$$\dot{v}_i = \phi\left(\sum_j w_{ij} v_j\right) - v_i$$

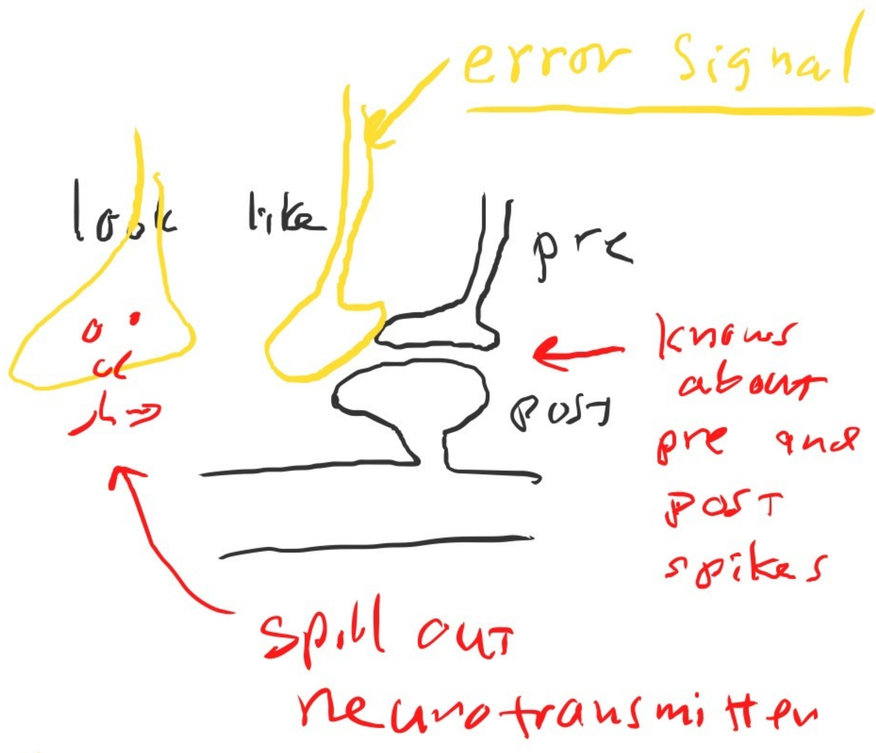
biologically plausible learning rules look like

$$\Delta w_{ij} = f(v_i, v_j)$$

post

pre

is this a powerful enough learning rule to do anything useful?



more correct:

$y = \phi(\underset{\sim}{w} \cdot \underline{x})$

$y = \underset{\sim}{w} \cdot \underline{x}$

Oja's rule

$$\Delta \underset{\sim}{w} = \eta\, y(\underline{x} - y\,\underset{\sim}{w})$$

$$= \eta\, \underset{\sim}{w} \cdot \underline{x}\,(\underline{x} - \underset{\sim}{w} \cdot \underline{x}\,\underset{\sim}{w}) = \eta\, \underset{\sim}{w} \cdot \underline{x}\,\underline{x} - \eta\, \underset{\sim}{w}\,\underset{\sim}{w} \cdot \underline{x}\,\underline{x} \cdot \underset{\sim}{w}$$

$$\tau \dot{w} = \eta \left\{ \underset{\sim}{w} \cdot \langle \underline{x}\,\underline{x} \rangle - \underset{\sim}{w}\,\underset{\sim}{w} \cdot \langle \underline{x}\,\underline{x} \rangle \cdot \underset{\sim}{w} \right\}$$

$$\tau \dot{\underset{\sim}{w}} = \eta \left[\underline{\underline{C}} \cdot \underline{w} - \underline{w} \cdot \underline{\underline{C}} \cdot \underline{w} \, \underline{w} \right]$$

$$\underline{w} = \sum_k a_k \underline{v}_k \qquad \underline{\underline{C}} \cdot \underline{v}_k = \lambda_k \underline{v}_k \qquad \underline{v}_k \cdot \underline{v}_l = \delta_{kl}$$

$$\tau \sum_k \dot{a}_k \underset{\sim}{v}_k = \eta \left[\sum_k a_k \lambda_k \underline{v}_k - \left[\sum_l a_l^2 \lambda_l \right] \sum_k a_k \underline{v}_k \right]$$

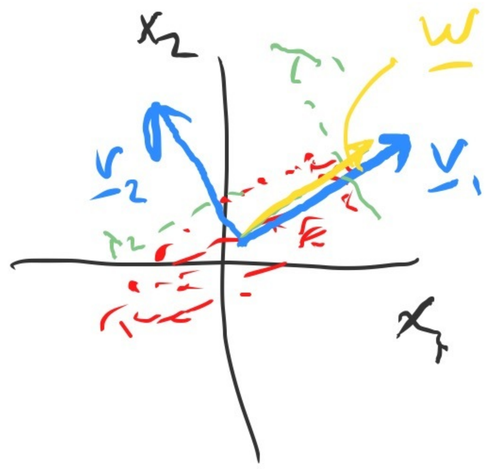
$$\tau \dot{a}_k = \eta \left[\lambda_k - \sum_l a_l^2 \lambda_l \right] a_k$$

assume all eigenvalues are different

$$\lambda_k = \sum_l a_l^2 \lambda_l \qquad \forall k$$

$$a_k = 1 \qquad a_{l \neq k} = 0$$

typically, largest eigenvalue wins



$$\sum_{lm} a_l \underline{v}_l \cdot \underline{\underline{C}} \cdot a_m \underline{v}_m$$

$$= \sum_{lm} a_l a_m \underline{v}_l \cdot \underline{v}_m \lambda_m$$

$$= \sum_l a_l^2 \lambda_l$$

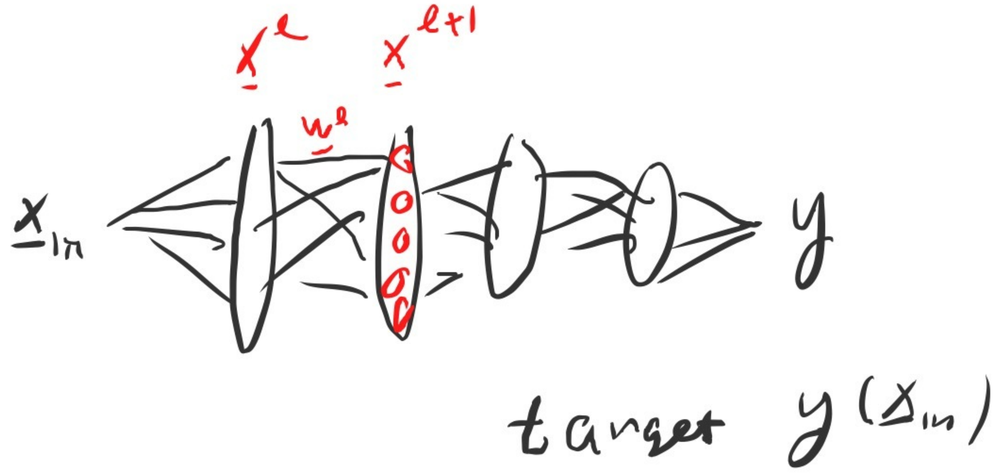
$$\underset{\sim}{w} = \underline{v}_k$$

learning in real networks

1. vanilla deep networks

$$\underline{x}^{l+1} = \phi(\underline{h}^l)$$

$$\underline{h}^l = \underline{\underline{w}}^l \cdot \underline{x}^l$$



$$\mathcal{L}(\underline{x}_{in}, y(\underline{x}_{in}))$$

$$\Delta \underline{\underline{w}}^l = -\gamma \frac{\partial \mathcal{L}}{\partial \underline{\underline{w}}^l}$$

$$\frac{\partial \mathcal{L}}{\partial \underline{\underline{w}}^l} = \frac{\partial \mathcal{L}}{\partial \underline{h}^l} \cdot \frac{\partial \underline{h}^l}{\partial \underline{\underline{w}}^l} = \frac{\partial \mathcal{L}}{\partial \underline{h}^l} \underline{x}^l$$

$$\left\| \frac{\partial \mathcal{L}}{\partial w_{ij}^l} = \frac{\partial \mathcal{L}}{\partial h_i^l} x_j^l \right.$$

$$= \underset{\sim}{\delta}^l \underline{x}^l$$

$$\frac{\partial L}{\partial \underline{h}^{l}} = \frac{\partial L}{\partial \underline{h}^{l+1}} \cdot \frac{\partial \underline{h}^{l+1}}{\partial \underline{h}^{l}}$$

$$\underline{h}^{l+1} = \underline{\underline{W}}^{l+1} \cdot \underline{x}^{l+1}$$

$$= \underline{\underline{W}}^{l+1} \cdot \phi(\underline{h}^{l})$$

$$\frac{\partial \underline{h}^{l+1}}{\partial \underline{h}^{l}} = \underline{\underline{W}}^{l+1} \odot \phi'(\underline{h}^{l})$$

$$\underset{\sim}{\delta}^{l} = \frac{\partial L}{\partial \underline{h}^{l}}$$

$$\underset{\sim}{\delta}^{l} = \underline{\delta}^{l+1} \cdot \underline{\underline{W}}^{l+1} \odot \phi'(\underline{h}^{l})$$

$$\delta_i^{l} = \sum_j \delta_j^{l+1} W_{ji}^{l+1} \phi'(h_i^{l})$$

$$\underset{\sim}{\delta}^{l} = \underset{=}{\delta}^{l+1} \cdot \underset{=}{w}^{l+1} \circ \phi'(\underset{-}{h}^{l})$$

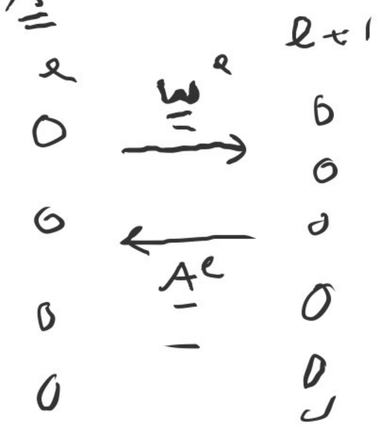
outgoing weights

① Tim Lillicrap: replace $\underset{=}{w}^{l+1}$ with a random matrix $\underset{=}{A}$

(feedback alignment)

- works well on simple problems
- doesn't work on hard problems

② upgrade: train $\underset{=}{A}$



training:

$$\Delta \underset{=}{A}^{l} = \eta \langle \underset{-}{x}^{l} \underset{-}{x}^{l+1} \rangle$$

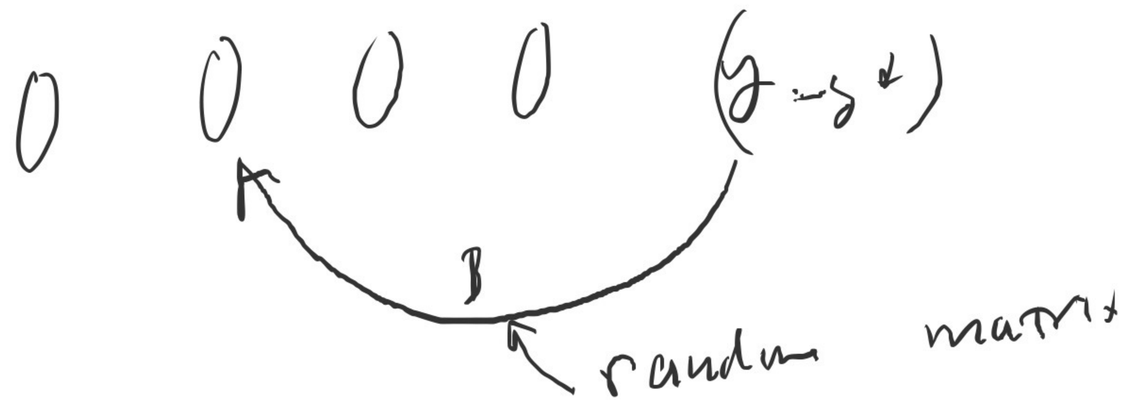
during training: add white noise to $\underset{-}{x}^{l}$

$$\Delta \underset{=}{A}^{l} = \eta \langle \underset{-}{x}^{l} \underset{-}{x}^{l} \cdot \underset{=}{w}^{l\,T} \rangle - \Sigma \underset{=}{A}^{l}$$

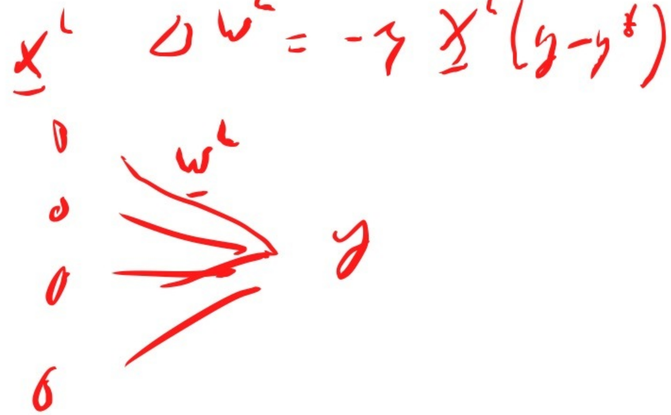
$$= \eta \langle \underset{-}{x}^{l} \underset{-}{x}^{l} \rangle \cdot \underset{=}{w}^{l\,T} - \Sigma \underset{=}{A}^{l}$$

$$= \eta \underset{=}{w}^{l\,T} - \Sigma \underset{=}{A}^{l}$$

Direct feedback alignment

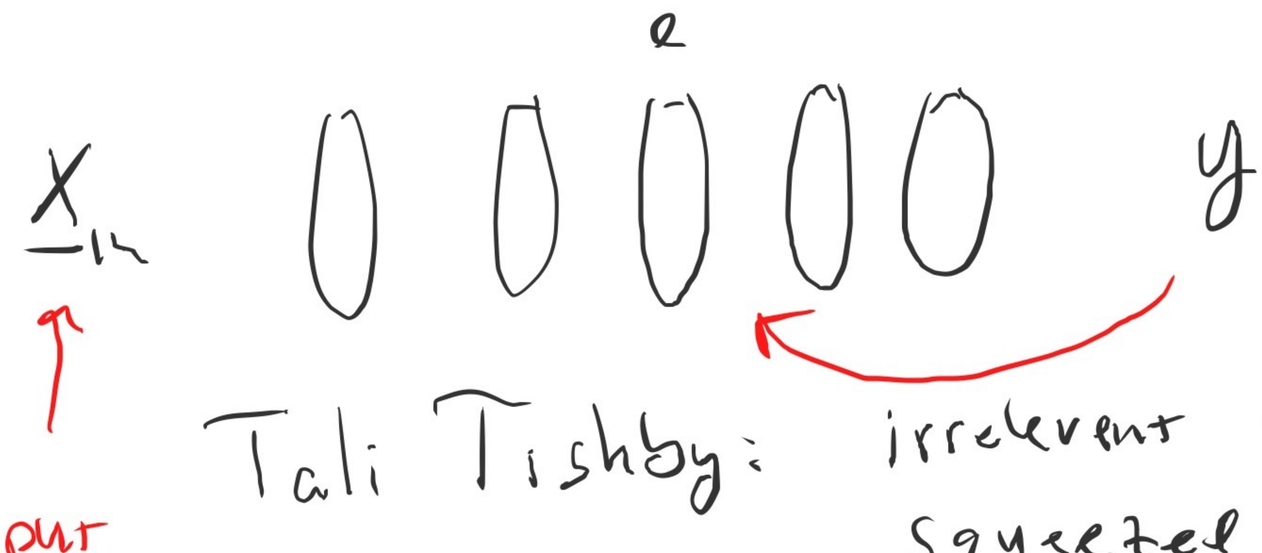


$$\Delta w^l = B^l (y - y^*)$$



Works pretty well on problems that aren't too hard

Bottleneck approach

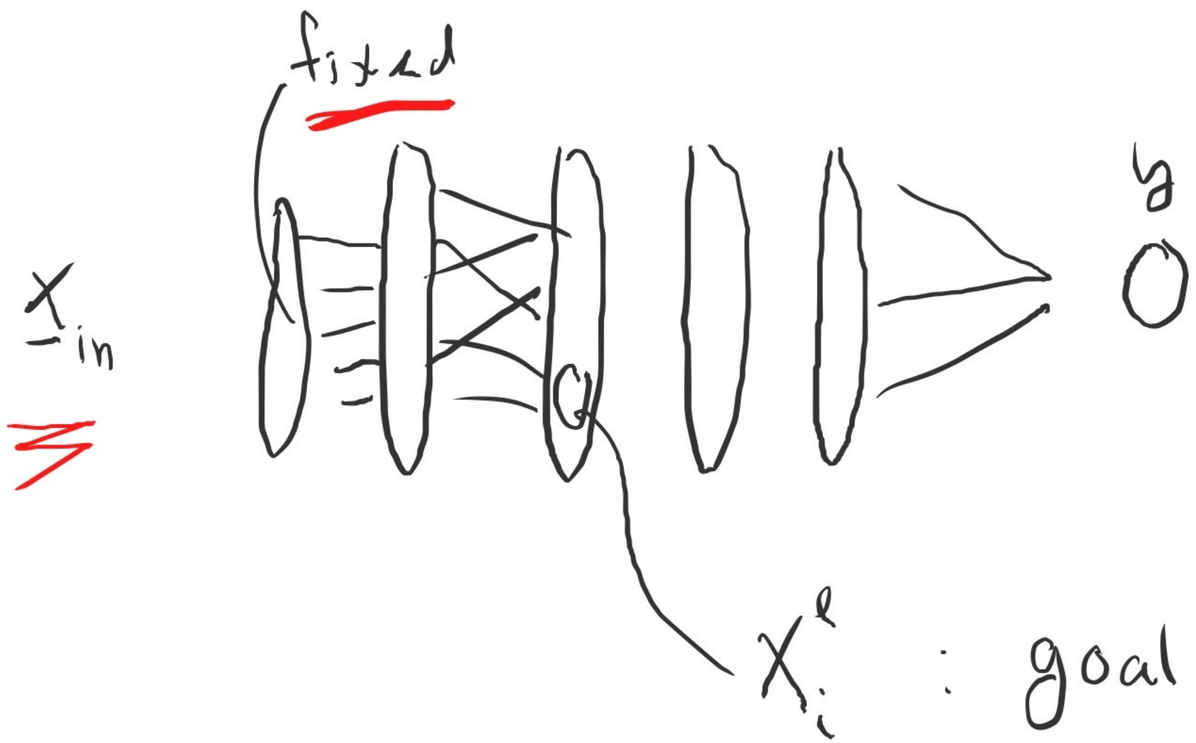


Tali Tishby: irrelevant information is getting squeezed out.

maximize $I(\underline{x}^l, y) - \beta I(\underline{x}_{in}, \underline{x}^l)$

parameter

replace Info w/ some other measure



$$\mathcal{L} = \frac{1}{2} \sum_{\ell, i} \left(x_i^\ell - y^*(\underline{x}_{in}) \right)^2$$

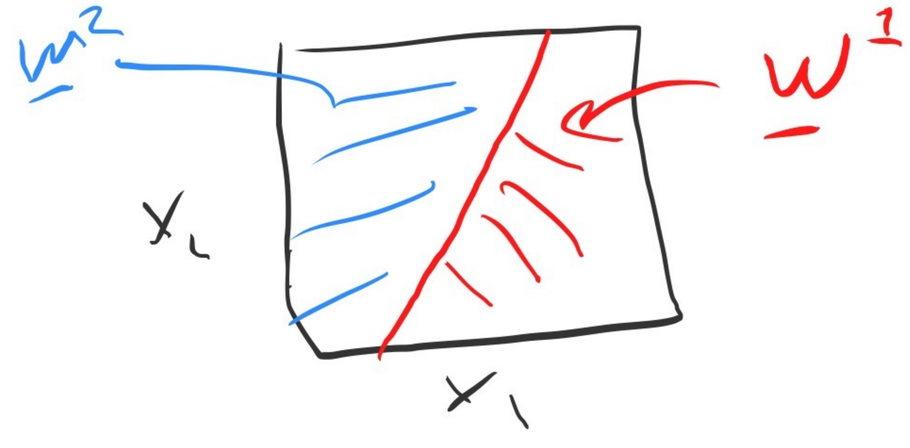
local: Hebb rule

$$= \frac{1}{2} \sum_{\ell, i} \left(\sum_j w_{ij}^{\ell-1} x_j^{\ell-1} - y^*(\underline{x}_{in}) \right)^2$$

$$\Delta w_{ij}^{\ell-1} = -\eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{\ell-1}} = -\eta \underbrace{(x_i^\ell - y^*)}_{\text{error (post)}} x_j^{\ell-1}$$ ← pre-synaptic

$$x_i^l = \sum_j w_{ij}^{l-1,c(\underline{x}_m)} \, x_j^{l-1}$$

- each neuron has a bank of weights
- which one it chooses depends on input
- pre-chosen

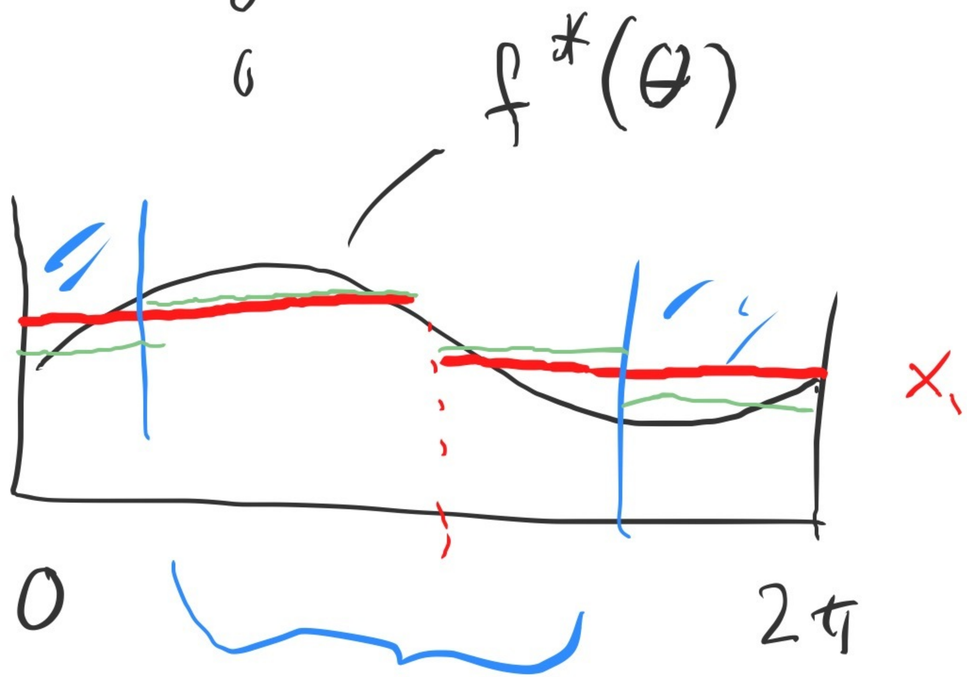


x_{1w} $x_1 = w_1 \cdot 1$ $x_2 = w_2 x_1$

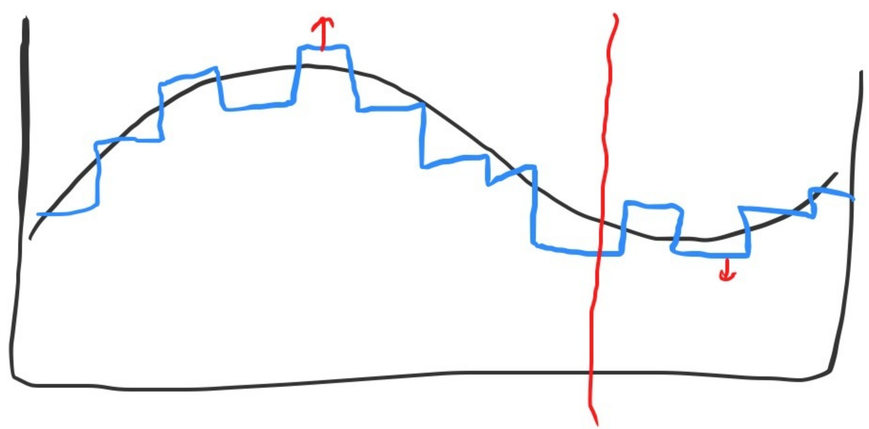
θ $\quad$ $1 \to 0$ $\quad$ 0 $\quad$ 0 $\quad$ 0 $\quad$ 0 $\quad$ 0 $\quad$ y

0

0



1. Convex
2. gets better with depth



GLN

gated linear network

$$r^l = f\left(\underline{\underline{w}}^l \cdot f^{-1}\left(\underline{r}^{l-1}\right)\right)$$

$$= f\left(\underline{\underline{w}}^l \cdot \underline{\underline{w}}^{l-1} \cdot f^{-1}\left(\underline{r}^{l-2}\right)\right)$$

each branch is gated on or off (inhibitory neurons)

DGN

dendritic gated network

Summary:

- backprop works really well
- not biologically plausible
  1. needs forward and backward pass
  2. neurons need to know their outgoing weights

lots of fixes

- mainly hacks
- GLM (provably converges - convex)
- what the brain uses is unknown

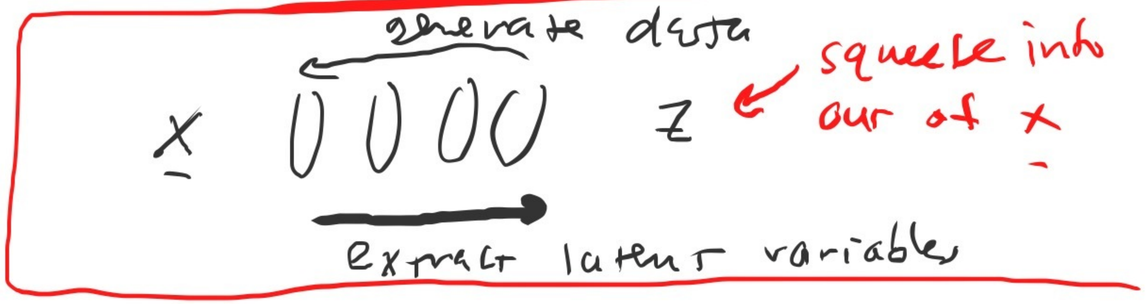
---

the brain mainly uses unsupervised learning

$\underset{\sim}{x}$, model $P_\theta(\underset{\sim}{x})$

$$\mathcal{L} = -\sum_{\{\underset{\sim}{x}\}} \log P_\theta(\underset{\sim}{x})$$

$$\Delta\theta = -\gamma \frac{\partial \mathcal{L}}{\partial \theta}$$



goal of most science:

bridge levels of description

neuroscience

lower level: neural activity

upper level: behavior

goal is to understand behavior in terms of neural activity

neurons (micro level)

$$\tau \frac{dV_i}{dt} = \text{(single neuron model)} - \sum_j W_{ij} (V_i - \Sigma_j) g_j(t)$$

↙ conductance changes due to pre-synaptic spikes

$\Delta W_{ij} = \ldots$

behavior = $f(\underline{S})$ (output neurons)

$\underline{S}$: spikes on all the neurons

10^{11} eqns

10-100 dimensional

Understand: solve analytically mapping from spikes to behavior

2 parts:

1. What are the equations? (including initial conditions)

2. What's the solution?

Weather prediction

Simplified equations

make up kind of reasonable equations and cross our fingers. (usually smoker)

biophysics

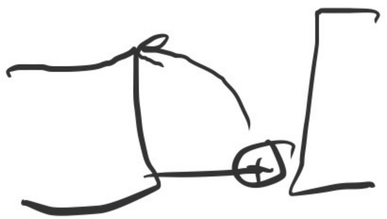
$$C\frac{dv}{dt} = I$$

$$I = -\sum_x g_x (v - \mathcal{E}_x)$$

conductance

reversal potential

derive all of biophysics once you know g_x



constant (leak)

voltage dependent (HH)

- add geometry (cable eqn.) → axons

concentration-dependence → synaptic dynamics

network

understand this quantity

$$\sum_j w_{ij} (v_i - \xi_j) g_j(t)$$

static piece

fast fluctuations

$$\longrightarrow \sum_j \tilde{w}_{ij} v_j + \sum_j \tilde{w}_{ij} \delta v_j(t)$$

famous nullcline

$\tilde{w}_{ij}$ ↖ random $\Rightarrow$ $\bar{v}_\Sigma, \bar{v}_\Sigma$ are all that matters

v_j ↖ not random

w_μ ↖ overlaps

↳ large sums are Gaussian

- weights fixed

- switch to learning
- we wrote down learning rules in supervised setting.
- now things get really hard

$$\frac{dW_{ij}}{dt} = f(\underline{W}) \quad \leftarrow \text{averaged over activity}$$

$\uparrow$ 10^{14}

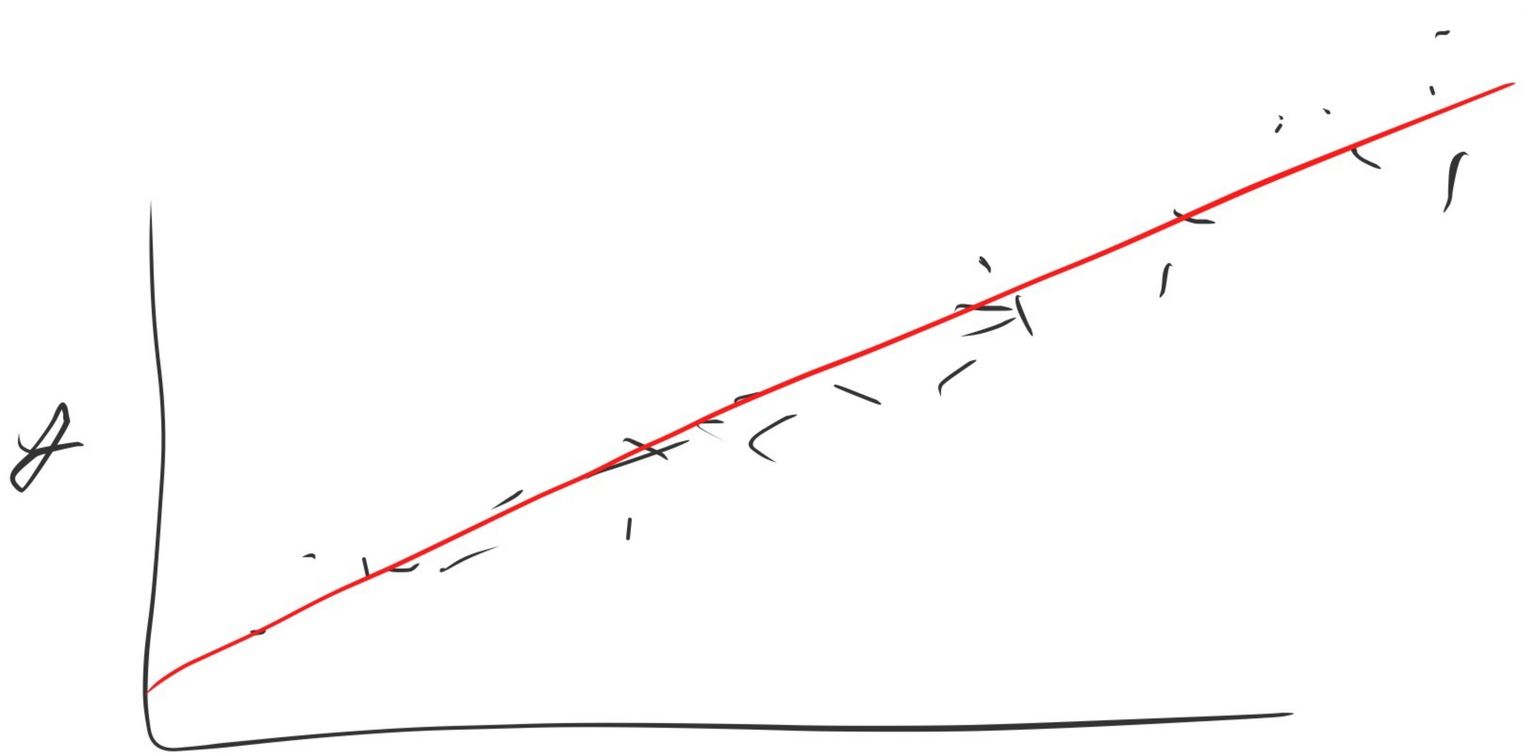
we don't know this one

$$\frac{dW_{ij}}{dt} = f(W_{ij}, \underline{x}) \quad \leftarrow \text{activity of lots of neuron}$$

$$f(W_{ij}, x_i, x_j) \quad \leftarrow \text{standard picture: pre-post}$$

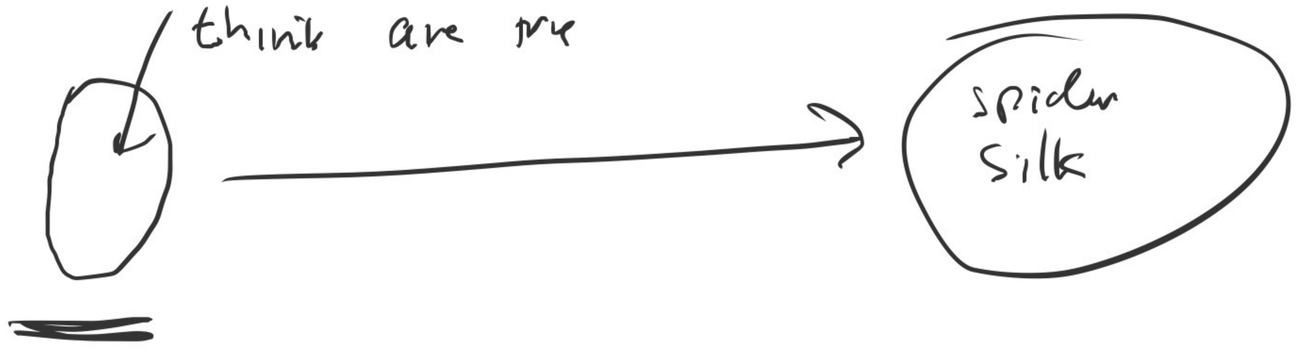
$$\frac{dx_i}{dt} = f\left(\sum_j W_{ij} x_j\right)$$

$\leftarrow$ equations highly nonlinear



x

if x, y, z true then



$$\frac{P}{n} = 0.138$$