \in \mathbb{R}^p$. This is 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 denoting $\mathbf{h}^0 = \mathbf{x}$ and $\mathbf{h}^{N+1} = \hat{\mathbf{y}}$. Some loss function L is defined in terms of the network output: $L(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{x}))$. Let $\mathcal{L}$ denote the loss as a function of $(\mathbf{x}, \mathbf{y})$: $\mathcal{L}(\mathbf{x}, \mathbf{y}) = L(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{x}))$. Let data $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ be drawn from a distribution ρ. Then we aim to minimize:

$$\mathbb{E}\left[\mathcal{L}(\mathbf{x}, \mathbf{y})\right].$$

Backpropagation relies on the error signal $\mathbf{e}^i$, computed in a top-down fashion:

$$\mathbf{e}^i = \begin{cases} \partial \mathcal{L}/\partial \hat{\mathbf{y}} \circ \sigma'(W^i \mathbf{h}^{i-1}), & i = N+1; \\ \left((W^{i+1})^T \mathbf{e}^{i+1}\right) \circ \sigma'(W^i \mathbf{h}^{i-1}), & 1 \le i \le N. \end{cases}$$

2.1 Basic setup

Let the loss gradient term be denoted as

$$\lambda^i = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^i} = (W^{i+1})^T \mathbf{e}^{i+1}.$$

In this work we replace λ^i with an approximation with its own parameters to be learned (known as a synthetic gradient [24, 10], or error critic [56]):

$$\lambda^i \approx \mathbf{g}(\mathbf{h}^i, \mathbf{e}^{i+1}; \theta),$$

for parameters θ. This setup can accommodate both top-down and bottom-up information, and encompasses a number of published models [24, 10, 35, 44, 34, 58].

2.2 Stochastic networks and gradient descent

To learn a synthetic gradient we use stochasticity inherent to biological neural networks. A number of biologically plausible learning rules exploit random perturbations in neural activity [59, 52, 12, 11, 54]. Here, at each time each unit produces a noisy response:

$$\mathbf{h}_t^i = \sigma\left(\sum_k W^i_{\cdot k} \mathbf{h}_t^{i-1}\right) + c_h \xi_t^i,$$

for independent Gaussian noise $\xi^i \sim \nu = \mathcal{N}(0, I)$ and standard deviation $c_h > 0$. This 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 $\mathcal{L}$ – the gradients used for weight updates are computed using the deterministic baseline.

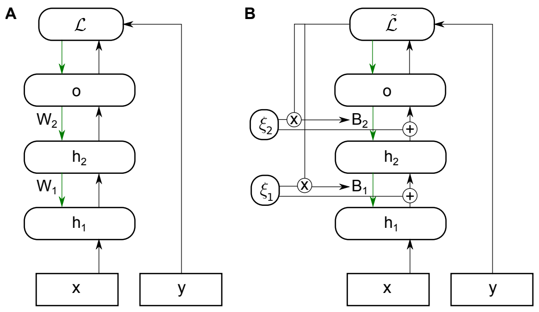

Figure 1: Learning backpropagation through node perturbation. (A) Backpropagation sends error information from an output loss function, $\mathcal{L}$, through each layer from top to bottom via the same matrices W_i used in the feedforward network. (B) Node perturbation introduces noise in each layer, ξ_i, that perturbs that layer's output and resulting loss function. The perturbed loss function, $\tilde{\mathcal{L}}$, is correlated with the noise to give an estimate of the error current. This estimate is used to update feedback matrices B_i to better approximate the error signal.

2.3 Synthetic gradients via node perturbation

For Gaussian white noise, the well-known REINFORCE algorithm [57] coincides with the node-perturbation method [12, 11]. Node perturbation works by linearizing the loss:

$$\tilde{\mathcal{L}} \approx \mathcal{L} + \frac{\partial \mathcal{L}}{\partial h_j^i} c_h \xi_j^i, \tag{1}$$

such that

$$\mathbb{E}\left((\tilde{\mathcal{L}} - \mathcal{L}) c_h \xi_j^i | \mathbf{x}, \mathbf{y}\right) \approx c_h^2 \frac{\partial \mathcal{L}}{\partial h_j^i}\bigg|_{\mathbf{x},\mathbf{y}},$$

with expectation taken over the noise distribution $\nu(\xi)$. This provides an estimator of the loss gradient

$$\hat{\lambda}^i := (\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}, \xi) - \mathcal{L}(\mathbf{x}, \mathbf{y})) \frac{\xi^i}{c_h}. \tag{2}$$

The approximation (6) is made more precise in Lemma 3.

2.4 Training a feedback network

There are many possible sensible choices of $\mathbf{g}(\cdot)$. For example, taking $\mathbf{g}$ as simply a function of each layer's activations: $\lambda^i = \mathbf{g}(\mathbf{h}^i)$ is in fact sufficient parameterization to express the true gradient function [24]. We may expect, however, that the gradient estimation problem be simpler if each layer is provided with some error information obtained from the loss function and propagated in a top-down fashion. Symmetric feedback weights may not be biologically plausible, and random fixed weights may only solve certain problems of limited size or complexity [35]. However, a system that can learn to appropriate feedback weights B may be able to align the feedforward and feedback weights as much as is needed to successfully learn.

We investigate $\mathbf{g}(\mathbf{h}^i, \mathbf{e}^{i+1}; \theta_i) = (B^{i+1})^T \mathbf{e}^{i+1}$, which describes a non-symmetric feedback network (Figure 1). Parameters B^{i+1} are estimated by solving the least squares problem:

$$\hat{B}^{i+1} = \text{argmin}_B \mathbb{E} \left\| B^T \mathbf{e}^{i+1} - \hat{\lambda}^i \right\|_2^2 . \tag{3}$$

Here, unless otherwise noted, this was solved by gradient-descent. Refer to the supplementary material for additional experimental descriptions and parameters.

3 Theoretical results

We can prove the estimator (7) is consistent as $c_h \to 0$ in two particular cases. To establish these results we must distinguish the true loss gradients from their synthetic estimates. Let $\tilde{\mathbf{e}}^i$ be loss gradients computed by backpropagating the synthetic gradients

$$\tilde{\mathbf{e}}^i = \begin{cases} \partial \mathcal{L} / \partial \hat{\mathbf{y}} \circ \sigma'(W^i \mathbf{h}^{i-1}), & i = N+1; \\ \left((\hat{B}^{i+1})^T \tilde{\mathbf{e}}^{i+1} \right) \circ \sigma'(W^i \mathbf{h}^{i-1}), & 1 \le i \le N. \end{cases}$$

To prove consistency we must show the expectation of the Taylor series approximation (6) is well behaved. That is, we must show the expected remainder term of the expansion:

$$\mathcal{E}_j^i(c_h) = \mathbb{E}\left[\frac{1}{c_h^2} \sum_{m=2}^{\infty} \frac{\mathcal{L}_{ij}^{(m)}}{m!} (c_h \xi_j^i)^{m+1} | \mathbf{x}, \mathbf{y} \right],$$

is finite. This requires some additional assumptions on the problem. We prove the result under the following assumptions:

- A1: the noise ξ is subgaussian,
- A2: the loss function $\mathcal{L}(\mathbf{x}, \mathbf{y})$ is analytic on $\mathcal{D}$,
- A3: the error matrices $\tilde{\mathbf{e}}^n (\tilde{\mathbf{e}}^n)^T$ are full rank, for $1 \le n \le N+1$,
- A4: the mean of the remainder and error terms is bounded:

$$\mathbb{E}\left[\mathcal{E}^n(c_h) (\tilde{\mathbf{e}}^{n+1})^T \right] < \infty,$$

for $1 \le n \le N$.

Under these assumptions convergence follows from consistency of the least squares estimator for linear models.

Consider first convergence of the final layer feedback matrix, B^{N+1}.

Theorem 1. *Assume A1-4. Then the least squares estimator*

$$(\hat{B}^{N+1})^T := \hat{\lambda}^N (\mathbf{e}^{N+1})^T \left(\mathbf{e}^{N+1} (\mathbf{e}^{N+1})^T \right)^{-1}, \tag{4}$$

solves (8) *and converges to the true feedback matrix, in the sense that:*

$$\lim_{c_h \to 0} \operatorname*{plim}_{T \to \infty} \hat{B}^{N+1} = W^{N+1},$$

where plim *indicates convergence in probability.*

Theorem 1 thus establishes convergence of B in a shallow (1 hidden layer) non-linear network, provided the activation function and loss function are smooth.

In a deep, linear network we can also use Theorem 1 to establish convergence over the rest of the layers of the network.

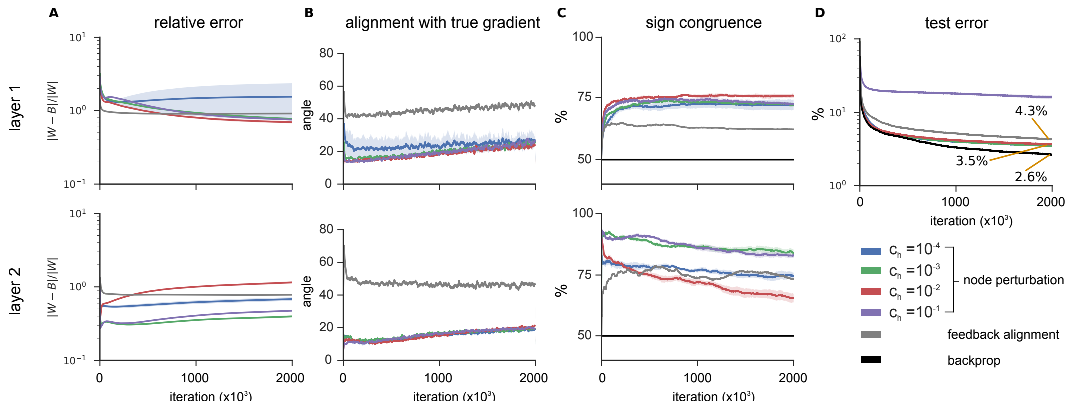

Figure 2: Node perturbation in small 4-layer network (784-50-20-10 neurons), for varying noise levels c, compared to feedback alignment and backpropagation. (A) Relative error between feedforward and feedback matrix. (B) Angle between true gradient and synthetic gradient estimate for each layer. (C) 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.

Theorem 2. *Assume A1-4. For $\sigma(x) = x$, the least squares estimator*

$$(\hat{B}^n)^T := \hat{\lambda}^{n-1}(\tilde{\mathbf{e}}^n)^T \left(\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T\right)^{-1} \qquad 1 \le n \le N+1, \tag{5}$$

solves (8) *and converges to the true feedback matrix, in the sense that:*

$$\lim_{c_h \to 0} \operatorname*{plim}_{T\to\infty} \hat{B}^n = W^n, \qquad 1 \le n \le N+1.$$

Proofs and a discussion of the assumptions are provided in the supplementary material.

Thus either for a non-linear shallow network, or a deep linear network, we have the result that, for sufficiently small c_h, if we fix the network weights W and train B through node perturbation then we converge to W. Validation that the method learns to approximate W, for fixed W, is provided in the supplementary material. In practice, we update B and W simultaneously. Some convergence theory is established for this case in [24, 10].

4 Applications

4.1 Solving MNIST

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 (Figure 2). Updates to W^i are made using the synthetic gradients

$$\Delta W^i = \eta \tilde{\mathbf{e}}^i \mathbf{h}^{i-1},$$

for learning rate η. The feedback network needs to co-adapt with the feedforward network in order to continue to provide a useful error signal. We observed that the system is able to adjust to provide a close correspondence between the feedforward and feedback matrices in both layers of the network (Figure 2A).

We observed that the relative error between B_i and W_i is lower than what is observed for feedback alignment, suggesting that this co-adaptation of both W_i and B_i is indeed beneficial. The relative error depends on the amount of noise used in node perturbation – lower variance doesn't necessarily imply the lowest error between W and B, suggesting there is an optimal noise level that balances bias in the estimate and the ability to co-adapt to the changing feedforward weights.

Consistent with the low relative error in both layers, 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 low in each layer – much lower for node perturbation than for feedback alignment, again suggesting that the method is much better at communicating error signals between layers (Figure 2B). In fact, recent studies have shown

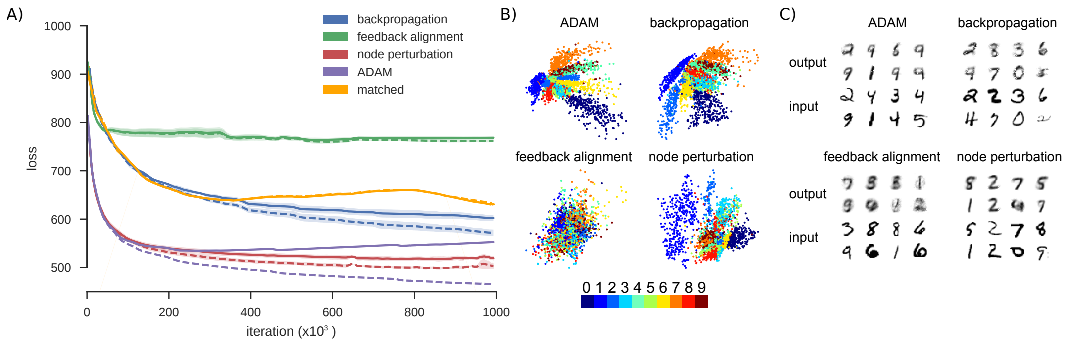

Figure 3: Results with five-layer MNIST autoencoder network. a) Mean loss plus/minus standard error over 10 runs. Dashed lines represent training loss, solid lines represent test loss. b) Latent space activations, colored by input label for each method. c) Sample outputs for each method.

that sign congruence of the feedforward and feedback matrices is all that is required to achieve good performance [34, 58]. Here the sign congruence is also higher in node perturbation, again depending somewhat the variance. The amount of congruence is comparable between layers (Figure 2C).

Finally, the learning performance of node perturbation is comparable to backpropagation (Figure 2D) – achieving close to 3% test error. It is better than feedback alignment in this case. The same learning rate was used for all experiments here, and was not optimized individually for each method. Thus this result is not indicative of the superior performance of one method over the other – all methods do converge, and each likely could be optimized to converge faster. These results instead highlight the qualitative differences between the methods. They suggest node perturbation for learning to learn can be used in deep networks.

4.2 Auto-encoder

The above results demonstrate node perturbation provides error signals closely aligned with the true gradients. However, performance-wise they do not demonstrate any clear advantage over feedback alignment or backpropagation in this small network. A known shortcoming of feedback alignment is in very deep networks and in autoencoding networks with tight bottleneck layers [35]. To see if node perturbation has the same shortcoming, we test performance on a simple auto-encoding network with MNIST input data (size 784-200-2-200-784). In this more challenging case we also compare the method to the 'matching' learning rule [48, 39], in which updates to B match updates to W.

As expected, feedback alignment performs poorly, while node perturbation performs better than backpropagation and comparable ADAM (Figure 3a). In fact ADAM begins to overfit in training, while node perturbation does not. The increased performance relative to backpropagation is surprising. It may be a similar effect to that speculated to explain feedback alignment – the method strikes the right balance between providing a useful gradient signal to learn, and constraining the updates to be sufficiently aligned with B, acting as a type of regularization [35]. Th matched learning rule performs similarly to backpropagation. In line with these results, the latent space (bottleneck layer) learnt by node perturbation shows a useful separation between the digits, as do the networks trained by backpropagation and ADAM. In contrast, feedback alignment does not learn to separate digits in the bottleneck layer (Figure 3b). This results in scrambled output (Figure 3c). These results show that node perturbation is able to successfully communicate error signals through thin layers of a network as needed.

4.3 Recurrent networks

Node perturbation can also be applied to approximate gradients in recurrent networks. We demonstrate this with a network setup as

$$\mathbf{h}_t = \sigma(W\mathbf{h}_{t-1} + U\mathbf{x}_t),$$

with output

$$\hat{\mathbf{y}}_t = \sigma(V\mathbf{h}_t).$$

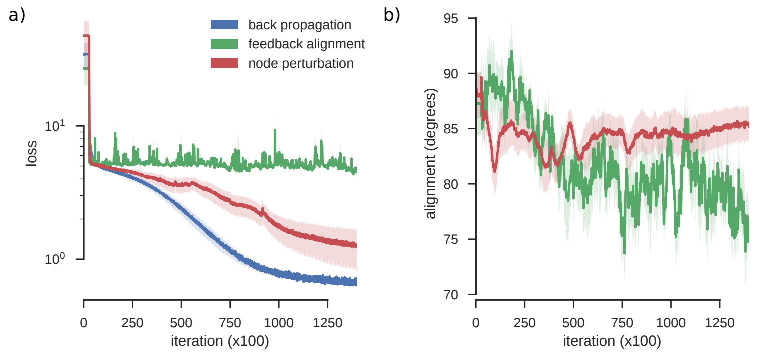

Figure 4: Delayed XOR application. (A) Loss of backpropagation, feedback alignment and node perturbation method. (B) Alignment between true gradient and approximated gradient for feedback alignment and node perturbation. Curves show mean plus and minus standard error over 10 runs.

Node perturbation is applied as in the feedforward setting to generate estimates of the gradient for updating W and U. Truncated BPTT of length T is used to propagate the error signal $\tilde{\mathrm{e}}$ from time $t+T$ to t. Here 50 hidden units are used, and $T = 10$. While long term dependencies are challenging for vanilla RNNs [6], we used a vanilla RNN as a simple demonstration of node perturbation. Other architectures such as LSTMs may be used to improve performance [20].

We test the method on a delayed XOR task. Here, at random times a go cue is given, $x_{2,t} = 1$, and two random binary inputs $x_1 \in \{-1, 1\}$ are generated. With delay τ the network is required to output $y_{t+1} = \text{xor}(x_{1,t-\tau}, x_{1,t-\tau-1})$. Here $\tau = 3$. With node perturbation the network is able to learn to perform this task (Figure 4 A), converging in a time comparable to backpropagation through time. In this setup feedback alignment does not converge. Averaged over all layers (times unrolled) the angles of alignment to the true gradient are both near ninety, suggesting only a weak relation to the true error signals (Figure 4B). Regardless, the learned error signal is useful enough for node perturbation to solve the task.

5 Discussion

Here we implement a perturbation-based synthetic gradient method to train neural networks. We show that this hybrid approach can solve both feedforward and recurrent tasks. By removing both the symmetric forward backward weight requirement and the update locking 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 problems at a large scale [59, 17, 18, 52, 11]. Moreover, recently proposed causal estimation techniques [28] promise to provide lower variance estimators than node perturbation. We thus believe this type of learning to learn approach may ultimately provide a powerful and biologically plausible learning algorithm.

Computationally, the method has a number of benefits. First, training of the forward and backward weights may be performed separately, and hence the forward and backward pass of backpropagation may be performed asynchronously. Thus the method may be applied on distributed systems in which synchronization is difficult or time consuming [10, 9], including some integrated circuits [1]. Second, by relying on random perturbations to measure gradients, the method does not rely on the environment to provide gradients. It works in cases common in reinforcement learning, where gradients in the environment cannot be backpropagated through. And third, the method is mathematically quite straightforward, facilitating analysis. This allows us to provide proofs of convergence in some special cases.

These proofs extend the theory of synthetic gradients and feedback alignment. Previous results in synthetic gradients prove convergence in a deep linear networks with MSE loss [10], with a synthetic gradient module in a single layer. Here, by assuming smoothness and sufficiently concentrated noise, we are able extend these results somewhat. Further, proof of convergence for more general synthetic gradient function choices, g, may be possible using the same ideas presented here.

As a REINFORCE-style estimator, it may seem corresponding theory would be relevant – the REINFORCE estimator provides an unbiased estimate of the loss gradient of a non-linear, noisy system [57]. However, here we define synthetic gradients in terms of the system without noise. So REINFORCE theory cannot be used directly (though perhaps results with a noisy baseline are possible, e.g. [17]). Our results instead show that, as the noise tends to zero, the REINFORCE-style estimator can be used to estimate the true parameters of the deterministic system. As pointed out in [10], synthetic gradient methods are closely related in form and motivation to actor-critic methods. Thus it is likely further ideas from reinforcement learning could provide additional theory and insight.

While previous research has provided some insight and theory for how feedback alignment works [35, 45, 43, 4, 3], 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 [34, 58]. Yet what mechanism may produce congruence in biological networks has not been addressed. Instead, here we show that some of the shortcomings of feedback alignment can be addressed in another way – the system can adjust the weights as needed to provide a useful error signal. While we have just investigated one choice of g function in which the approach directly approximates backpropagation, the theory may be extended easily to other forms of g. Our work is closely related to a recent proposal from 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. A future combination of the two approaches may prove fruitful.

The method does have some drawbacks. Our approach does not, by itself, reach state-of-the-art performance in common benchmarks like CIFAR or ImageNet, which would require convolutional networks. Further, as implemented here, distortion of the error signal does accumulate in a layer-wise fashion, from top to bottom. This means it is unlikely to be a practical approach to learning in very deep networks. It is possible these drawbacks can be addressed to some extent, for example by using direct feedback alignment [44] to produce a system in which convergence does not proceed layer by layer. Recent studies have shown that shallow spiking networks can competitively solve problems often tackled with deep networks [23]. Further, noise injection may be replaced with an estimate of the effect on a cost function that doesn't require the injection of noise [28]. Thus perhaps some of these drawbacks can be mitigated.

How to solve the credit assignment problem remains a challenge not just in biological networks. In artificial networks, training a system that can learn long term dependencies is difficult. Synthetic gradients show how a method can be trained to solve a problem beyond its truncated BPTT horizon. Yet these have not been demonstrated to solve very long-term dependencies. Recent research has thus focused on the notion of attention to bridge long time spans [22, 25, 5]. Other recent work has shown how recurrent networks can be trained in an online fashion [8], in an approach that can be seed as making REINFORCE-type updates with small noise perturbations. This may be related to our framework, and this is the subject of future work.

Though we are interested in biologically plausible learning, our method is at the computational and algorithmic level: it operates within constraints consistent with neurobiology, but does not specify exactly how it may be implemented. Rather, we focused on theoretical analysis and testing the method in an idealized setting. In a similar fashion, feedback alignment was first analyzed in an artificial network setting and now forms a part of some biologically plausible models of learning in cortex [15]. We thus believe this work is an important first step before more detailed models are considered.

Notably, however, the method is consistent with neurobiology in two important ways. First, it involves separate learning of feedforward and feedback weights. This is possible in cortical networks, where complex feedback connections exist between layers [27, 47] and pyramidal cells have apical and basal compartments that allow for separate integration of feedback and feedforward signals [15]. A recent finding that apical dendrites receive reward information is particularly interesting [27]. Models like Guergiev et al 2017 [15] are thus quite compelling. We believe such models can be augmented with a perturbation-based rule like ours to provide a better learning system. The second feature is that perturbations are used to learn the feedback weights. How can a neuron measure these perturbations? There are many plausible mechanisms [52, 59, 12, 11]. For instance, birdsong learning uses 'empiric synapses' from area LMAN [11], others proposed it is approximated [32, 21], or neurons could use a learning rule that does not require knowing the noise [28]. Further, 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 replaced with a approximation [32, 36]. Thus both separation of feedforward and feedback systems and perturbation-based estimators can be implemented by neurons.

Learning to learn is a powerful mechanism not just to learn efficient learning rules, but also to learn rules that generalize well to new data on the basis of common structure [55, 37, 2]. Its potential to provide realistic accounts of efficient learning in the brain is only just beginning to be explored.

References

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

[2] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, and Matthew W Hoffman. Learning to learn by gradient descent by gradient descent. *Adv. Neural Inf. Process. Syst.*, (Nips):1–17, 2016.

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

[4] 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.

[5] Yoshua Bengio. The Consciousness Prior. *ArXiv e-prints*, (1):1–4, 2017.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult, 1994.

[7] Guy Bouvier, Claudia Clopath, Célian Bimbard, Jean-Pierre Nadal, Nicolas Brunel, Vincent Hakim, and Boris Barbour. Cerebellar learning using perturbations. *bioRxiv*, page 053785, 2016.

[8] Tim Cooijmans and James Martens. On the Variance of Unbiased Online Recurrent Optimization. *ArXiv e-prints*, 2019.

[9] Brian Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct Feedback Alignment with Sparse Connections for Local Learning. *ArXiv e-prints*, pages 1–13, 2019.

[10] Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding Synthetic Gradients and Decoupled Neural Interfaces. *ArXiv e-prints*, 2017.

[11] 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.

[12] Ila R Fiete and H Sebastian Seung. Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Physical Review Letters*, 97, 2006.

[13] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility Traces and Plasticity on Behavioral Time Scales : Experimental Support of neoHebbian Three-Factor Learning Rules. *ArXiv e-prints*, pages 1–23, 2018.

[14] Jordan Guergiuev, Timothy P. Lillicrap, and Blake A. Richards. Towards deep learning with segregated dendrites. *eLife*, 6:1–37, 2017.

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

[16] H A Haenssle, C Fink, R Schneiderbauer, F Toberer, T Buhl, A Blum, A Kalloo, A Ben Hadj Hassen, L Thomas, A Enk, L Uhlmann, and Reader study level-I and level-II Groups. Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Ann. Oncol.*, 29(8):1836–1842, August 2018.

[17] Kazuyuki Hara, Kentaro Katahira, and Masato Okada. Statistical mechanics of node-perturbation learning with noisy baseline. *Journal of the Physical Society of Japan*, 86(2):1–7, 2017.

[18] Kazuyuki Hara, Kentaro Katahira, Kazuo Okanoya, and Masato Okada. Statistical Mechanics of On-line Node-perturbation Learning. *Information processing society of Japan*, 4:23–32, 2011.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing Human-Level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[20] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1–32, 1997.

[21] Gregor M. Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cerebral Cortex*, 24(3):677–690, 2014.

[22] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing Agent Behavior over Long Time Scales by Transporting Value. *ArXiv e-prints*, 8:1–60, 2018.

[23] Bernd Illing, Wulfram Gerstner, and Johanni Brea. Biologically plausible deep learning – but how far can we go with shallow networks ? *ArXiv e-prints*, 2019.

[24] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients. *ArXiv e-prints*, 1, 2016.

[25] Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7651–7662, 2018.

[26] Konrad Kording and Peter Konig. Supervised and Unsupervised Learning with Two Sites of Synaptic Integration. *Journal of Computational Neuroscience*, 11:207–215, 2001.

[27] Clay O Lacefield, Eftychios A Pnevmatikakis, Liam Paninski, and Randy M Bruno. Reinforcement Learning Recruits Somata and Apical Dendrites across Layers of Primary Sensory Cortex. *Cell Reports*, 26(8):2000–2008.e2, 2019.

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

[29] Benjamin James Lansdell and Konrad Paul Kording. Towards learning-to-learn. pages 1–8, 2018.

[30] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[31] Dong Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9284:498–515, 2015.

[32] 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.

[33] Marco Lehmann, He Xu, Vasiliki Liakoni, Michael Herzog, Wulfram Gerstner, and Kerstin Preuschoff. Evidence for eligibility traces in human learning. *ArXiv e-prints*, pages 2–7, 2017.

[34] Qianli Liao, Joel Z. Leibo, and Tomaso Poggio. How Important is Weight Symmetry in Backpropagation? 2015.

[35] 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.

[36] Y. Loewenstein and H. S. Seung. Operant matching is a generic outcome of synaptic plasticity based on the covariance between reward and neural activity. *Proceedings of the National Academy of Sciences*, 103(41):15224–15229, 2006.

[37] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

[38] David Marr. A theory of cerebellar cortex. *J. Physiol*, 202:437–470, 1969.

[39] 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.

[40] Thomas Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6:1–24, 2017.

[41] Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic neural networks with backpropagation. *ArXiv e-prints*, 2018.

[42] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[43] Theodore H. Moskovitz, Ashok Litwin-kumar, and L.f. Abbott. Feedback alignment in deep convolutional networks. *arXiv Neural and Evolutionary Computing*, pages 1–10, 2018.

[44] Arild Nøkland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. *NIPS*, (Nips), 2016.

[45] Alexander G. Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles. Conducting Credit Assignment by Aligning Local Representations. *ArXiv e-prints*, pages 1–27, 2018.

[46] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of the 31st International Conference on Machine Learning, PMLR*, 32(2):1278–1286, 2014.

[47] Blake A Richards and Timothy P Lillicrap. Dendritic solutions to the credit assignment problem. *Current Opinion in Neurobiology*, 54:28–36, 2019.

[48] Jaldert O Rombouts, Sander M Bohte, and Pieter R Roelfsema. How Attention Can Create Synaptic Tags for the Learning of Working Memories in Sequential Tasks. *PLoS Computational Biology*, 11(3):1–34, 2015.

[49] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[50] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[51] Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap Between Energy-Based Models and Backpropagation. *arXiv*, 11(1987):1–13, 2016.

[52] Sebastian Seung. Learning in Spiking Neural Networks by Reinforcement of Stochastics Transmission. *Neuron*, 40:1063–1073, 2003.

[53] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[54] H Francis Song, Guangyu R Yang, and Xiao Jing Wang. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *eLife*, 6:1–24, 2017.

[55] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[56] Paul Werbos. Approximate dynamic programming for real-time control and neural modeling. In *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, chapter 13. Multiscience Press, Inc., New York, 1992.

[57] Ronald Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8:299–256, 1992.

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

[59] Xiaohui Xie and H. Sebastian Seung. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69, 2004.

Supplementary material

A Validation with fixed W

We demonstrate the method's convergence in a small non-linear network solving MNIST for different noise levels, c_h, and layer widths (Supplementary Figure 5). As basic validation of the method, in this experiment the feedback matrices are updated while the feedforward weights W^i are held fixed. In contrast to the text, to best understand how close the method can get to getting B to approximate W, we used the exact ridge regression solution to update B:

$$
\begin{aligned}
V_t &= V_{t-1} + \mathbf{e}_t^T \mathbf{e}_t, \qquad V_0 = \gamma \mathbb{I}, \\
S_t &= S_{t-1} + \lambda_t^T \mathbf{e}_t, \\
\hat{B}_t &= (V_t)^{-1} S_t,
\end{aligned}
$$

with identity matrix $\mathbb{I}$ and regularization parameter $\gamma > 0$.

We should expect the feedback matrices B^i to converge to the feedforward matrices W^i. Here different noise variance does results equally accurate estimators (Supplementary Figure 5A). The estimator correctly estimates the true feedback matrix W^2 to a relative error of 0.8%. The convergence is layer dependent, with the second hidden layer matrix, W_2, being accurately estimated, and the convergence of the first hidden layer matrix, W_1, being less accurately estimated. Despite this, the angles between the estimated gradient and the true gradient (proportional to $\mathbf{e}^T W B^T \tilde{\mathbf{e}}$) are very close to zero for both layers (Supplementary Figure 5B) (less than 90 degrees corresponds to a descent direction). Thus the estimated gradients strongly align with true gradients in both layers. Recent studies have shown that sign congruence of the feedforward and feedback matrices is all that is required to achieve good performance [34, 58]. Here significant sign congruence is achieved in both layers (Supplementary Figure 5C), despite the matrices themselves being quite different in the first layer. The number of neurons has an effect on both the relative error in each layer and the extent of alignment between true and synthetic gradient (Supplementary Figure 5D,E). The method provides useful error signals for a variety of sized networks, and can provide useful error information to layers through a deep network.

With fixed W, only the top layer feedforward and feedback matrices were in close correspondence (compare with Figure 2, which shows both layers converge as well). Thus it seems that in the co-adapting case, a similar effect to feedback alignment may be occurring – the feedforward matrices adapt to the feedback matrices to allow for a more useful error signal to propagate to deeper layers and allow for greater correspondence between W_i and B_i throughout the network than what occurs with fixed W_i.

B Proofs

We review the key components of the model. Data $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ are drawn from a distribution ρ. The loss function is linearized:

$$
\tilde{\mathcal{L}} \approx \mathcal{L} + \frac{\partial \mathcal{L}}{\partial h_j^i} c_h \xi_j^i, \tag{6}
$$

such that

$$
\mathbb{E}\left((\tilde{\mathcal{L}} - \mathcal{L}) c_h \xi_j^i | \mathbf{x}, \mathbf{y} \right) \approx c_h^2 \frac{\partial \mathcal{L}}{\partial h_j^i}\bigg|_{\mathbf{x},\mathbf{y}},
$$

with expectation taken over the noise distribution $\nu(\xi)$. This suggests a good estimator of the loss gradient is

$$
\hat{\lambda}^i := (\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}, \xi) - \mathcal{L}(\mathbf{x}, \mathbf{y})) \frac{\xi^i}{c_h}. \tag{7}
$$

Let $\tilde{\mathbf{e}}^i$ be the error signal computed by backpropagating the synthetic gradients:

$$
\tilde{\mathbf{e}}^i = \begin{cases} \partial \mathcal{L} / \partial \hat{\mathbf{y}} \circ \sigma'(W^i \mathbf{h}^{i-1}), & i = N+1; \\ \left((\hat{B}^{i+1})^T \tilde{\mathbf{e}}^{i+1} \right) \circ \sigma'(W^i \mathbf{h}^{i-1}), & 1 \le i \le N. \end{cases}
$$

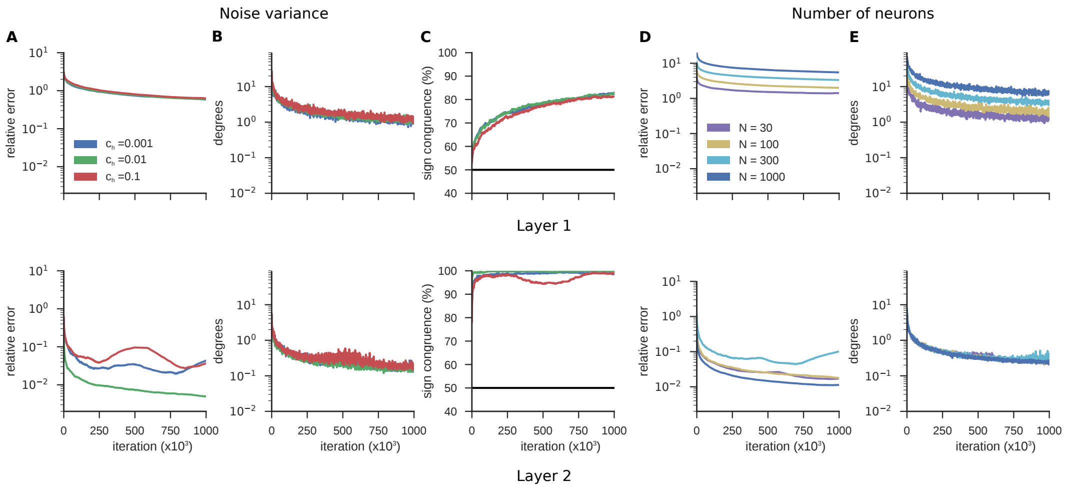

Figure 5: Convergence of node perturbation method in a two hidden layer neural network (784-50-20-10) with MSE loss, for varying noise levels c. Node perturbation is used to estimate feedback matrices that provide gradient estimates for fixed W. (A) Relative error ($\|W_i - B_i\|_F/\|W_i\|_F$) for each layer. (B) Angle between true gradient and synthetic gradient estimate at each layer. (C) Percentage of signs in W_i and B_i that are in agreement. (D) Relative error when number of neurons is varied (784-N-50-10). (E) Angle between true gradient and synthetic gradient estimate at each layer.

Then parameters B^{i+1} are estimated by solving the least squares problem:

$$\hat{B}^{i+1} = \text{argmin}_B \mathbb{E} \left\| B^T \tilde{\mathbf{e}}^{i+1} - \hat{\lambda}^i \right\|_2^2. \tag{8}$$

Under what conditions can we show that $\hat{B}^{i+1} \to W^{i+1}$ (with enough data)?

One way to find an answer is to define the synthetic gradient in terms of the system without noise added. Then $B^T \tilde{\mathbf{e}}$ is deterministic with respect to $\mathbf{x}, \mathbf{y}$ and, assuming $\tilde{\mathcal{L}}$ has a convergent power series around $\xi = 0$, we can write

$$\begin{aligned}\mathbb{E}(\hat{\lambda}^i|\mathbf{x},\mathbf{y}) &= \mathbb{E}\left(\frac{1}{c_h^2}\left[\frac{\partial \mathcal{L}}{\partial h^i}(c_h\xi_j^i)^2 + \sum_{m=2}^{\infty}\frac{\mathcal{L}_{ij}^{(m)}}{m!}(c_h\xi_j^i)^{m+1}\right]|\mathbf{x},\mathbf{y}\right)\\ &= (W^{i+1})^T\mathbf{e}^{i+1} + \mathbb{E}\left(\frac{1}{c_h^2}\sum_{m=2}^{\infty}\frac{\mathcal{L}_{ij}^{(m)}}{m!}(c_h\xi_j^i)^{m+1}|\mathbf{x},\mathbf{y}\right).\end{aligned}$$

Taken together these suggest we can prove $\hat{B}^{i+1} \to W^{i+1}$ in the same way we prove consistency of the linear least squares estimator.

For this to work we must show the expectation of the Taylor series approximation (6) is well behaved. That is, we must show the expected remainder term of the expansion:

$$\mathcal{E}_j^i(c_h) = \mathbb{E}\left[\frac{1}{c_h^2}\sum_{m=2}^{\infty}\frac{\mathcal{L}_{ij}^{(m)}}{m!}(c_h\xi_j^i)^{m+1}|\mathbf{x},\mathbf{y}\right],$$

is finite and goes to zero as $c_h \to 0$. This requires some additional assumptions on the problem.

We make the following assumptions:

- A1: the noise ξ is subgaussian,
- A2: the loss function $\mathcal{L}(\mathbf{x},\mathbf{y})$ is analytic on $\mathcal{D}$,
- A3: the error matrices $\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T$ are full rank, for $1 \le n \le N+1$,

- A4: the mean of the remainder and error terms is bounded:
$$\mathbb{E}\left[\mathcal{E}^n(c_h)(\tilde{\mathbf{e}}^{n+1})^T\right] < \infty,$$
for $1 \le n \le N$.

Consider first convergence of the final layer feedback matrix, B^{N+1}. In the final layer it is true that $\mathbf{e}^{N+1} = \tilde{\mathbf{e}}^{N+1}$.

Theorem 3. *Assume A1-4. Then the least squares estimator*
$$(\hat{B}^{N+1})^T := \hat{\lambda}^N(\mathbf{e}^{N+1})^T \left(\mathbf{e}^{N+1}(\mathbf{e}^{N+1})^T\right)^{-1}, \tag{9}$$
solves (8) *and converges to the true feedback matrix, in the sense that:*
$$\lim_{c_h \to 0} \operatorname*{plim}_{T\to\infty} \hat{B}^{N+1} = W^{N+1}.$$

Proof. Let $\mathcal{L}^{(m)}_{ij} := \frac{\partial^m \mathcal{L}}{\partial h_j^{im}}$. We first show that, under A1-2, the conditional expectation of the estimator (7) converges to the gradient $\mathcal{L}^{(1)}_{Nj}$ as $c_h \to 0$. For each $\hat{\lambda}^N_j$, by A2, we have the following series expanded around $\xi = 0$:
$$\hat{\lambda^N_j} = \frac{1}{c_h^2}\sum_{m=1}^{\infty} \frac{\mathcal{L}^{(m)}_{ij}}{m!}(c_h\xi^N_j)^{m+1}.$$
Taking a conditional expectation gives:
$$\mathbb{E}(\hat{\lambda}^N_j|\mathbf{x},\mathbf{y}) =(W^{N+1})^T\mathbf{e}^{N+1} + \mathbb{E}\left[\frac{1}{c_h^2}\sum_{m=2}^{\infty}\frac{\mathcal{L}^{(m)}_{Nj}}{m!}(c_h\xi^N_j)^{m+1}|\mathbf{x},\mathbf{y}\right].$$
We must show the remainder term
$$\mathcal{E}^N(c_h) = \mathbb{E}\left[\frac{1}{c_h^2}\sum_{m=2}^{\infty}\frac{\mathcal{L}^{(m)}_{Nj}}{m!}(c_h\xi^N_j)^{m+1}|\mathbf{x},\mathbf{y}\right],$$
goes to zero as $c_h \to 0$. This is true provided each moment $\mathbb{E}((\xi^N_j)^m|\mathbf{x},\mathbf{y})$ is sufficiently well-behaved. Using Jensen's inequality and the triangle inequality in the first line, we have that
$$\begin{aligned}
\left|\mathcal{E}^N(c_h)\right| &\le \mathbb{E}\left[\frac{1}{c_h^2}\sum_{m=2}^{\infty}\left|\frac{\mathcal{L}^{(m)}_{Nj}}{m!}\right| |c_h\xi^N_j|^{m+1}|\mathbf{x},\mathbf{y}\right], \quad \forall(\mathbf{x},\mathbf{y})\in\mathcal{D}\\
\text{[monotone convergence]} \quad &= \sum_{m=2}^{\infty}\left|\frac{\mathcal{L}^{(m)}_{Nj}}{m!}\right|(c_h)^{m-1}\mathbb{E}\left[|\xi^N_j|^{m+1}\right]\\
\text{[subgaussian]} \quad &\le K\sum_{m=2}^{\infty}\left|\frac{\mathcal{L}^{(m)}_{Nj}}{m!}\right|(c_h)^{m-1}(\sqrt{m+1})^{m+1}\\
&= \mathcal{O}(c_h) \qquad \text{as } c_h \to 0.
\end{aligned} \tag{10}$$

With this in place, we have that the problem (8) is close to a linear least squares problem, since
$$\hat{\lambda}^N = (W^{N+1})^T\mathbf{e}^{N+1} + \mathcal{E}^N(c_h) + \eta^N, \tag{11}$$
with residual $\eta^N = \hat{\lambda}^N - \mathbb{E}(\hat{\lambda}^N|\mathbf{x},\mathbf{y})$. The residual satisfies
$$\begin{aligned}
\mathbb{E}\left(\mathbf{e}^{N+1}(\eta^N)^T\right) &= \mathbb{E}(\mathbf{e}^{N+1}(\hat{\lambda}^N)^T - \mathbf{e}^{N+1}\mathbb{E}((\hat{\lambda}^N)^T|\mathbf{x},\mathbf{y}))\\
&= \mathbb{E}\left(\mathbf{e}^{N+1}(\hat{\lambda}^N)^T - \mathbb{E}\left(\mathbf{e}^{N+1}(\hat{\lambda}^N)^T|\mathbf{x},\mathbf{y}\right)\right)\\
&= 0.
\end{aligned} \tag{12}$$

This follows since $\mathbf{e}^{N+1}$ is defined in relation to the baseline loss, not the stochastic loss, meaning it is measurable with respect to $(\mathbf{x},\mathbf{y})$ and can be moved into the conditional expectation.

From (11) and A3, we have that the least squares estimator (9) satisfies

$$(\hat{B}^{N+1})^T = (W^{N+1})^T + (\mathcal{E}^N(c_h) + \eta^N)(\mathbf{e}^{N+1})^T(\mathbf{e}^{N+1}(\mathbf{e}^{N+1})^T)^{-1}.$$

Thus, using the continuous mapping theorem

$$\begin{aligned}
\operatorname*{plim}_{T\to\infty}(\hat{B}^{N+1})^T &= (W^{N+1})^T + \left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}(\mathcal{E}^N(c_h)+\eta^N)(\mathbf{e}^{N+1})^T\right]\left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\mathbf{e}^{N+1}(\mathbf{e}^{N+1})^T\right]^{-1} \\
\text{[WLLN]} \quad &= (W^{N+1})^T + \mathbb{E}\left[(\mathcal{E}(c_h)+\eta^N)(\mathbf{e}^{N+1})^T\right]\left[\mathbb{E}(\mathbf{e}^{N+1}(\mathbf{e}^{N+1})^T)\right]^{-1} \\
\text{[Eq. (12)]} \quad &= (W^{N+1})^T + \mathbb{E}\left[\mathcal{E}(c_h)(\mathbf{e}^{N+1})^T\right]\left[\mathbb{E}(\mathbf{e}^{N+1}(\mathbf{e}^{N+1})^T)\right]^{-1} \\
\text{[A4 and Eq. (10)]} \quad &= (W^{N+1})^T + \mathcal{O}(c_h).
\end{aligned}$$

Then we have:

$$\lim_{c_h\to 0}\operatorname*{plim}_{T\to\infty}\hat{B}^{N+1} = W^{N+1}.$$

□

We can use Theorem 1 to establish convergence over the rest of the layers of the network when the activation function is the identity.

Theorem 4. *Assume A1-4. For $\sigma(x) = x$, the least squares estimator*

$$(\hat{B}^n)^T := \hat{\lambda}^{n-1}(\tilde{\mathbf{e}}^n)^T\left(\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T\right)^{-1} \qquad 1 \le n \le N+1, \tag{13}$$

solves (8) *and converges to the true feedback matrix, in the sense that:*

$$\lim_{c_h\to 0}\operatorname*{plim}_{T\to\infty}\hat{B}^n = W^n, \qquad 1 \le n \le N+1.$$

Proof. Define

$$\tilde{W}^n(c) := \operatorname*{plim}_{T\to\infty}\hat{B}^n,$$

assuming this limit exists. From Theorem 1 the top layer estimate $\hat{B}^{N+1}$ converges in probability to $\tilde{W}^{N+1}(c)$.

We can then use induction to establish that $\hat{B}^j$ in the remaining layers also converges in probability to $\tilde{W}^j(c)$. That is, assume that $\hat{B}^j$ converge in probability to $\tilde{W}^j(c)$ in higher layers $N+1 \ge j > n$. Then we must establish that $\hat{B}^n$ also converges in probability.

To proceed it is useful to also define

$$\tilde{\tilde{\mathbf{e}}}(c)^n := \begin{cases} \partial\mathcal{L}/\partial\hat{\mathbf{y}} \circ \sigma'(W^i\mathbf{h}^{i-1}), & i = N+1; \\ \left((\tilde{W}^{i+1}(c))^T\tilde{\tilde{\mathbf{e}}}^{i+1}\right)\circ\sigma'(W^i\mathbf{h}^{i-1}), & 1 \le i \le N, \end{cases}$$

as the error signal backpropagated through the converged (but biased) weight matrices $\tilde{W}(c)$. Again it is true that $\tilde{\tilde{\mathbf{e}}}^{N+1} = \mathbf{e}^{N+1}$.

As in Theorem 1, the least squares estimator has the form:

$$(\hat{B}^n)^T = \hat{\lambda}^{n-1}(\tilde{\mathbf{e}}^n)^T\left(\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T\right)^{-1}.$$

Thus, again by the continuous mapping theorem:

$$\begin{aligned}
\operatorname*{plim}_{T\to\infty}(\hat{B}^n)^T &= \left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\hat{\lambda}^{n-1}(\tilde{\mathbf{e}}^n)^T\right]\left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T\right]^{-1} \\
&= \left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\hat{\lambda}^{n-1}(\mathbf{e}^{N+1})^T\hat{B}^{N+1}\cdots\hat{B}^{n+1}\right]\left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T\right]^{-1}
\end{aligned}$$

In this case continuity again allows us to separate convergence of each term in the product:

$$\begin{aligned}\operatorname*{plim}_{T\to\infty}\frac{1}{T}\hat{\lambda}^{n-1}(\mathbf{e}^{N+1})^T\hat{B}^{N+1}\cdots\hat{B}^{n+1} &= \left[\operatorname*{plim}_{T\to\infty}\frac{1}{T}\hat{\lambda}^{n-1}(\mathbf{e}^{N+1})^T\right]\left[\operatorname*{plim}_{T\to\infty}\hat{B}^{N+1}\right]\cdots\left[\operatorname*{plim}_{T\to\infty}\hat{B}^{n+1}\right] \qquad (14)\\ &= \mathbb{E}(\hat{\lambda}^{n-1}(\mathbf{e}^{N+1})^T)W^{N+1}(c)\cdots W^{n+1}(c),\\ &= \mathbb{E}(\hat{\lambda}^{n-1}(\tilde{\tilde{\mathbf{e}}}^n(c))^T)\end{aligned}$$

using the weak law of large numbers in the first term, and the induction assumption for the remaining terms. In the same way

$$\operatorname*{plim}_{T\to\infty}\frac{1}{T}\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T = \mathbb{E}(\tilde{\tilde{\mathbf{e}}}^n(c)(\tilde{\tilde{\mathbf{e}}}^n(c))^T).$$

Note that the induction assumption also implies $\lim_{c\to 0}\tilde{\tilde{\mathbf{e}}}^n(c) = \mathbf{e}^n$. Thus, putting it together, by A3, A4 and the same reasoning as in Theorem 3 we have the result:

$$\begin{aligned}\lim_{c_h\to 0}\operatorname*{plim}_{T\to\infty}(\hat{B}^n)^T &= \lim_{c\to 0}\left[(W^n)^T\mathbb{E}(\mathbf{e}^n(\tilde{\tilde{\mathbf{e}}}^n(c))^T) + \mathbb{E}(\mathcal{E}^{n-1}(c)(\tilde{\tilde{\mathbf{e}}}^n(c))^T\right]\left[\mathbb{E}(\tilde{\tilde{\mathbf{e}}}^n(c)(\tilde{\tilde{\mathbf{e}}}^n(c))^T)\right]^{-1}\\ &= (W^n)^T.\end{aligned}$$

□

B.1 Discussion of assumptions

It is worth making the following points on each of the assumptions:

- A1. In the paper we assume ξ is Gaussian. Here we prove the more general result of convergence for any subgaussian random variable.
- A2. In practice this may be a fairly restrictive assumption, since it precludes using relu non-linearities. Other common choices, such as hyperbolic tangent and sigmoid non-linearities with an analytic cost function do satisfy this assumption, however.
- A3. It is hard to establish general conditions under which $\tilde{\mathbf{e}}^n(\tilde{\mathbf{e}}^n)^T$ will be full rank. While it may be a reasonable assumption in some cases.

Extensions of Theorem 2 to a non-linear network may be possible. However, the method of proof used here is not immediately applicable because the continuous mapping theorem can not be applied in such a straightforward fashion as in Equation (14). In the non-linear case the resulting sums over all observations are neither independent or identically distributed, which makes applying any law of large numbers complicated.

C Experiment details

Details of each task and parameters are provided here. All code is implemented in TensorFlow.

C.1 Supplementary Figure 1

Networks are 784-50-20-10 (noise variance) or 784-N-50-10 (number of neurons) solving MNIST with an MSE loss function. A sigmoid non-linearity is used. A batch size of 32 is used. Here W is fixed, and B is updated according to the online ridge regression least-squares solution. Regularization parameter $\gamma = 0.1$.

C.2 Figure 2

Networks are 784-50-20-10. Unless stated otherwise, assume same parameters as in Figure 2. Now W is updated using synthetic gradient updates with learning rate $\eta = 0.0005$. Same step size is used for feedback alignment, backpropagation and node perturbation.

C.3 Figure 3

Network has dimensions 784-200-2-200-784. Activation functions are, in order: tanh, identity, tanh, relu. MNIST input data with MSE reconstruction loss is used. Unless stated otherwise, assume same parameters as in Figure 2. Step size for node perturbation is $\eta = 0.0001$, and noise variance $c_h = 0.018$. In this case node perturbation performance was more stable for stochastic gradient updates to B, instead of the exact least squares solution. The step size of updates to B was $\nu = 8 \times 10^{-6}$. Values for step size, noise variance and ν were found by random hyperparameter search in the range of 10^{-6} to 10^{-3} for step size and ν, and between 10^{-4} and 10^{-1} for the noise variance.

C.4 Figure 4

Data is generated as a long continuous input stream $\mathbf{x}$ and expected output stream y. One epoch was defined as 50,000 time steps. BPTT was unrolled 7 time steps, a batch size of 20 was used. 50 hidden units are used, with a tanh activation function, and MSE loss function is used. Same step size was used for node perturbation, feedback alignment and backpropagation: $\eta = 5 \times 10^{-6}$. In this case node perturbation performance was more stable for stochastic gradient updates to B, instead of the exact least squares solution. The step size of updates to B was $\nu = 5 \times 10^{-5}$, and noise variance was $c_h = 0.5$. Values for noise variance and ν were found by random hyperparameter search.

View publication stats