Biologically Plausible Learning Rules

August 18, 2026

1 Gradient descent via back-propagation

Consider a feedforward network with L layers

$$a^0 = x \qquad\qquad a^{l+1} = W^l \phi(a^l) \qquad\qquad l = 0, \cdots, L-1$$

with output

$$\hat{y}_\theta(x) = a^L.$$

Denote all parameters of the network together by $\theta = \{W^l\}_{l=0}^{L-1}$. Given Data $\{x_\mu, y_\mu\}_\mu$, define an objective function / loss function

$$\mathcal{L}(\theta; x, y) = \frac{1}{2}||\hat{y}_\theta(x) - y||^2$$

and change weights according to gradient descent

$$\Delta\theta = -\eta \nabla_\theta \mathcal{L}(\theta; x, y).$$

Back-propagation (Rumelhart et al. 1986, Nature) is an effective way to calculate this for all weights recursively,

$$\frac{d\mathcal{L}}{dW^l_{jk}} = \underbrace{\frac{d\mathcal{L}}{da_i^{l+1}}}_{\equiv \delta_i^{l+1}} \frac{da_i^{l+1}}{dW^l_{jk}} = \delta_j^{l+1} \phi(a^l)_k$$

where

$$\delta_i^l := \frac{d\mathcal{L}}{da_i^l} = \frac{d\mathcal{L}}{da_j^{l+1}} \frac{da_j^{l+1}}{da_i^l} = \delta_j^{l+1} W^l_{ji} \phi'(a_i^l)$$

Together, for $l = 0, \cdots, L-1$,

$$\begin{aligned} \delta^L &= y - a^L \\ \delta^l &= (W^l)^T \delta^{l+1} \odot \phi'(a^l) \\ \Delta W^l &= \eta\, \delta^{l+1} \phi(a^l)^T \end{aligned}$$

Here $\odot$ is the point-wise (or Hadamard) product. One first calculates all $\{\delta^l\}_{l=0}^{L-1}$ and then updates the weights. The update actually looks like Hebbian learning in the sense of presynaptic activity $\phi(a^l)$ times postsynaptic activity δ^{l+1}. Here δ^l can be interpreted as the activations of a feed-back network, connected by $(W^l)^T$ to the the higher layer δ^{l+1}, and interacting point-wise with the (derivative of the) feed-forward activations $\phi'(a^l)$.

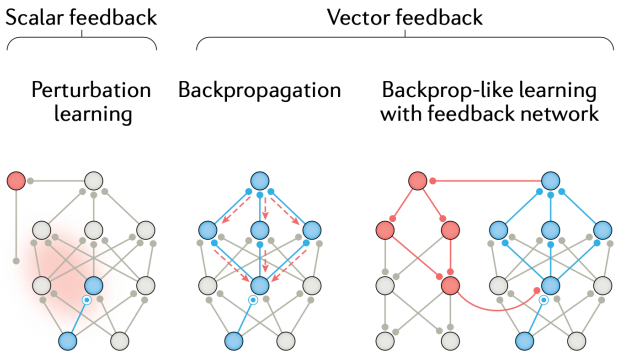

Figure 1: Illustration of different learning rules discussed here: Node perturbation, back propagation, back-propagation with a random feedback network

Does the brain do back-propagation? The problem of updating a synapse such that it contributes to an overall objective is called the "credit assignment problem". Here we consider a feed-forward network as a rate model for neurons. Then back propagation is one solution. But it does not seem very plausible from a biological point of view.:

- Weight transport problem: Backward feedback needs exactly the transpose of the forward weights $W^l_{\text{back}} = (W^l_{\text{forward}})^T$. It is unrealistic that the brain has an extra mechanism for maintaining this symmetry.
- Feedback of signed errors in deep networks, such as the cortex, appears problematic.
- Information delivered via δ only influences synaptic updates, but in the brain, feedback also alters activity
- Back-prop requires an alteration of a forward pass (to calculate all a^l) and a backward pass (to calculate all δ^l). It would appear more plausible that learning in the brain occurs continually without the need for alternative forward and backward passes.

In the following we relax some of the assumption of back-prop and see if it still works. We only consider modifications that solve the weights transport problem.

2 Node perturbation with scalar feedback

Add a perturbation to all weights $\theta' = \theta + \sigma\xi$ with $\xi \sim \mathcal{N}(0, 1_d)$, small σ and d the dimension of θ.Then update according to a scalar error feedback $\mathcal{L}(\theta') - \mathcal{L}(\theta)$

$$\Delta\theta = -\eta \frac{\mathcal{L}(\theta') - \mathcal{L}(\theta)}{\sigma}\xi$$

This estimates the gradient $g = \nabla_\theta \mathcal{L}$ as

$$\hat{g} = \frac{\mathcal{L}(\theta + \sigma\xi) - \mathcal{L}(\theta)}{\sigma}\xi = (g \cdot \xi)\,\xi + \mathcal{O}(\sigma)$$

In mean, we get

$$\mathbb{E}[\hat{g}] = g + \mathcal{O}(\sigma).$$

But the larger d, the worse the variance

$$\mathbb{E}[|\hat{g}|^2] = \sum_{i,j,k} g_i g_j \underbrace{\mathbb{E}[\xi_i \xi_j \xi_k^2]}_{\delta_{ij}+2\delta_{ik}\delta_{jk}} + \mathcal{O}(\sigma) = (d+2)|g|^2 + \mathcal{O}(\sigma)$$

$$\mathbb{E}[|\hat{g} - g|^2] = \mathbb{E}[|\hat{g}|^2] - |\mathbb{E}[g]|^2 = (d+1)|g|^2 + \mathcal{O}(\sigma).$$

So the variance scales with d and make the updates more noisy the more parameters there are in the network.

3 (Direct) feedback alignment

Direct feedback alignment was introduced by (Lillicrap-Cownden-Tweed-Akerman, 2014). Instead of $\delta^l = (W^l)^T \delta^{l+1} \odot \phi'(a^l)$ we use

$$\begin{aligned} \delta^l &= B^l \delta^{l+1} \odot \phi'(a^l) \\ \delta^L &= y - a^L \\ \Delta W^l &= \eta\, \delta^{l+1} \phi(a^l)^T \end{aligned}$$

where B^l with $l = 1, \cdots, L-1$ are fixed random weights. Why this works?

Toy example. Produce data points (x, y) according to the rule ("teacher") $y = \beta x$ where $\beta \in \mathbb{R}$ and $x \sim \mathcal{N}(0, 1)$. Learning this data means learning β. Use a 2-layer linear network with weights $\theta = (W^1, W^2)$ denoted as $W^0 = u \in \mathbb{R}^N$and $W^1 = v^T$ and output

$$\hat{y} = v^T u x.$$

Loss is

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{2}\mathbb{E}[(v^T u x - \beta x)^2] \\ &= \frac{1}{2}(v^T u - \beta)^2 =: \frac{1}{2} e^2 \end{aligned}$$

Gradient decent $\Delta\theta = -\eta \nabla_\theta \mathcal{L}$ for small learning rate η becomes $\dot{\theta} = -\nabla_\theta \mathcal{L}$ (gradient flow)

$$\dot{u} = -e(t)v(t) \qquad\qquad \dot{v} = -e(t)u(t).$$

Replace by feedback alignment (FA) with random vector $B^1 = b^T$

$$\dot{u} = -e(t)b \qquad\qquad \dot{v} = -e(t)u(t).$$

Start with $u(0) = 0$ and $v(0) = v_0$. Solve equations for u

$$u(t) = -b \int_0^t e(\tau) d\tau$$

and substitute into $\dot{v}$

$$\begin{aligned} \dot{v} &= -be(t) \int_0^t e(\tau) d\tau = -\frac{1}{2} b \partial_t \left(\int_0^t e(\tau) d\tau \right)^2 \\ v(t) &= v(0) - \frac{1}{2} b \left(\int_0^t e(\tau) d\tau \right)^2 \end{aligned}$$

Hence v aligns to the random feedback direction b. If we started from this configuration, then gradient decent updates ($\dot{u} = e(t)v(t)$) point into the same direction as the FA updates ($\dot{u} = e(t)b$). Hence FA has an initial alignment phase and then approximates gradient descent. Interesting that one can get effective gradient descent, while starting from something else.

Direct feedback alignment. As proposed in (Nøkland, 2026), one can equally well replace the back-propagation of random errors, but directly updating each layer with a random projection of the error.

$$\delta^l = F^l(y - a^L) \odot \phi'(a^l)$$
$$\Delta W^l = \eta\, \delta^{l+1} \phi(a^l)^T$$

Performance. To be completed....

4 Predictive Coding

Predictive coding is an old idea that was formalized by (Rao–Ballard, 1999). The connection to back-propagation was made in (Song-Lukasiewicz-Xu-Bogacz. 2020). Instead of fixed hidden activation $a^l = W^{l-1}\phi(a^{l-1})$, predictive coding introduces dynamic variables $x^0, \cdots, x^L$. They aim to minimize the error

$$e^l = x^l - W^{l-1}\phi(x^{l-1})$$

between adjacent layers, by minimizing an energy function

$$E(x, W) := \sum_{l=1}^{L} \frac{1}{2}\|e^l\|^2$$

via

$$\partial_t x^l = -\frac{\partial E}{\partial x^l} = -e^l + (W^l)^T e^{l+1} \odot \phi'(x^l)$$

or discretized with time step γ

$$x^l_{t+1} = x^l_t + \gamma(-e^l_t + (W^l)^T e^{l+1}_t \odot \phi'(x^l_t)).$$

Weights are updated according to

$$\Delta W^l = -\eta \frac{\partial E}{\partial W^l} = \eta e^{l+1}\phi(x^l)^T$$

This is like a local Hebbian learning rule of pre- times post-synaptic activity, if the error e can be interpreted as a neuron/compartment. However, if at some point of time $\delta^l = -e^l$ and $a^l = x^l$, then this is like back-propagation. To achieve this, one must use the transient regime.

- At $t < 0$, clamp $x^0 = x^{\text{input}}$ and leave $x^1, \cdots, x^L$ relax such that $e^l = 0$ for $l = 1, \cdots, L$. Therefore at time $t = 0$, we achieve $x^l_0 = a^l$.
- At time step $t = 0$, clamp $x^L_0 = y$. Then $e^L_0 = y - W^{L-1}\phi(x^{L-1}_0) = y - a^L$ matches the back-propagation error $e^L_0 = \delta^L$.
- At time step $t = 1$, the layer $L - 1$ is affected (set $\gamma = 1$) according to

$$x^{L-1}_1 = x^{L-1}_0 - \underbrace{e^{L-1}_0}_{0} + (W^{L-1})^T \underbrace{e^L_0}_{y-a^L} \odot \phi'(\underbrace{x^{L-1}_0}_{a^{L-1}})$$

  Therefore, since layer $L - 2$ has not moved yet ($x^{L-2}_1 = a^{L-2}$) we have

$$e^{L-1}_1 = x^{L-1}_1 - \underbrace{W^{L-2}\phi(x^{L-2}_1)}_{=x^{L-1}_0} = -(W^{L-1})^T\delta^L \odot \phi'(a^{L-1}) = \delta^{L-1}$$

  Now update

$$\Delta W^{L-1} = \eta\delta^L\phi(a^{L-1})$$

  while all other $\Delta W^{l<L-1} = 0$. Note that e^{L-1}_1 is computed with the old W^{L-1}.

- Iterating

$$
\begin{aligned}
e_0^L &= \delta^L, \\
e_1^{L-1} &= \delta^{L-1} && \Delta W^{L-1} = \eta e_0^{L-1} \phi(x_0^{L-2}) = \eta \delta^L \phi(a^{L-1}) \\
e_2^{L-2} &= \delta^{L-2} && \Delta W^{L-2} = \eta e_1^{L-1} \phi(x_1^{L-2}) = \eta \delta^{L-1} \phi(a^{L-1})
\end{aligned}
$$

Hence the three assumptions to reproduce BP with PC are:

- Start with clamped input such that all errors are zero
- Update the weights of each layer exactly when the error wave first reaches that layer
- Set $\gamma = 1$

Otherwise, updating weights not instantly, but only once equilibrium $\dot{x}^l = 0$ has been reached

$$
\begin{aligned}
\Delta W^l &= \eta e_\star^{l+1} \phi(x_\star^l)^T \\
e_\star^l &= (W^l)^T e_\star^{l+1} \odot \phi'(x_\star^l)
\end{aligned}
$$

This looks like BP, but the difference is that $x_\star^l \neq a^l$. This is called "prospective configuration" in (Song-Millidge-Salvatori-Lukasiewicz-Xu-Bogacz, 2024).