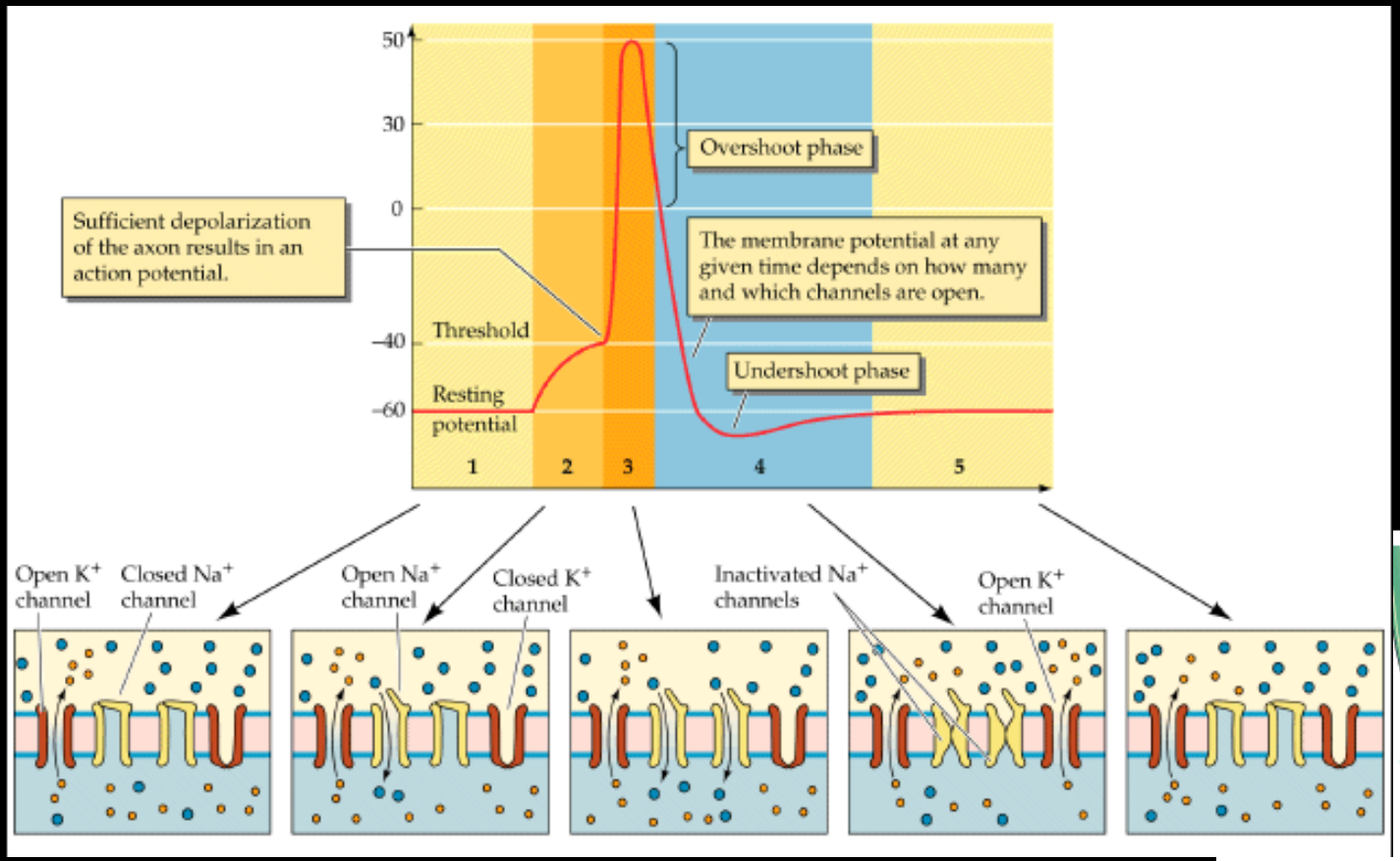
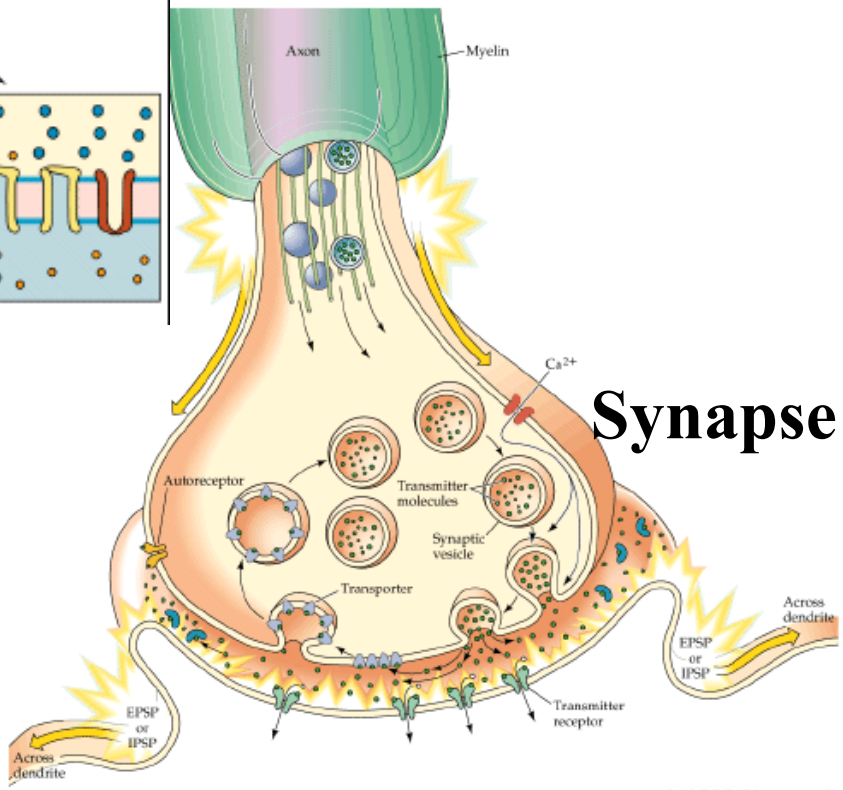
Backpropagation and biological plausibility

Review: How neurons work



"Weights" (W) change as one action potential leads to more or less transmitter release and/or depolarization of postsynaptic cell
LTP = long term potentiation, increased W
LTD = long term depression, decreased W



Synapse

Is Backpropagation biologically plausible?

- No. It requires implausible "transport" of weights, so that the same weights provide both synaptic efficacy and somehow propagate backward an error.

- BUT backpropagation IS still a really great machine learning method!

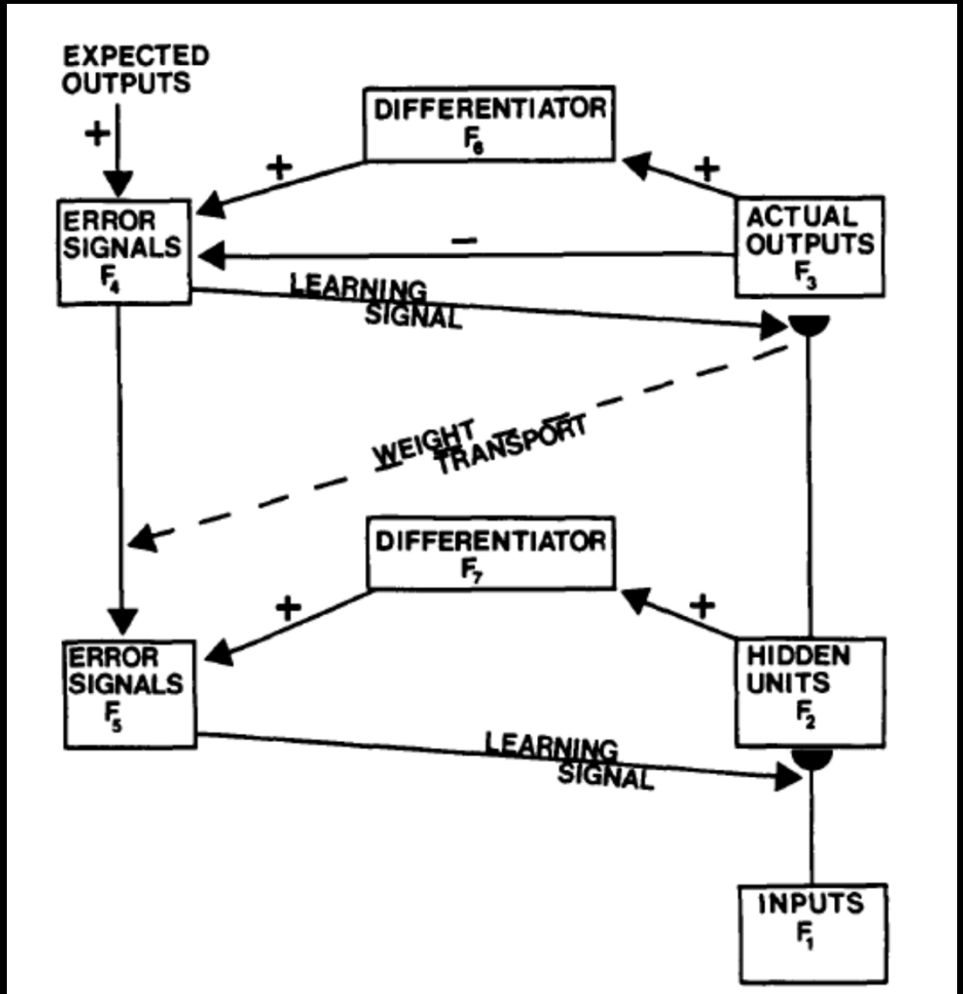


Figure 8. Circuit diagram of the back propagation model: In addition to the processing levels F_1, F_2, F_3, there are also levels F_4, F_5, F_6, and F_7 to carry out the computations which control the learning process. The transport of learned weights from the $F_2 - F_3$ pathways to the $F_4 - F_5$ pathways shows that this algorithm cannot represent a learning process in the brain.

Deep learning – how might the Brain do it??

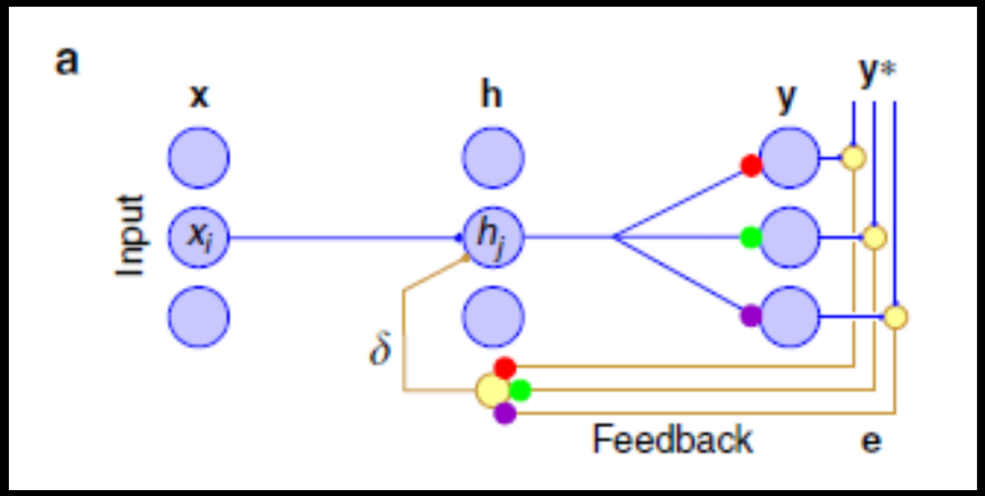
- If the backpropagation algorithm isn't biologically plausible, then HOW might the brain do supervised learning??

Background: Lillicrap et al. 2016

- The big idea: randomly initialize the "backward" weights, then FREEZE them. Basically TURN OFF learning in the backward weights.
- This means the backward weights are distinct from the forward weights, and only the forward weights can learn.
- This seems crazy, but it actually works, and very well at that!
- New algorithm called "feedback alignment"

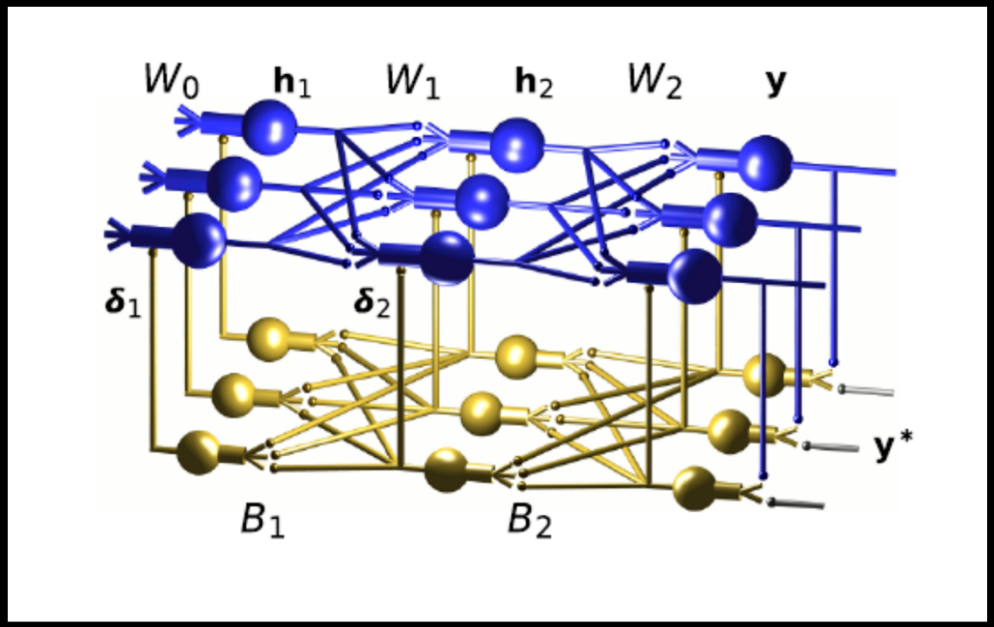
Lillicrap et al. 2016

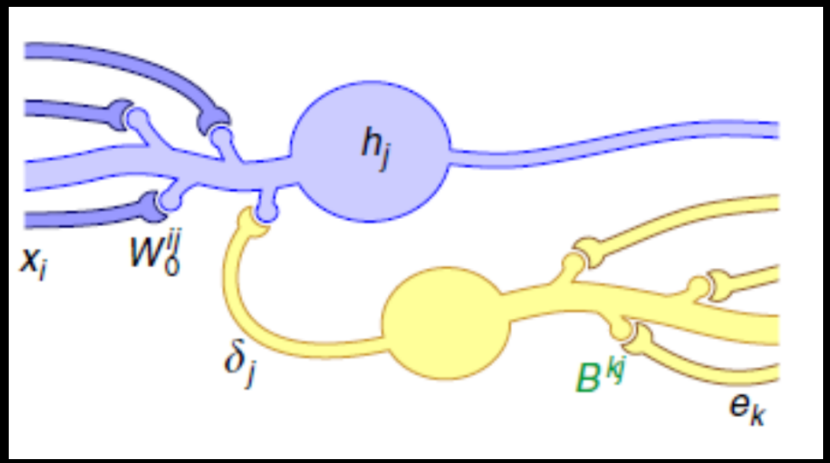
- Backpropagation (of errors) – addresses the credit assignment problem, i.e. how to assign credit/blame for a particular prediction error?
- Basic idea: activate the network forward, the compute the difference between actual and desired output: e=d-y
- With standard backpropagation, proceed to propagate the error signal backward, multiplying by the forward weights. Use the backpropagated error signal to train the forward connections via delta rule.



Lillicrap et al. 2016

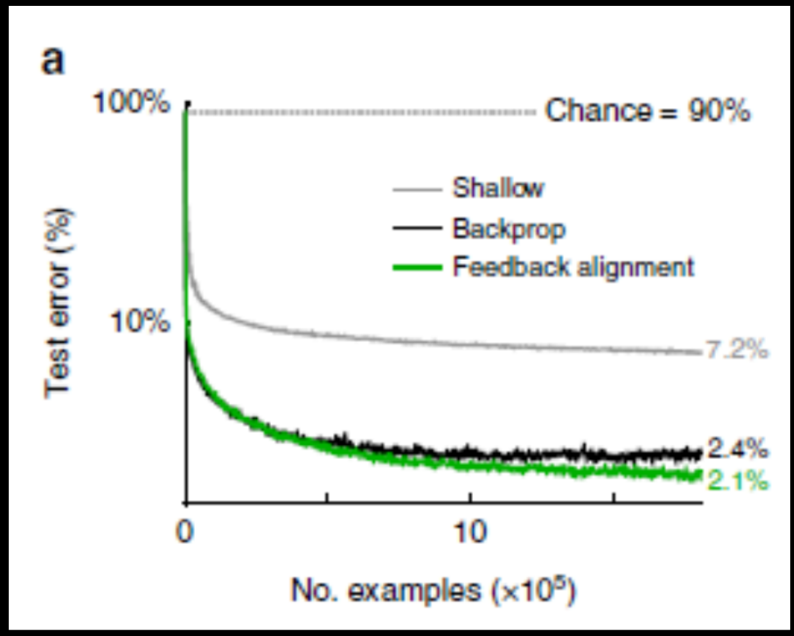
- With Feedback Alignment algorithm, SEPARATE forward and backward weights
- Initialize the backward weights to SMALL, RANDOM VALUES. Then FIX them (i.e. no learning occurs)
- This effectively projects the error signals onto a higher dimensional manifold in the network representation space
- Then compute the error signals by backpropagating errors through the fixed, backward weights. Train forward weights by the delta rule as usual.

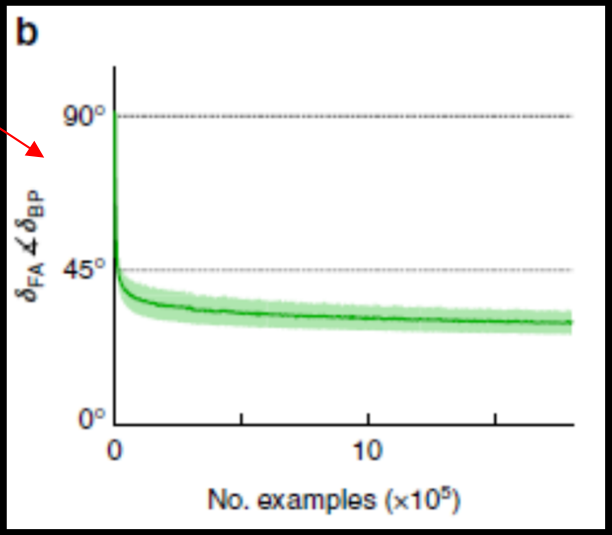


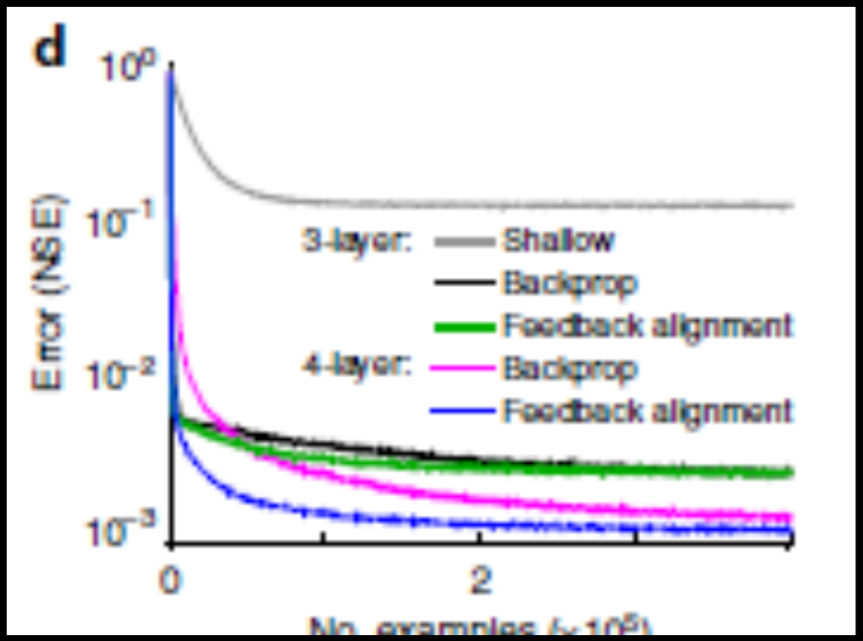


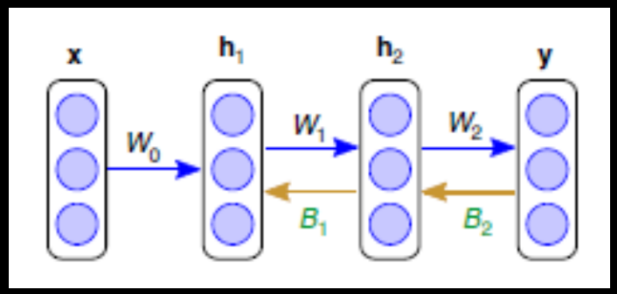
Lillicrap et al. 2016

- Works well on deep network with MNIST digit data
- Works especially well with deeper networks
- The network will converge as long as the error (B*e) lies within 90 degrees of what the backprop error would be (W^T*e)









Lillicrap et al. 2016

- Example: bolt on some spiking neurons in a deep network.
- Present input for 50 time steps each, compute error signals and train network.
- Deep network of 784-1500-1500-1500-1000, 3 hidden layers
- Solves vanishing gradient problem, too!

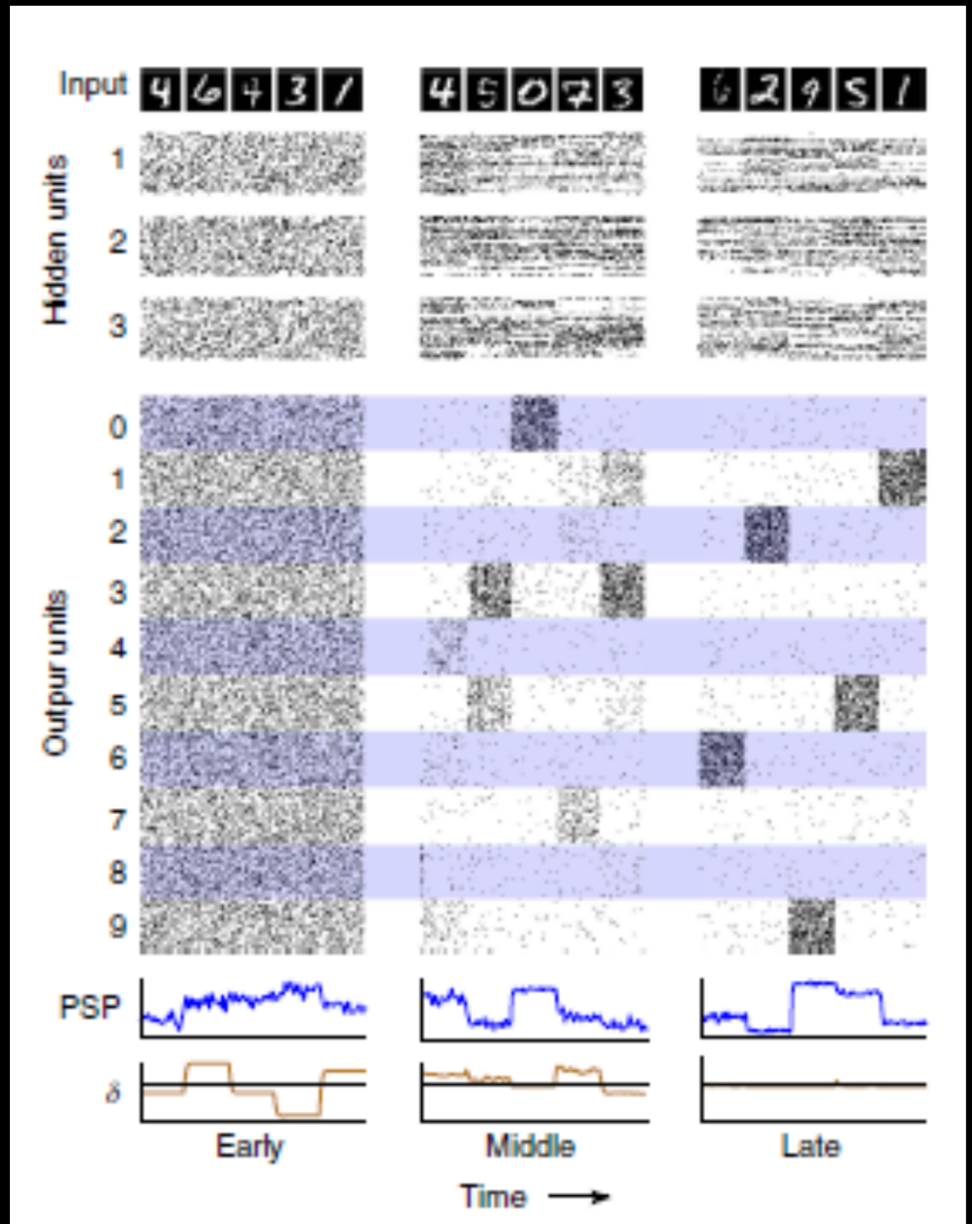
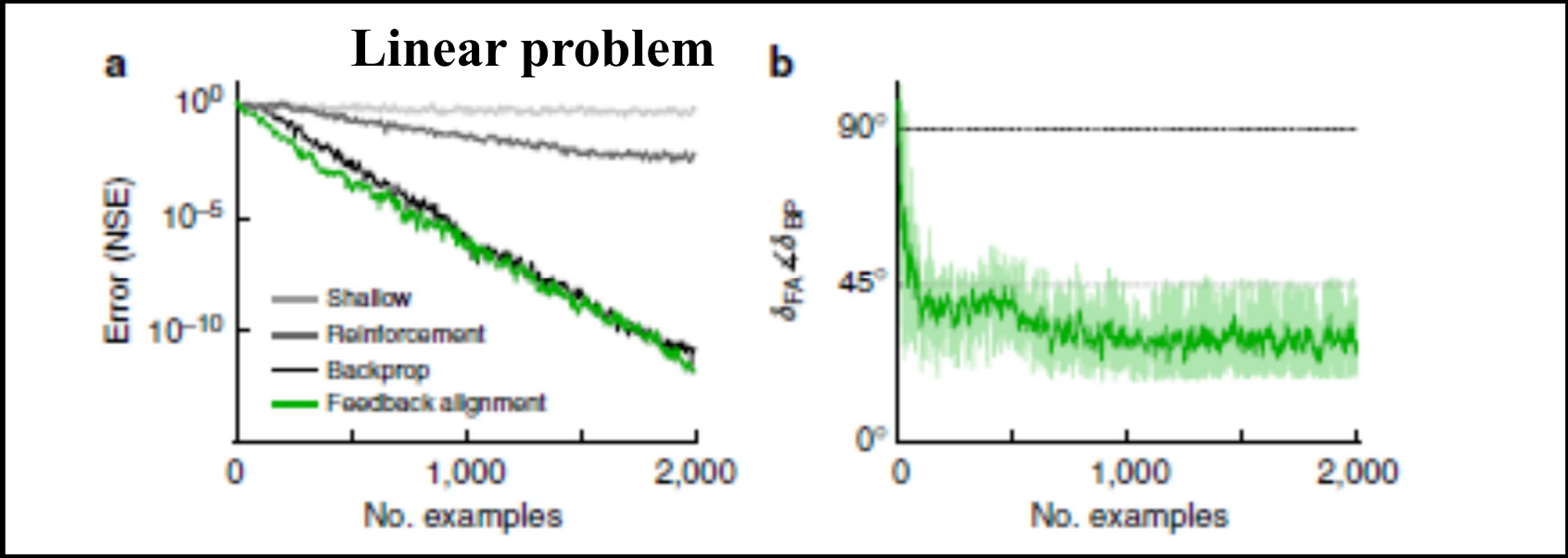


Figure 4 | Feedback alignment operates in larger networks with more complex dynamics. A network (784-1500-1500-1500-1000) of neurons that integrate their activity over time and spike stochastically, learns to recognize handwritten digits. The inputs (images along top)

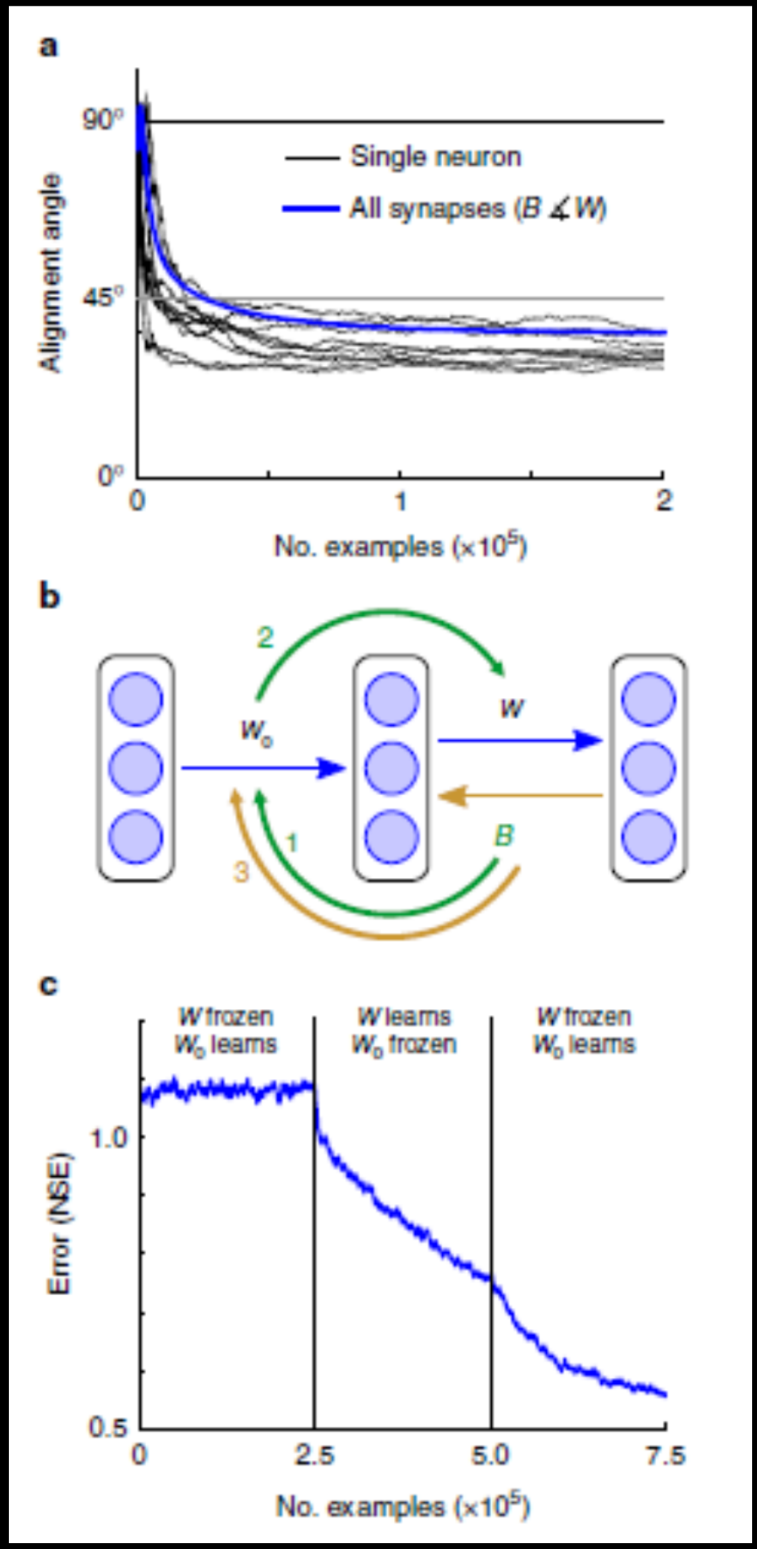
Lillicrap et al. 2016

- Results show performance better than RL and comparable to backprop.
- Key point is the “angle” between the forward and backward weights. They start off independent (90 degrees to each other), but eventually converge towards zero. (They never reach zero because the feedback weights are fixed and will not generally provide an exact match to the forward weight solution)



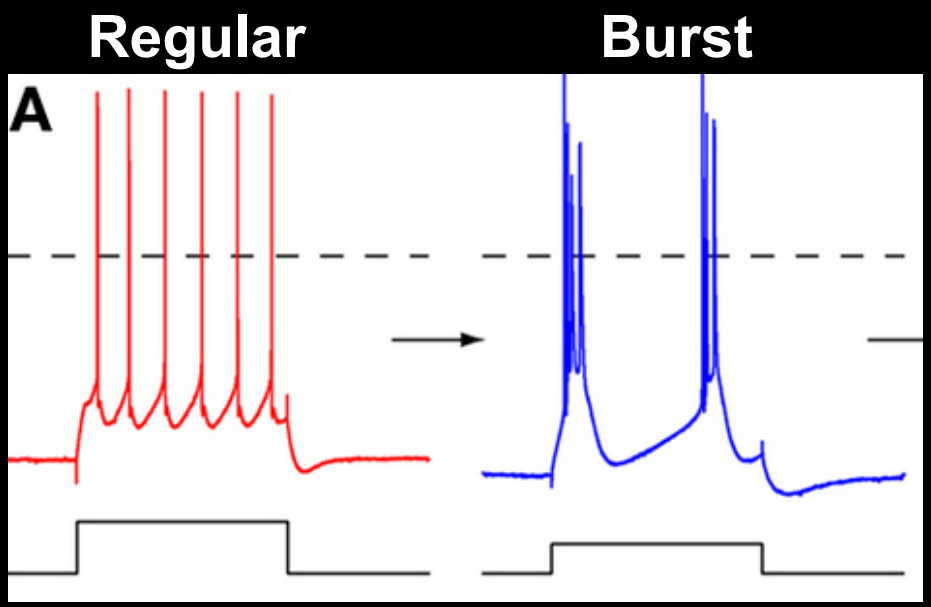
Lillicrap et al. 2016

- How does this work? Basically the fixed backward weights provide a large, expanded and distributed basis of feedback error signals throughout the network. Each forward synapse at each layer is adjusted to minimize the error across the whole error basis.
- Error information is still propagated backwards through the network and adjusts forward synapses at all layers



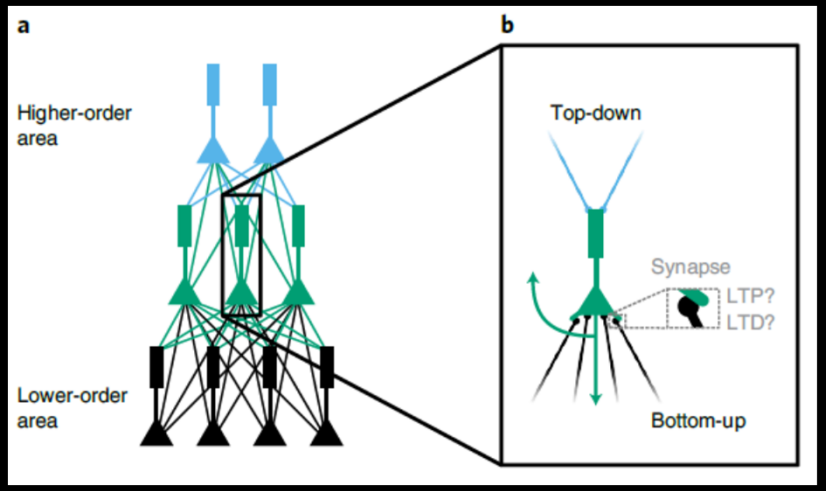
Payeur et al. 2021

- How might Backprop be implemented in spiking neurons?
- Intuition: Error signals activate apical dendrites in lower level cells, which increases the “burstiness” of these cells
- The error signal increases bursting **but not overall firing rate**, thus multiplexing error signals and forward activity
- Increased bursting (relative to the average bursting rate) paired with increased presynaptic activity implements the delta rule to increase synaptic weights



Payeur et al. 2021

- How might Backprop be implemented in spiking neurons?



$$\frac{dw_{ij}}{dt} = \eta[B_i(t) - \overline{P}_i(t)E_i(t)]\widetilde{E}_j(t). \quad (1)$$

W = weight
B = "bursts", the error signal from higher level neurons
~E = presynaptic eligibility trace
E = postsynaptic activity
P= moving average of bursts

- Bursts from higher level neurons provide the error signal
- Bursts activate apical dendrites of lower pyramidal cells?

Backward weights?

- Issue: What are the weights from the higher to lower level neurons?
- One possibility is fixed, random weights as in feedback alignment
- Another is Akrout et al. (2020) arXiv:1904.05391v5 – the weight mirror algorithm. Feedback alignment doesn't scale well, so train random B weights to converge to transpose of forward weights

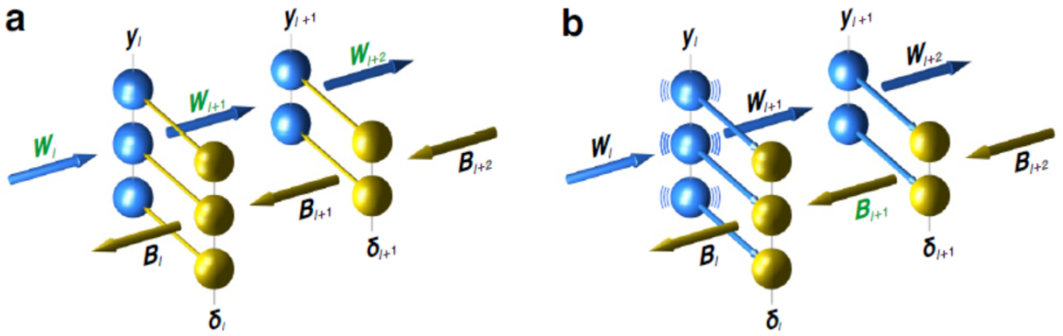
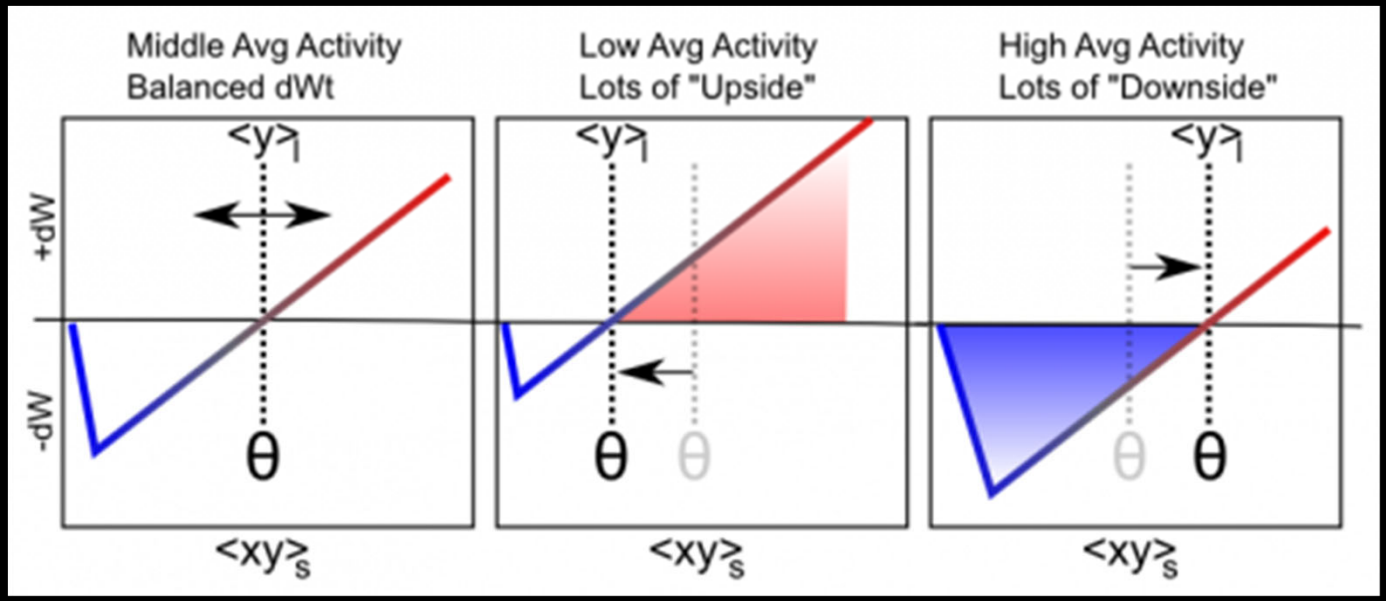


Figure 1: Network modes for weight mirroring. Both panels show the same two-layer section of a network. In both modes, the three neurons in layer l of the forward path (▬) send their output signal y_l through the weight array W_{l+1} (and other processing shown in equation (1)) to yield the next-layer signal y_{l+1}. And in the feedback path (▬), the two neurons in layer $l+1$ send their signal δ_{l+1} through weight array B_{l+1} to yield δ_l, as in (3). The figure omits the biases b, nonlinearities ϕ, and, in the top panel, the projections that convey y_l to the δ_l cells, allowing them to compute the factor $\phi'(\mathrm{y}_l)$ in equation (3). **a)** In *engaged mode*, cross-projections (▬) convey the feedback signals δ to the forward-path cells, so they can adjust the forward weights W using learning rule (4). **b)** In *mirror mode*, one layer of forward cells, say layer l, fires noisily. Its signal y_l still passes through W_{l+1} to yield y_{l+1}, but now the blue cross-projections (▬) control firing in the feedback path, so $\delta_l = \mathrm{y}_l$ and $\delta_{l+1} = \mathrm{y}_{l+1}$, and the δ_l neurons adjust the feedback weights B_{l+1} using learning rule (7). We call the circuit y_l, y_{l+1}, δ_{l+1}, δ_l a *weight mirror* because it makes the weight array B_{l+1} resemble W_{l+1}^T.

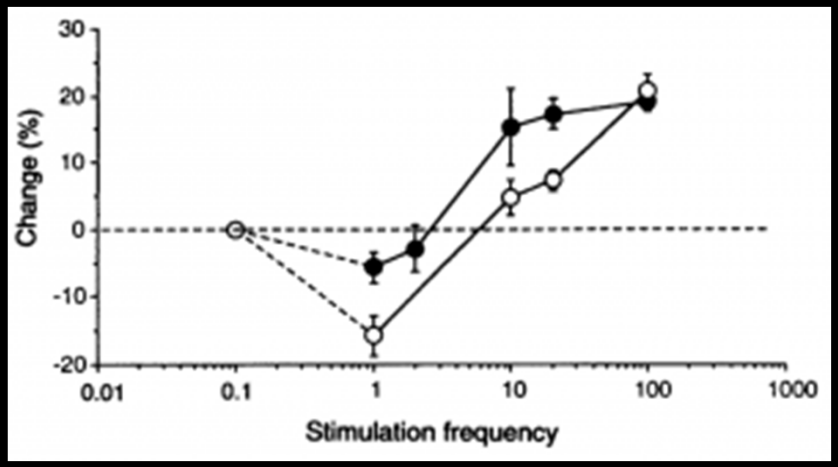
BCM model

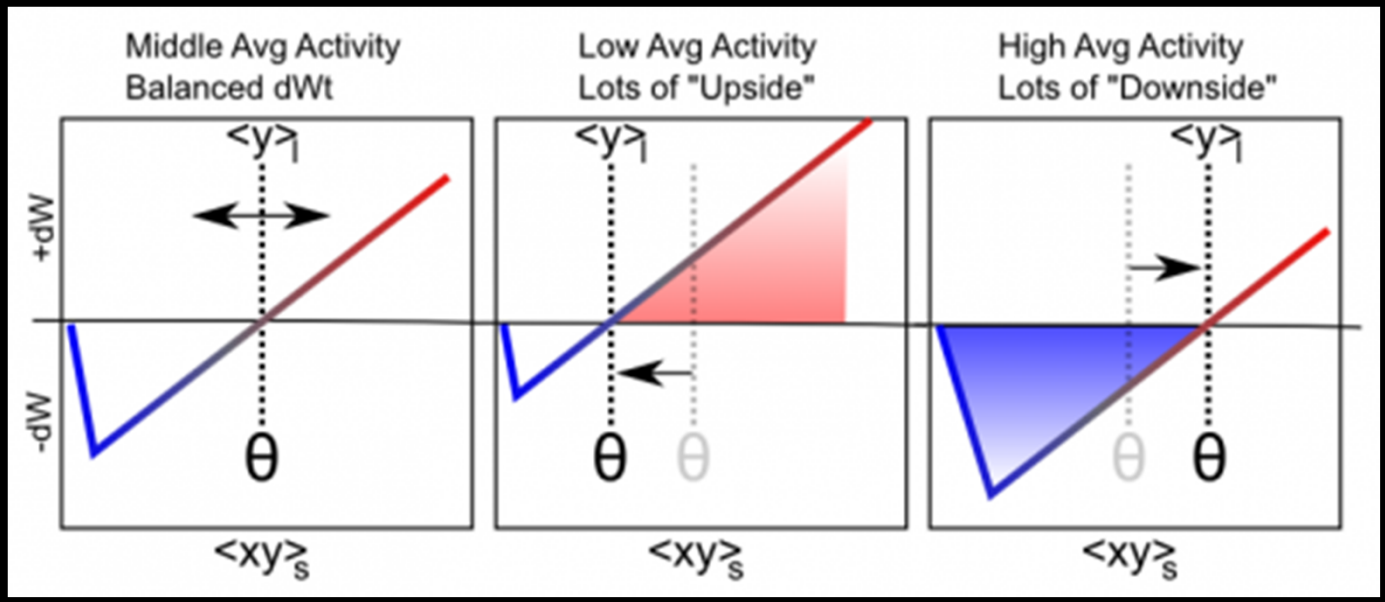
- Bienenstock, Cooper, & Munro (1982)
- Key point is that the threshold $\theta = \langle y^2 \rangle$. That means if the postsynaptic cell is very active, the threshold θ will be higher, which leads to more weight reduction. This helps normalize the input to a post-synaptic cell



BCM model

- Is there neural evidence of such a system?
- Yes, at least for weight changes vs. activation





https://compcogneuro.org/

Spike timing dependent plasticity

- Hebb didn't have the whole story
- If the presynaptic neuron fires an action potential BEFORE the postsynaptic cell fires, then the synaptic weight increase.
- If the presynaptic cell fires AFTER, then the synaptic weight DECREASES.
- This allows inference of causation: "if A fires before B, then A causes B"