

Learning without synaptic change: a mechanism for sensorimotor control

KRISTEN FORTNEY
University of Toronto

What are the mechanisms of learning in the brain?

Most theories assume we learn by adjusting our synaptic weights.

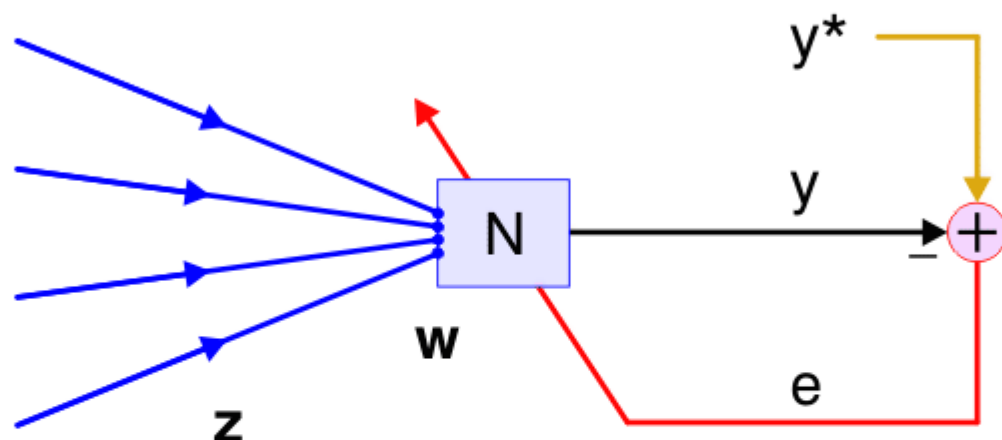
But there is an alternative: the brain could store new information in reverberating loops of activity.

I will show that loops provide a viable mechanism for learning — one which has advantages over synaptic learning, and which can handle challenging tasks.

The background features a complex geometric pattern. It consists of several large, thin, light-purple circles that overlap each other. On the right side, there is a spiral pattern made of many small, short, light-purple line segments that curve inward towards the center.

The basic idea

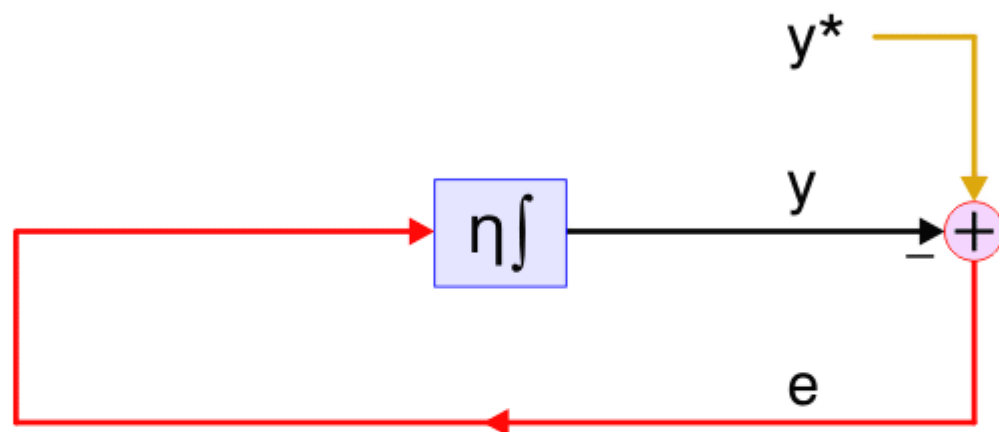
A neuron can learn by adjusting its synaptic weights



This neuron N fires at rate $y = \mathbf{w} \cdot \mathbf{z}$, and its desired firing rate is y^* .
We can drive $y \rightarrow y^*$ with the learning rule

$$\dot{\mathbf{w}} = \eta e \mathbf{z} .$$

It could learn the same task with a reverberating loop
in the form of an integrator



Again, we want to drive some signal $y \rightarrow y^*$. We can do this without synaptic change, by making y the time-integral of e ,

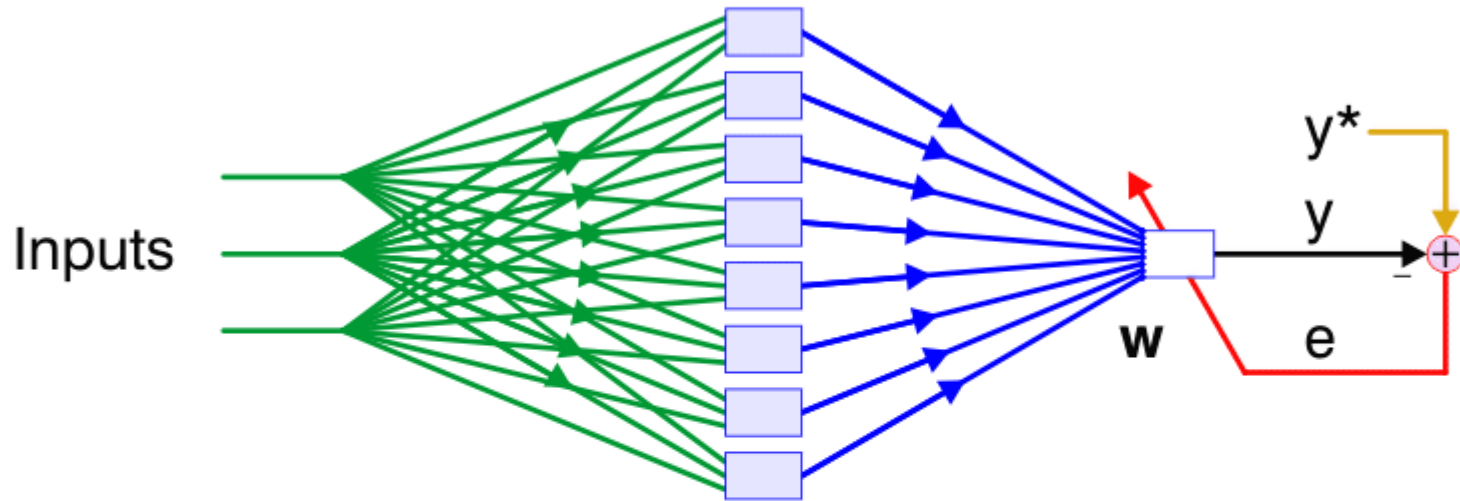
$$\dot{y} = \eta e.$$

Weightless learning is fast

Neural firing rates can be adjusted quickly, on a millisecond timescale.

Synaptic adjustments are likely slower, and even if they are not, training 1 neural signal means adjusting many synapses, which will slow down convergence.

Weightless learning needs fewer neurons



Many theories confine learning to 1 layer of synapses, so they won't need error backpropagation. But then the network's inputs must be sent through a large array of preprocessor neurons, slowing learning.

Weightless learning avoids this preprocessing by steering y directly.

On the other hand, weightless learning is
less permanent

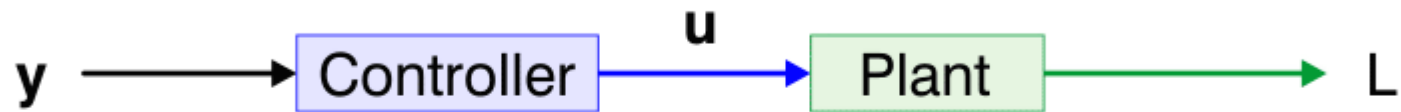
It learns on the fly — it doesn't accumulate knowledge in synapses, but updates neural firing rates based only on their current values and those of the input signals.

In light of these strengths and weaknesses, can weightless learning handle interesting tasks?

The background features a series of overlapping, thin, light-purple circles that create a complex, web-like pattern. On the right side, there is a spiral composed of numerous short, vertical, light-purple bars that curve inward towards the center of the slide.

Sensorimotor learning

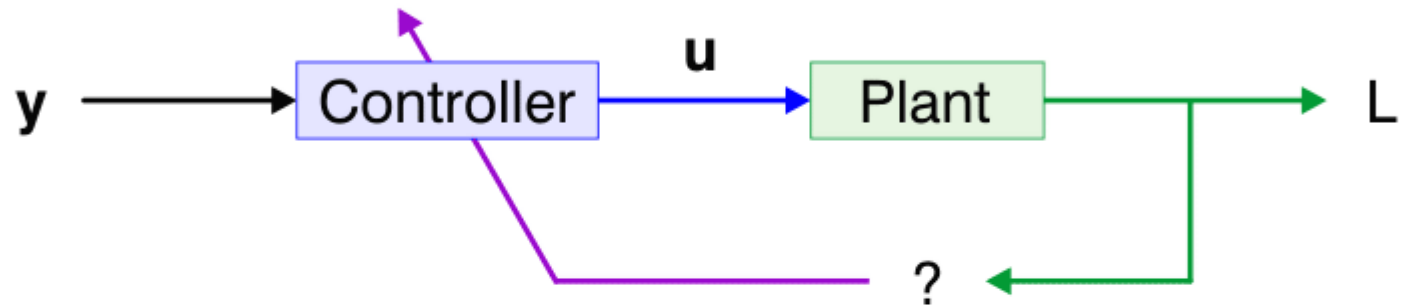
Sensorimotor systems use sensory data to guide motor action



A controller receives input y and sends a command u to its controlled object, or *plant* (e.g. eye, limb).

It tries to minimize some performance index, called the *loss*, L , e.g. in reaching, L might be the distance from hand to target.

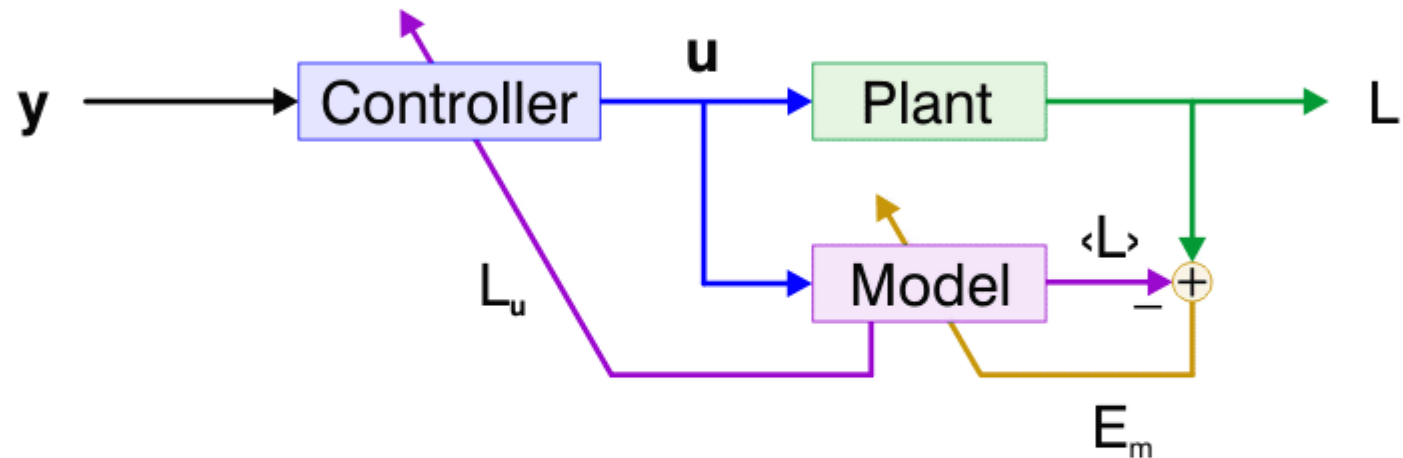
Sensorimotor learning is hard



There is no sensory signal that can tell the controller what it should do; *i.e.* there is no supervisor to guide its learning.

The brain can monitor L , but that doesn't tell it how to adjust u .

How can the brain learn to improve \mathbf{u} ?



To improve \mathbf{u} , the brain needs to deduce the relation between L and \mathbf{u} , *i.e.* it needs to know the partial derivative $\partial L / \partial \mathbf{u}$, or L_u .

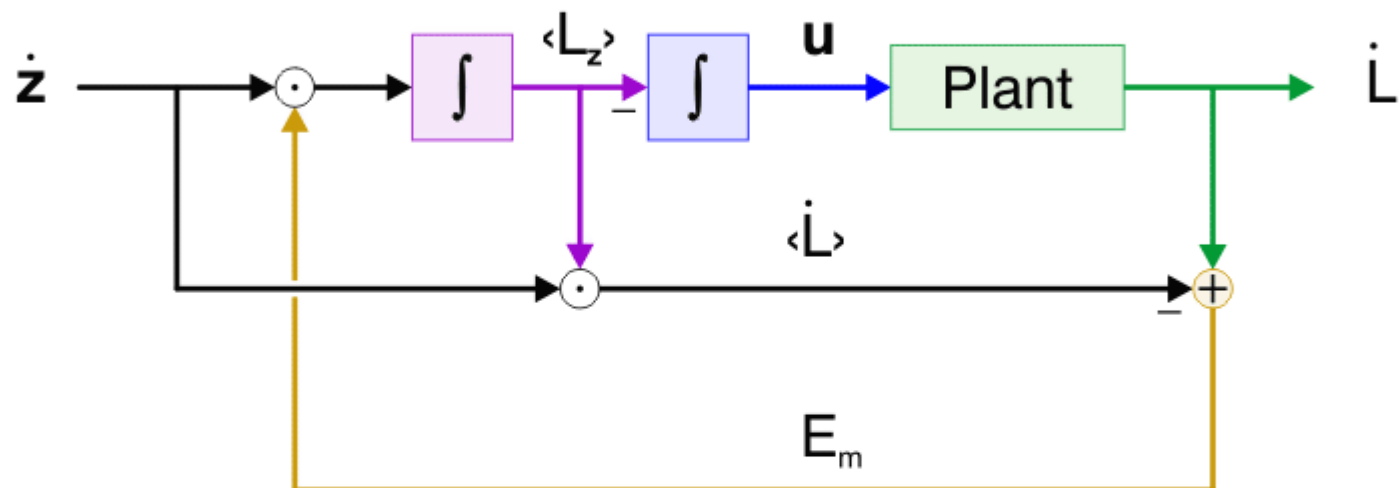
In some theories, a plant model deduces L_u and reports it to a controller, but the process is complex.

A flexible control system has to do 2 jobs

Find L_u : There has to be a process, usually called a plant model, that computes the dependence of L on \mathbf{u} .

Create \mathbf{u} : The controller has to generate a sequence of commands that will minimize L .

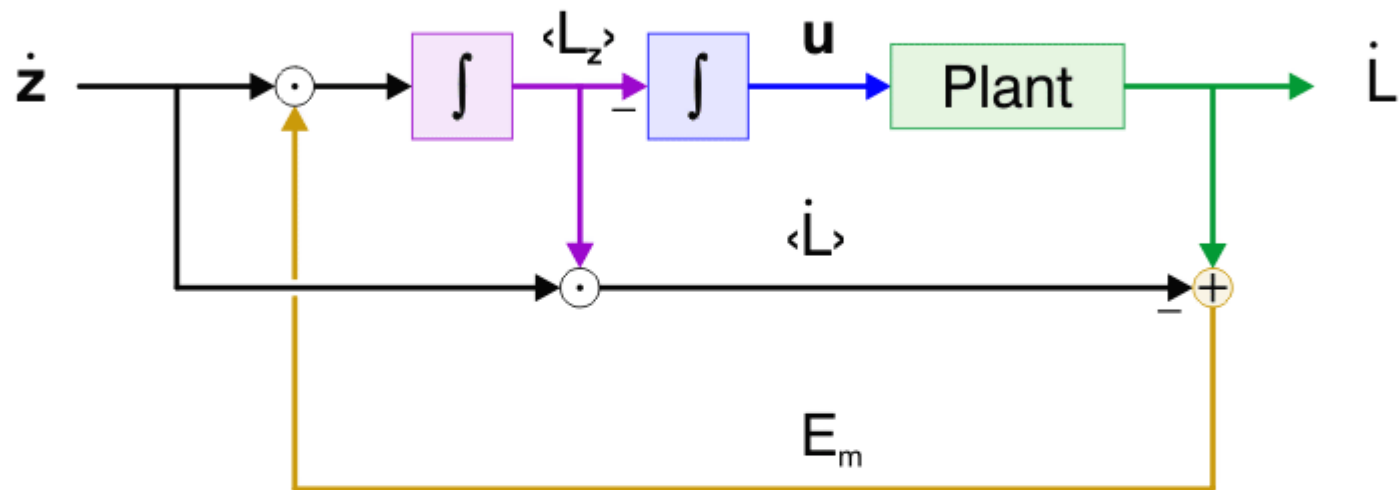
Weightless learning does both jobs by linking 2 integrators in series



The plant is the controlled object, u is the command the brain sends to the plant, L is the loss, dots mark time derivatives, and $\langle \rangle$ means an estimate; E_m is model error.

$z = (u, y)$ where y includes any other variables, besides u , that affect L (e.g. for the VOR, y includes head velocity and eye position) and L_z is dL/dz .

How does this network compute L_u ?

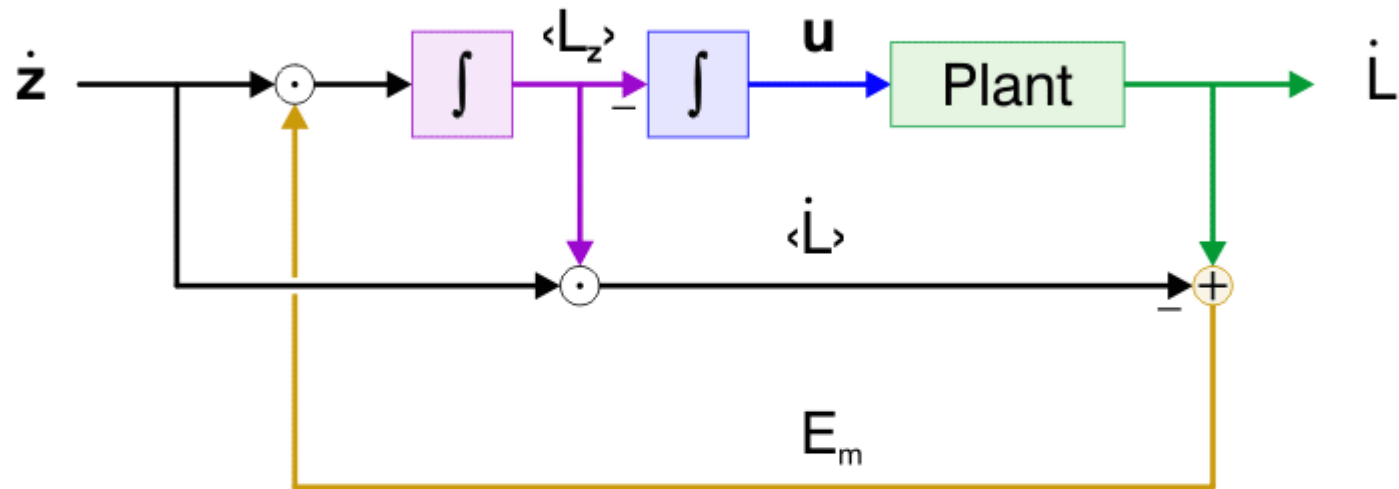


Start by guessing L_z . Use the estimate to generate a command u , and at the same time multiply $\langle L_z \rangle$ by \dot{z} to yield an estimate of \dot{L} . Compare the estimate $\langle \dot{L} \rangle$ with the true \dot{L} , and use the difference, E_m , to improve $\langle L_z \rangle$.

i.e. drive $\langle L_z \rangle$ down the gradient of E_m ²:

$$d\langle L_z \rangle / dt = -\eta_1 E_m \partial E_m / \partial \langle L_z \rangle = \eta_1 E_m \dot{z}.$$

How does this network compute \mathbf{u} ?

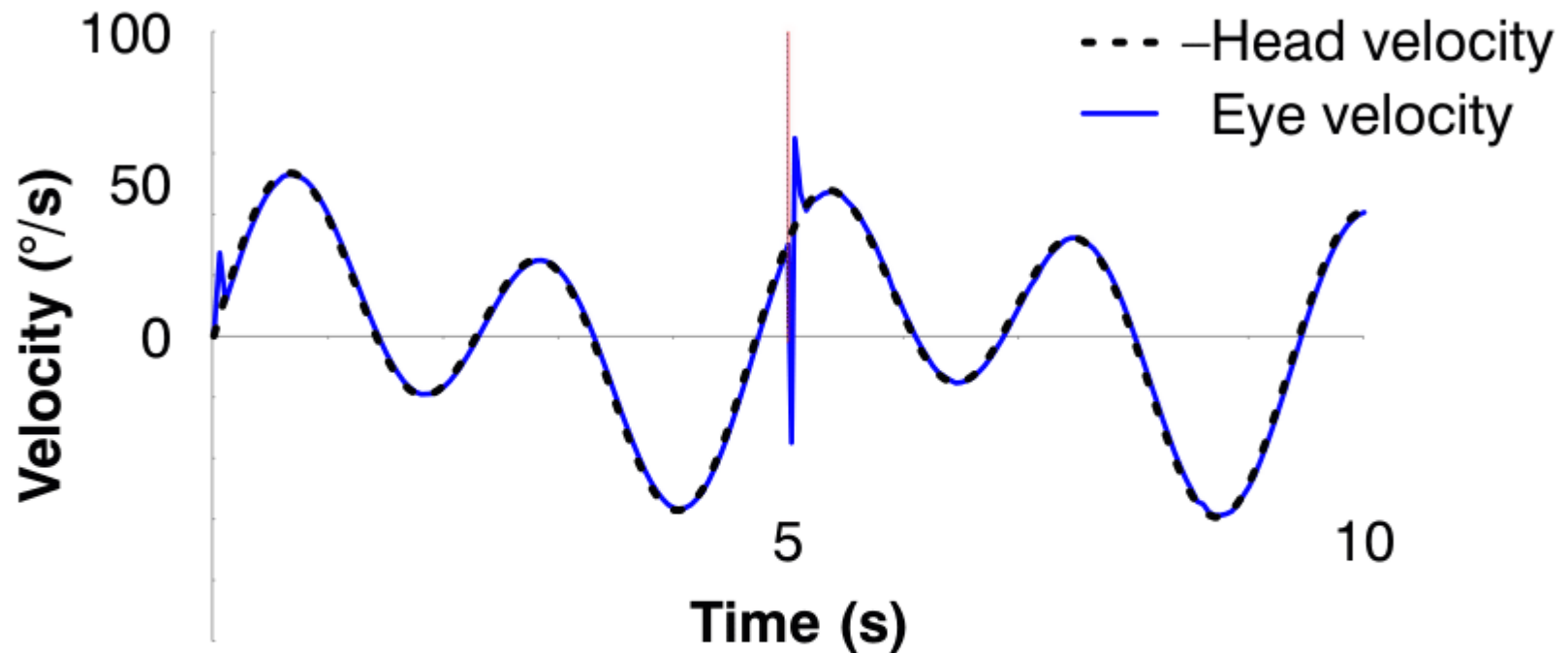


Essentially, it makes $\dot{\mathbf{u}} = -\eta_2 \langle L_u \rangle$ (where $\eta_2 > 0$) so \mathbf{u} is driven to the value that minimizes L .



Simulations

Weightless learning can stabilize the eyeball



A weightless controller quickly learns to counterrotate the eye when the head turns.

At 5 s the plant changes — the eye muscles are transposed — but the circuit regains control.

Weightless learning can control a planar 2-joint arm

The background features a series of thin, light purple circles that overlap to form a complex, web-like pattern. On the right side, there is a spiral composed of numerous short, vertical purple bars of varying lengths, creating a sense of depth and movement.

Implications

Weightlessness explains some puzzling properties of learning

It provides a mechanism for very fast learning, which is harder to explain based on synaptic change.

It explains why motor learning is initially unstable and needs to be consolidated.

If sensory inputs change quickly, weightless learning can't keep up. This could explain why we move slowly when learning and slow down when we start to make errors.

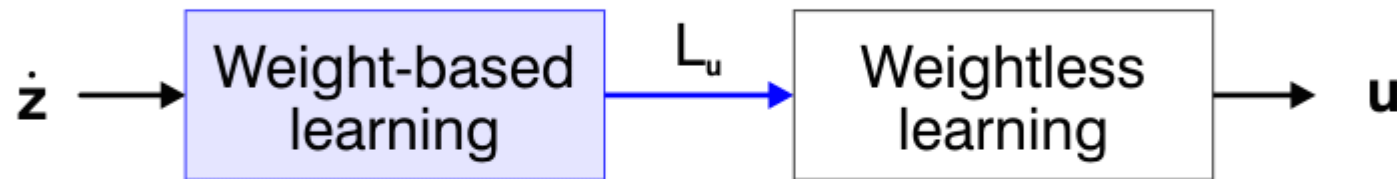
Predictions

Sensorimotor systems should contain cells that store information in sustained firing, and integrators with time constants \geq a few seconds.

Some learning should be possible even when all synaptic change is blocked, and some new learning should be erasable by electrically disrupting neural activity.

Learning should be impaired by altered sensory feedback which accurately represents L but not \dot{L} , and by forced-haste tasks where subjects receive no feedback unless they move quickly.

Weightless and synaptic learning can cooperate

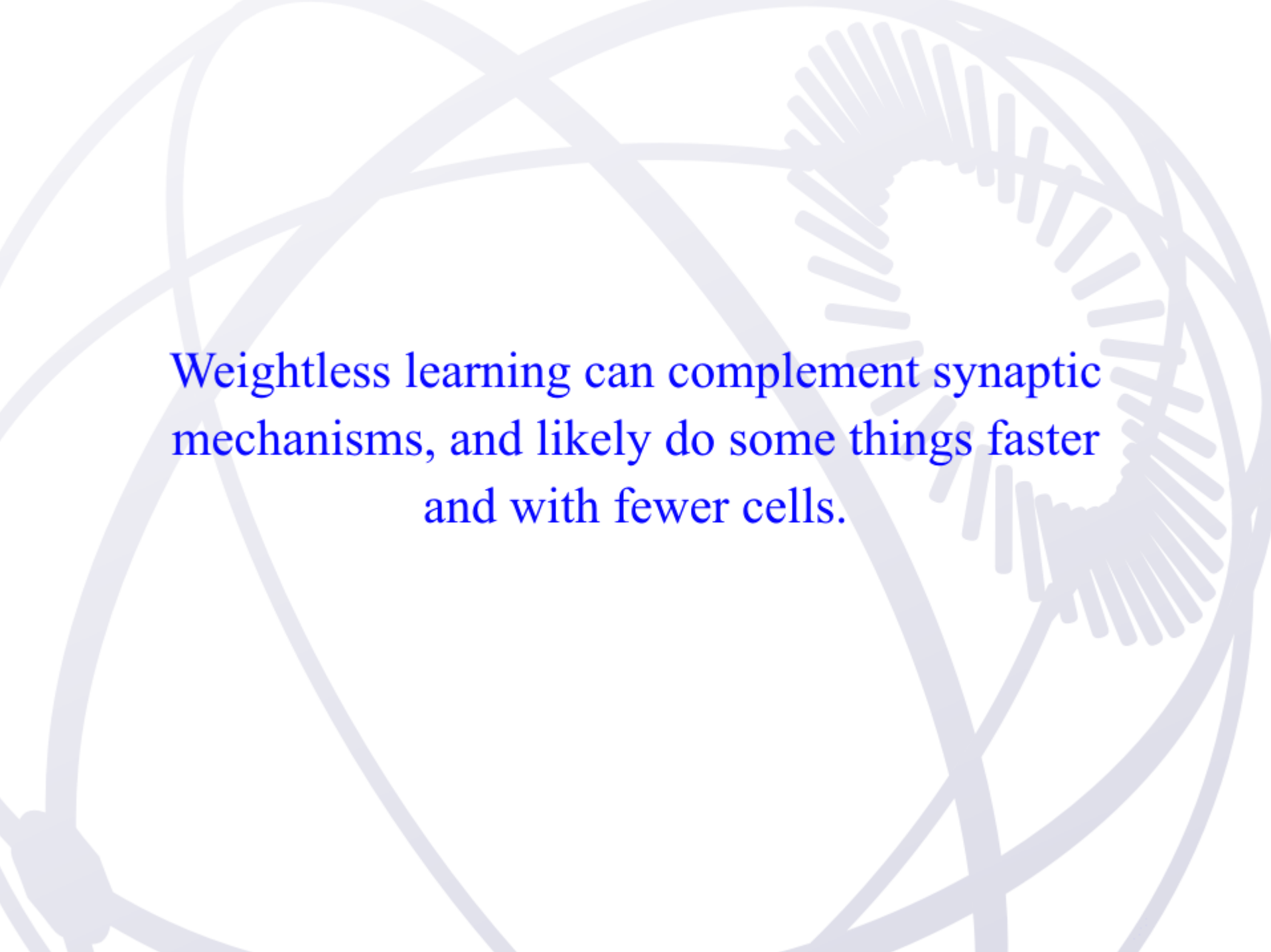


A weight-based plant model could compute the signals needed to drive weightless learning in a controller.

There are other ways the 2 types of learning might interact, *e.g.* fast, weightless learning might later be consolidated in more stable synaptic change.



Conclusion



Weightless learning can complement synaptic mechanisms, and likely do some things faster and with fewer cells.

Acknowledgements

Douglas Tweed

Mohamed Abdelghani

Lak Chinta Venkateswararao

Maryam Fesharaki

Danitza Goche Montes

CIHR

An experimental test for weightless learning: forced haste

When sensory inputs change quickly, weightless learning can't keep up. So if we depend on weightless mechanisms for short-term motor learning, then that learning should be severely impaired when we are forced to move quickly.

To test this prediction, I will have subjects learn to track a fast-moving point, or learn to move a cursor to a stationary target under conditions where they get no feedback about their performance unless they attain the target quickly.

An experimental test for weightless models: recovery from plant reversals

In a sensorimotor task, if the plant model learns weightlessly, it should learn fast enough that, when any component of $\partial L / \partial \mathbf{u}$ changes sign, the controller begins to improve immediately.

But if the model is slower-learning, there should be an initial phase where the controller can't improve because the model's estimate of $\partial L / \partial \mathbf{u}$ has the wrong sign.

We will have subjects learn to track a moving point using a joystick-driven cursor, and then we will change the cursor's dynamics to reverse $\partial L / \partial \mathbf{u}$, and chart the recovery.

How does the network compute \mathbf{u} ?

We want an adjustment $\Delta \mathbf{u}$ that makes $\Delta L = -L$. We assume $L = \frac{1}{2} \mathbf{e}^T \mathbf{e}$, where \mathbf{e} is a linear function of $\mathbf{z} = (\mathbf{u}, \mathbf{y})$.

Then we have $-L = \Delta L = -\frac{1}{2} \Delta \mathbf{e}^T \Delta \mathbf{e} = \frac{1}{2} \mathbf{e}^T \Delta \mathbf{e} = \frac{1}{2} \mathbf{e}^T \mathbf{e}_z \Delta \mathbf{z} = \frac{1}{2} L_z \Delta \mathbf{z}$, or

$$-2L = L_z \Delta \mathbf{z} = L_u \Delta \mathbf{u} + L_y \Delta \mathbf{y}.$$

So we want $\Delta \mathbf{u} : L_u \Delta \mathbf{u} = -2L - L_y \Delta \mathbf{y}$. The smallest such $\Delta \mathbf{u}$ is

$$-(2L + L_y \Delta \mathbf{y}) L_u / (L_u L_u^T).$$