Towards deep learning with segregated dendrites

Guergiev[1,2], Lillicrap[3] & Richards[1,2,4], eLife 2017

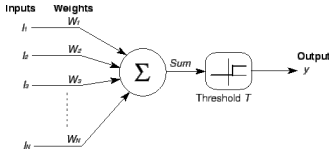
[1]Department of Biological Sciences, University of Toronto Scarborough, Toronto, Canada;
[2]Department of Cell and Systems Biology, University of Toronto, Toronto, Canada;
[3]DeepMind, London, United Kingdom;
[4]Learning in Machines and Brains Program, Canadian Institute for Advanced Research, Toronto, Canada

Jesse Geerts
Computational Hippocampus Journal Club
31 Jan 2018

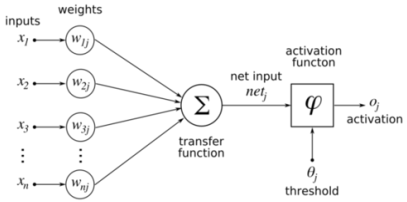
Outline

- **Introduction: comparing the brain to a neural network**
- The credit assignment problem in learning with networks
- Primer on the machine learning solution: back-propagation
- Biology's solution? Deep learning with segregated dendrites
- Results
- Discussion

Background: are artificial neural networks like the brain?



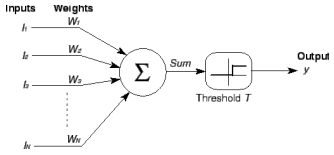
McCulloch & Pitts (1943):
"A logical calculus of the ideas immanent in nervous activity"



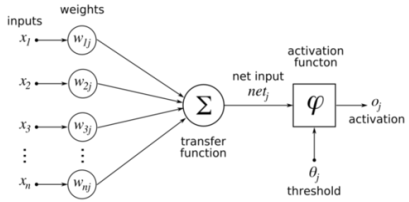
Rosenblatt (1958)
The perceptron

Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

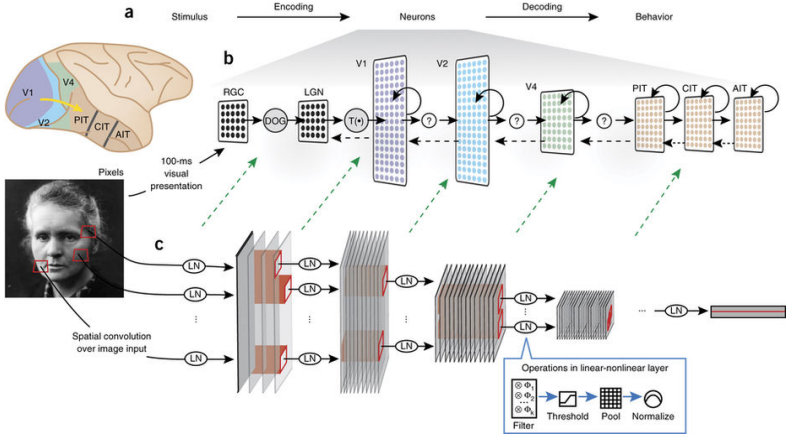
Background: are artificial neural networks like the brain?



McCulloch & Pitts (1943):
"A logical calculus of the ideas immanent in nervous activity"

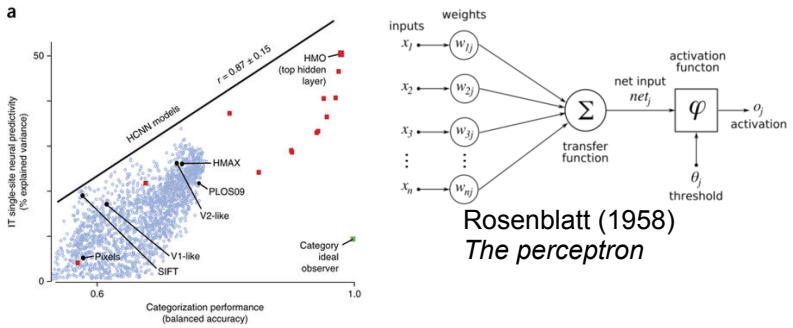


Rosenblatt (1958)
The perceptron

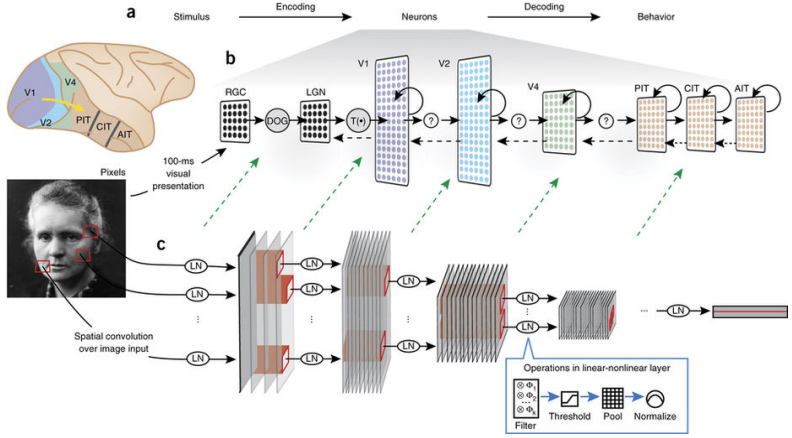


Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

Background: are artificial neural networks like the brain?

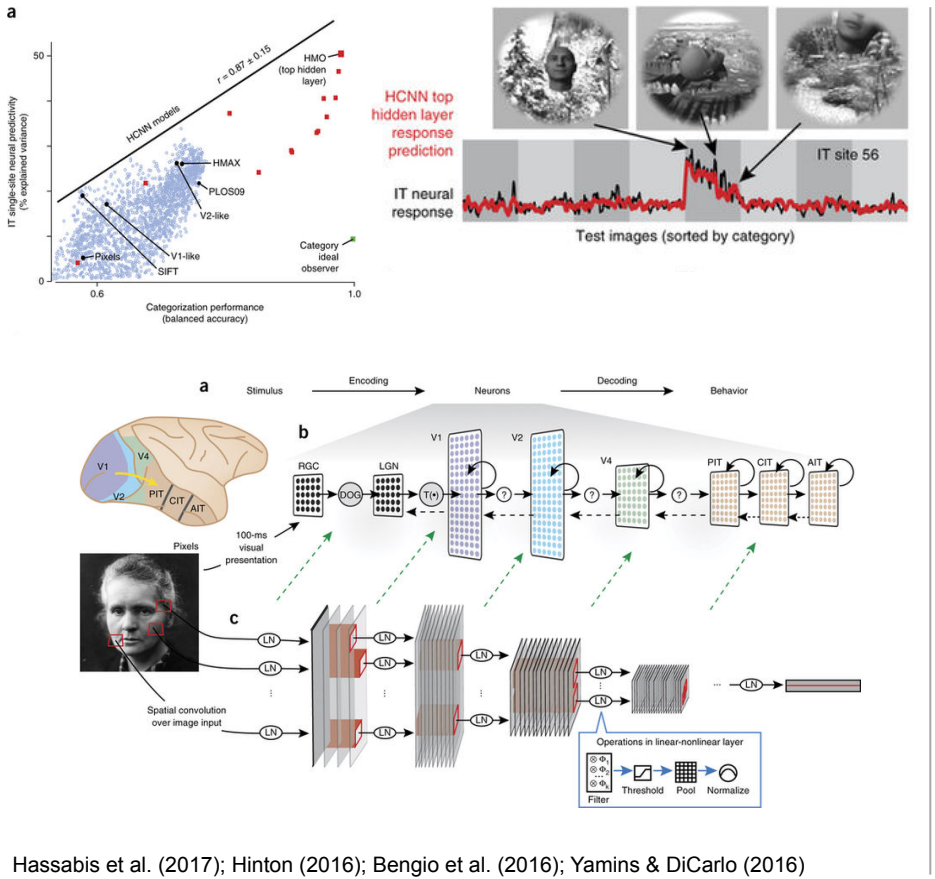


Rosenblatt (1958)
The perceptron



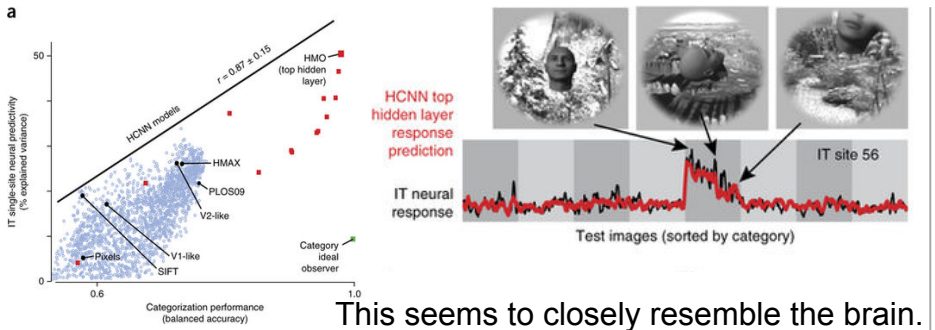
Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

Background: are artificial neural networks like the brain?

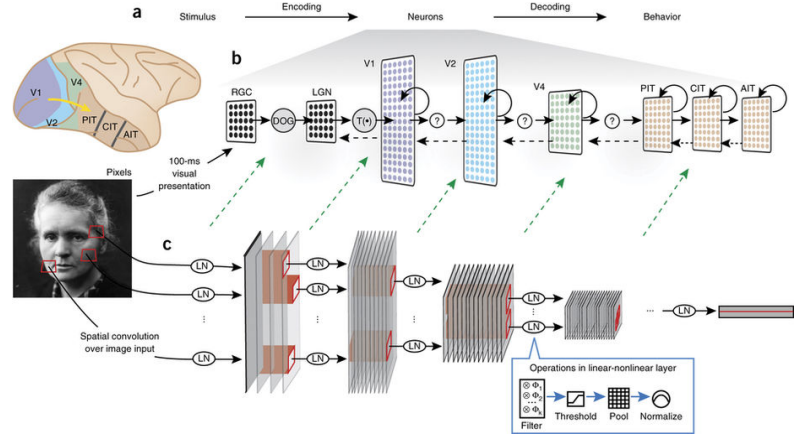


Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

Background: are artificial neural networks like the brain?

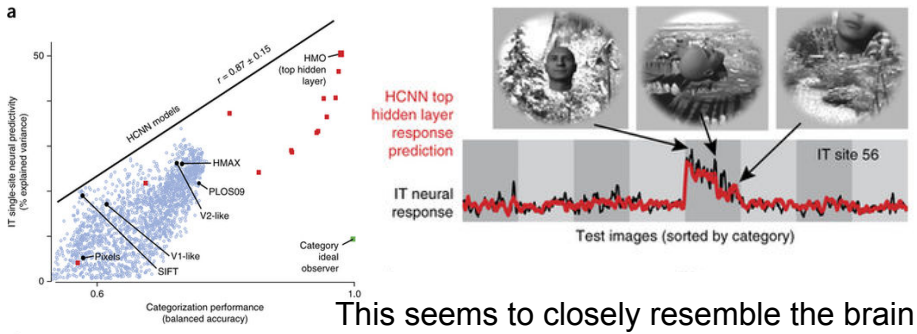


This seems to closely resemble the brain.
Does it work the same way?

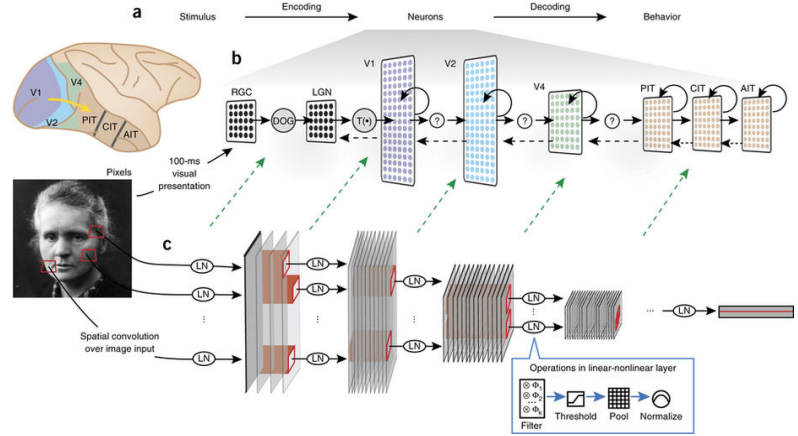


Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

Background: are artificial neural networks like the brain?



This seems to closely resemble the brain.
Does it work the same way?



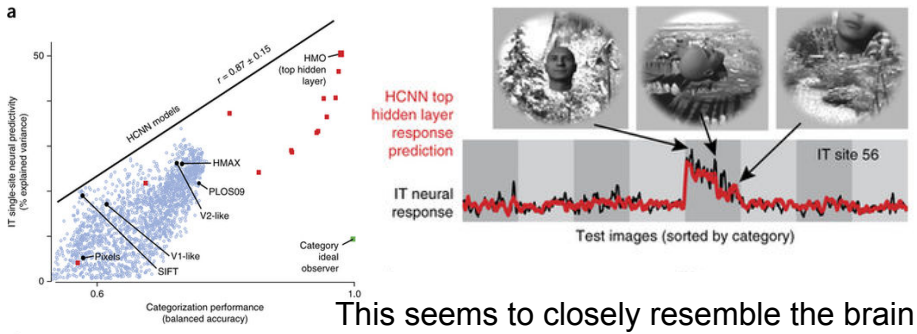
Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

NATURE VOL. 337 12 JANUARY 1989

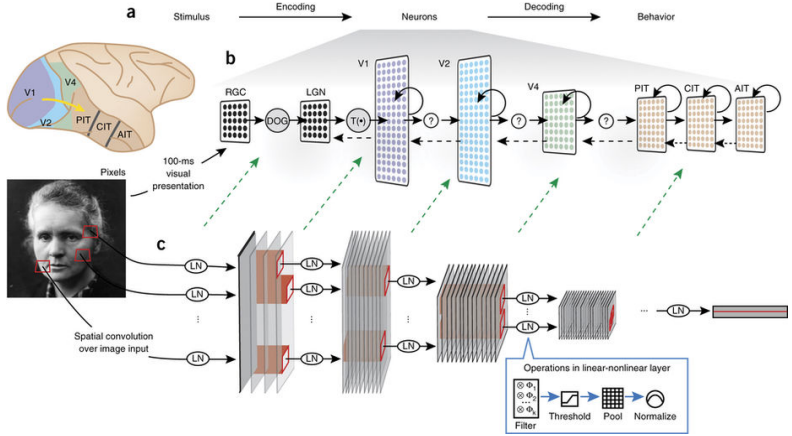
The recent excitement about neural networks

Francis Crick

Background: are artificial neural networks like the brain?



This seems to closely resemble the brain.
Does it work the same way?

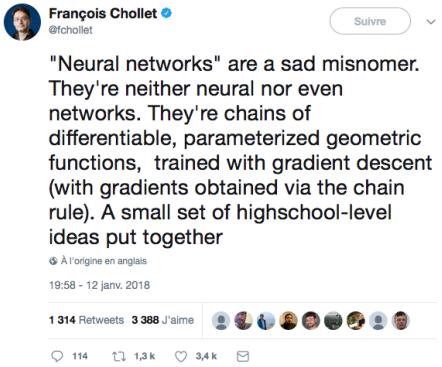


Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

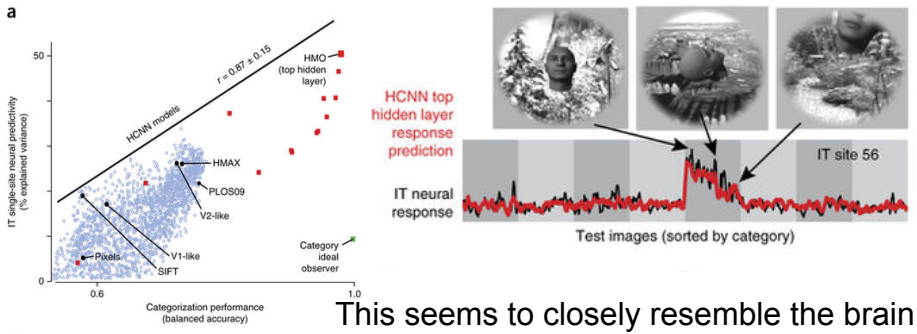
NATURE VOL. 337 12 JANUARY 1989 COMMENTARY 129

The recent excitement about neural networks

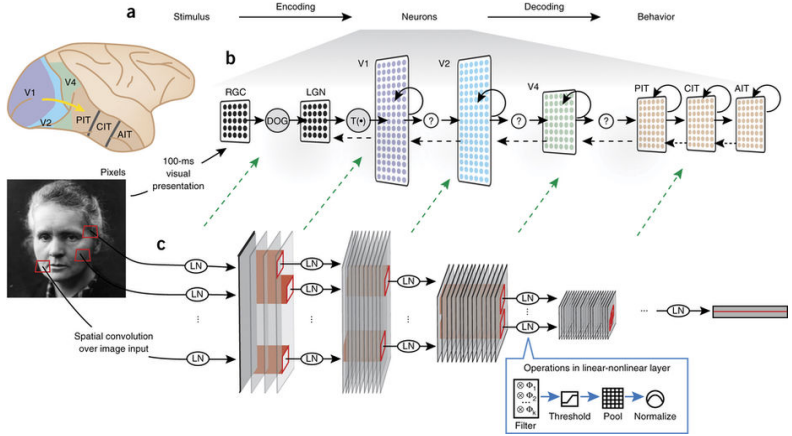
Francis Crick



Background: are artificial neural networks like the brain?



This seems to closely resemble the brain. Does it work the same way?



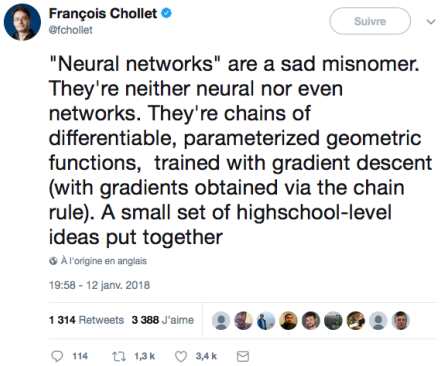
Hassabis et al. (2017); Hinton (2016); Bengio et al. (2016); Yamins & DiCarlo (2016)

NATURE VOL. 337 12 JANUARY 1989

COMMENTARY

The recent excitement about neural networks

Francis Crick



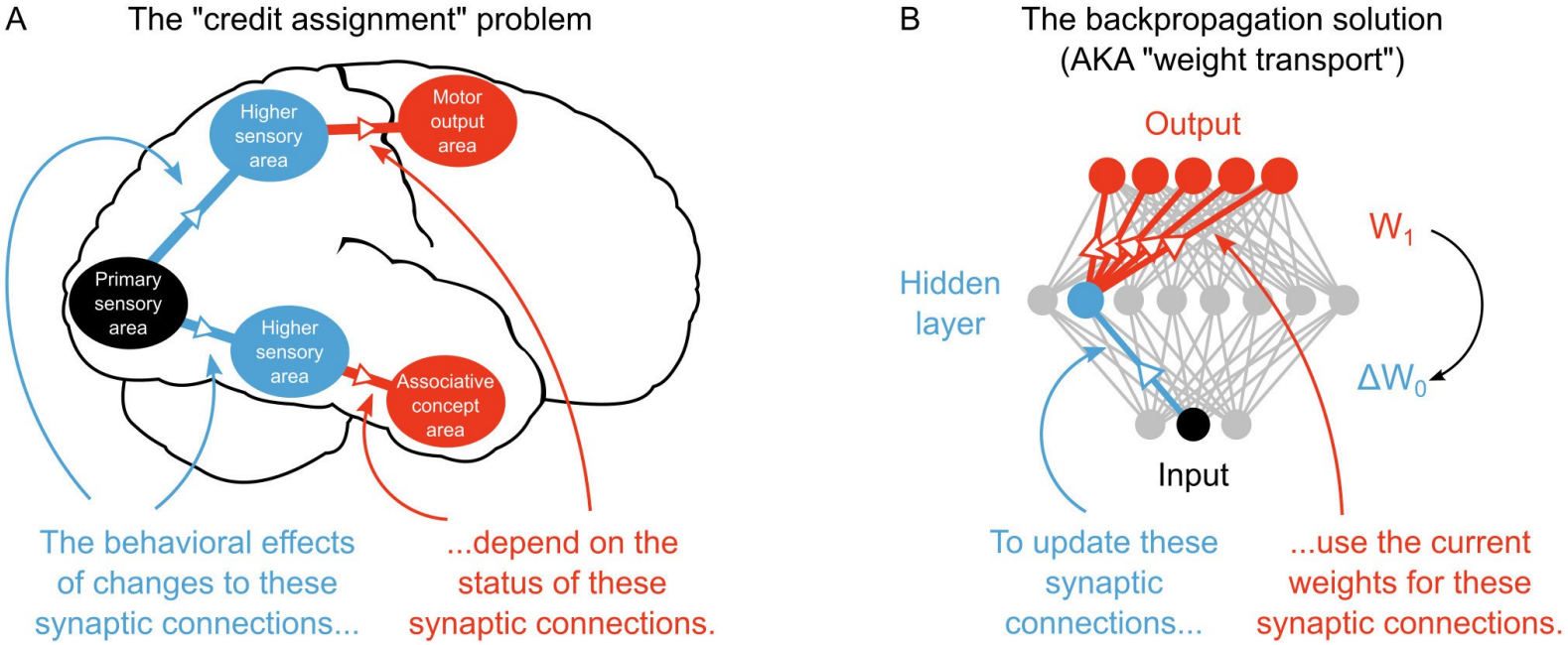
Central argument: Backpropagation is biologically unrealistic.

Crick (1987); Harris (2008); Urbanczik & Senn (2009)

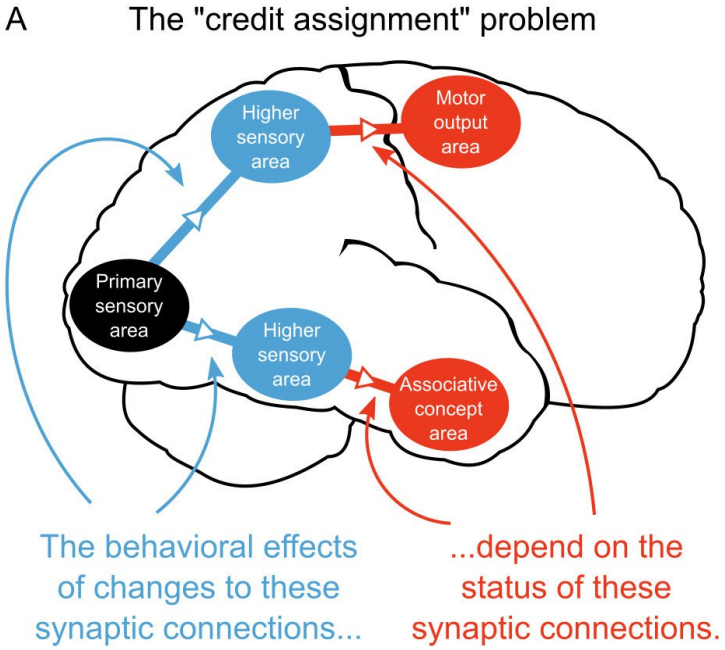
Outline

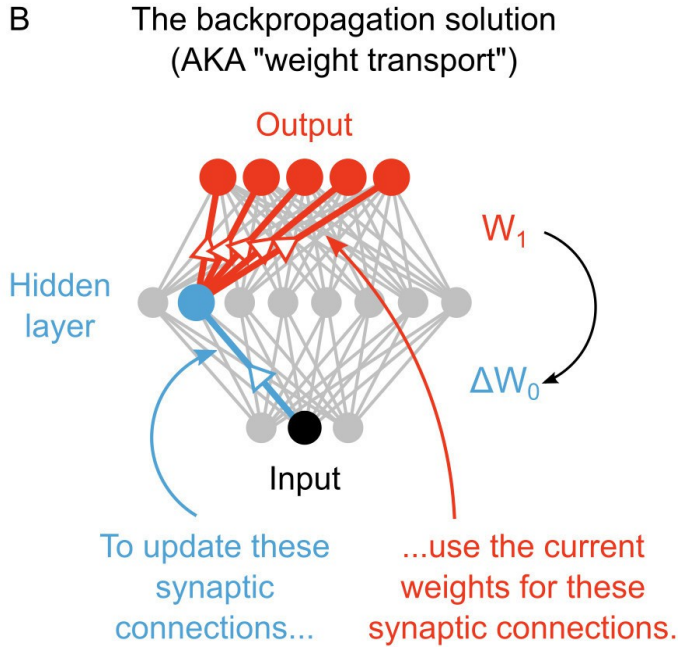
- Introduction: comparing the brain to a neural network
- **The credit assignment problem in learning**
- Primer on the machine learning solution: back-propagation
- Biology's solution? Deep learning with segregated dendrites
- Results
- Discussion

The credit-assignment problem



The credit-assignment problem



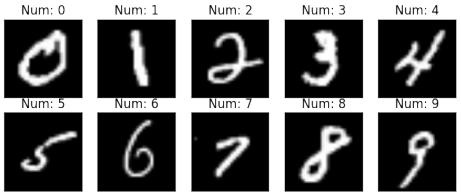


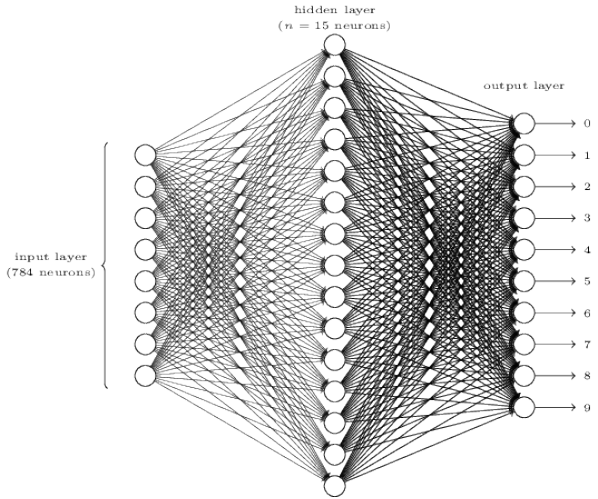
Unrealistic in real brains!

Outline

- Introduction: comparing the brain to a neural network
- The credit assignment problem in learning
- **Primer on the machine learning solution: back-propagation**
- Biology's solution? Deep learning with segregated dendrites
- Results
- Discussion

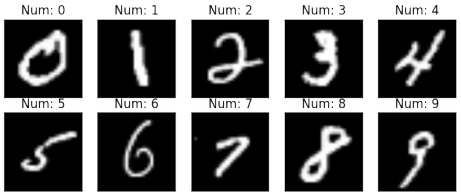
The architecture and terminology of a neural network

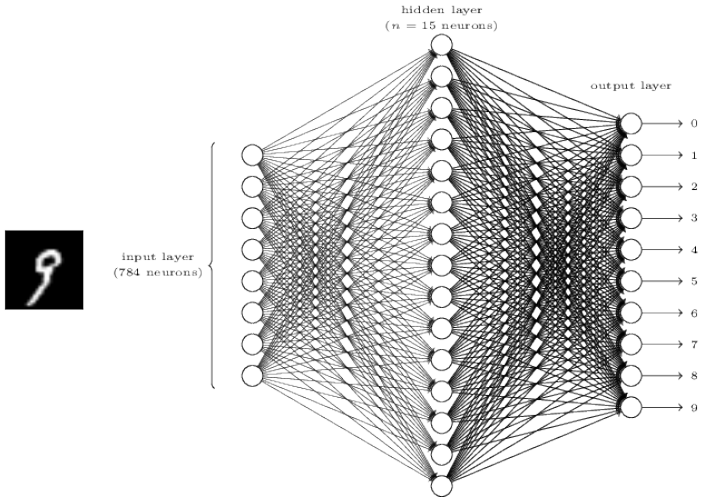




Input layer Hidden layer Output layer

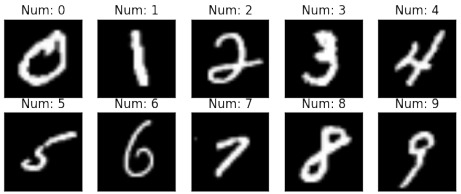
The architecture and terminology of a neural network

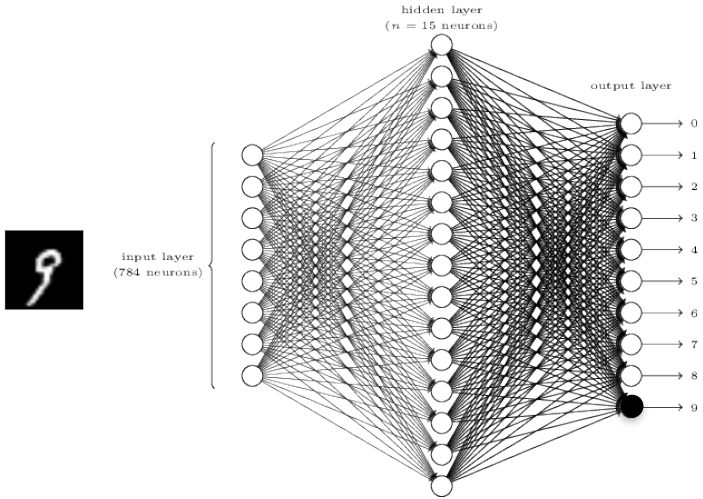




Input layer Hidden layer Output layer

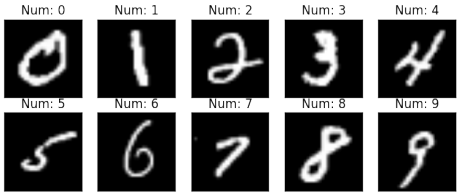
The architecture and terminology of a neural network

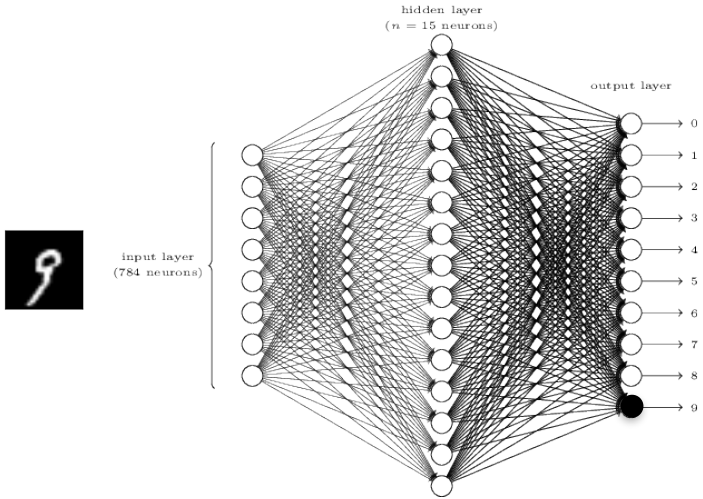




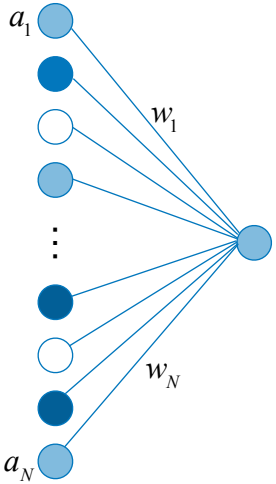
Input layer Hidden layer Output layer

The architecture and terminology of a neural network





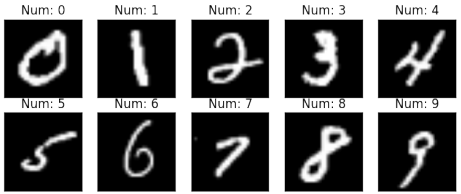
Input layer Hidden layer Output layer

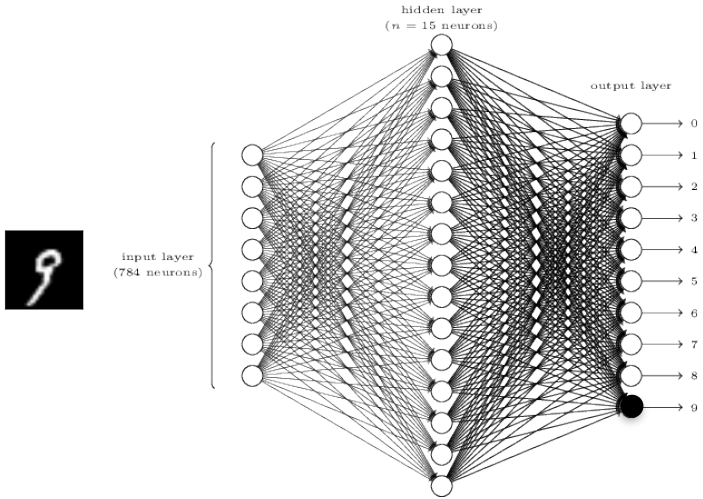


$$\sigma(a_1 w_1 + a_2 w_2 + ... + a_N w_N + bias)$$

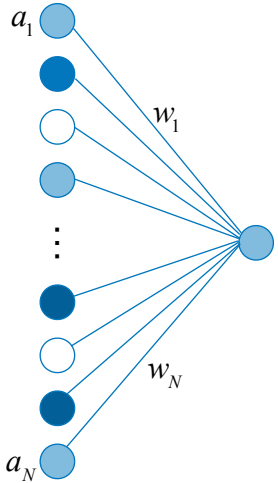
$$\sigma(w^T \vec{a}^l + b)$$

The architecture and terminology of a neural network





Input layer Hidden layer Output layer



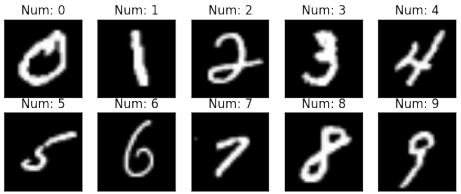
$$\sigma(a_1 w_1 + a_2 w_2 + ... + a_N w_N + bias)$$

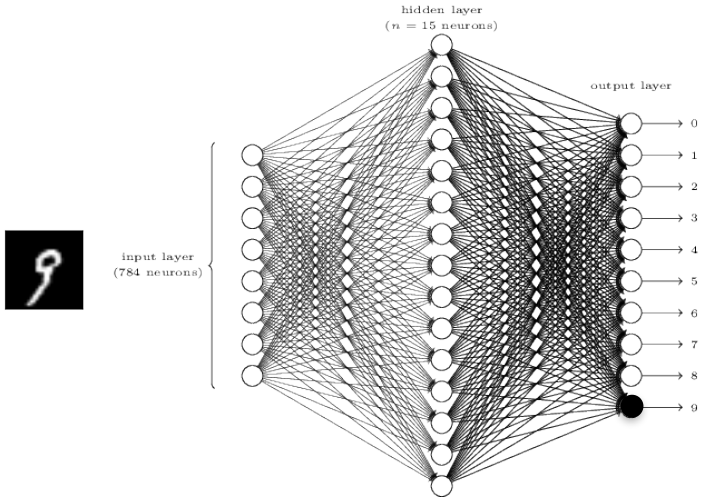
$$\sigma(w^T \vec{a}^l + b)$$

Training

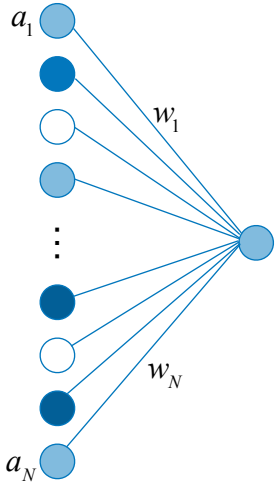
Testing

The architecture and terminology of a neural network





Input layer Hidden layer Output layer



$$\sigma(a_1 w_1 + a_2 w_2 + ... + a_N w_N + bias)$$

$$\sigma(w^T \vec{a}^l + b)$$

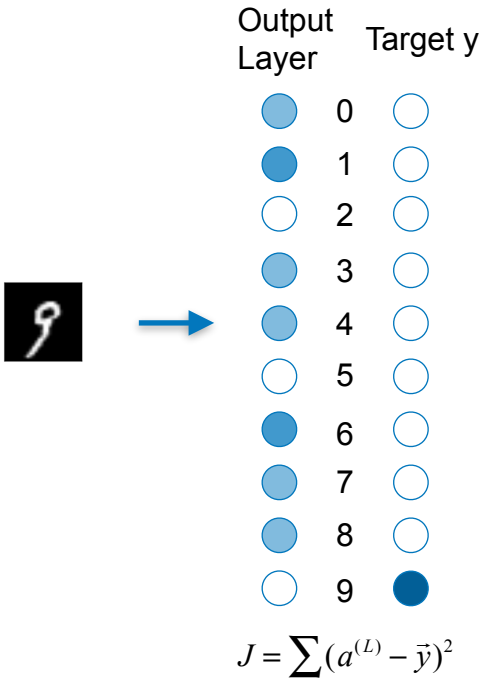
Training

Testing

23,860 parameters for a network with a single hidden layer with 30 neurons! How do we adjust these?

Backpropagation: credit (or blame) assignment in feedforward neural networks

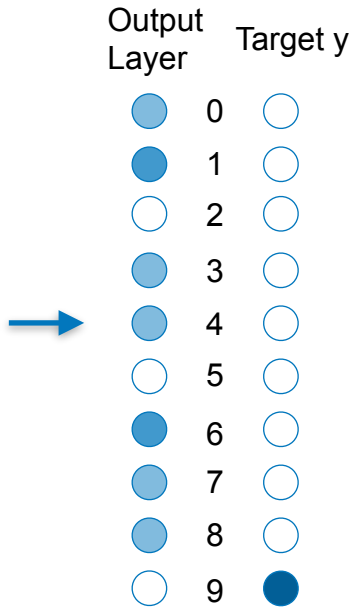
Minimising a cost function



$$J = \sum (a^{(L)} - \vec{y})^2$$

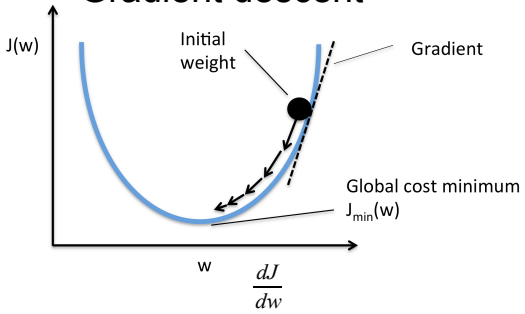
Backpropagation: credit (or blame) assignment in feedforward neural networks

Minimising a cost function



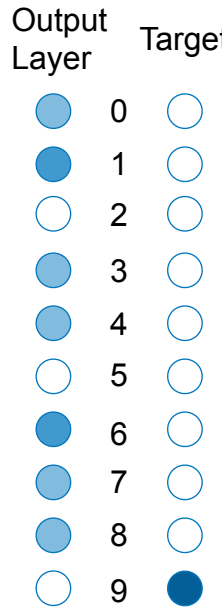
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent



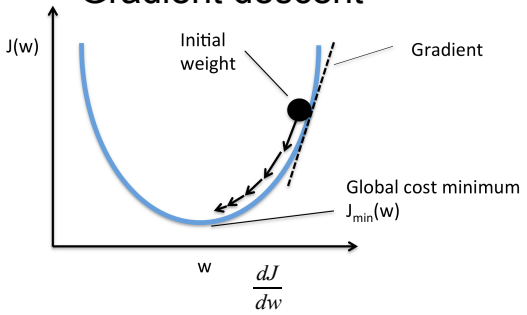
Backpropagation: credit (or blame) assignment in feedforward neural networks

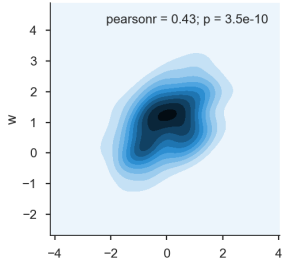
Minimising a cost function



$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

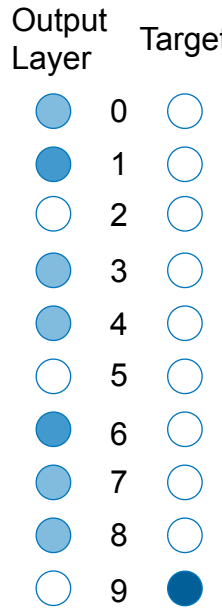




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

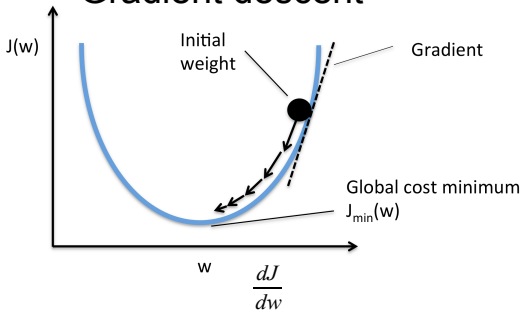
Backpropagation: credit (or blame) assignment in feedforward neural networks

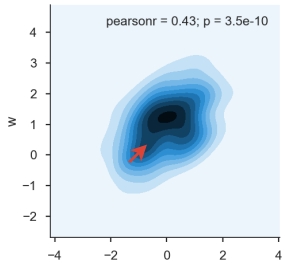
Minimising a cost function



$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

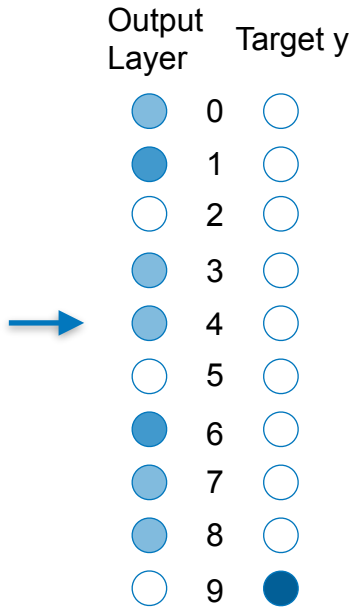




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

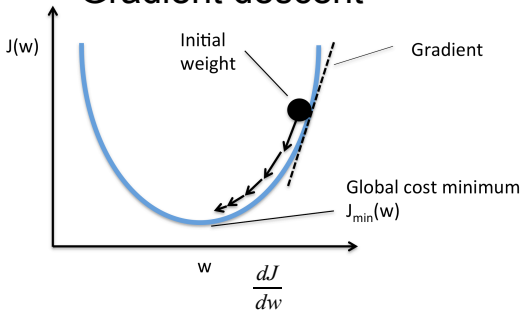
Backpropagation: credit (or blame) assignment in feedforward neural networks

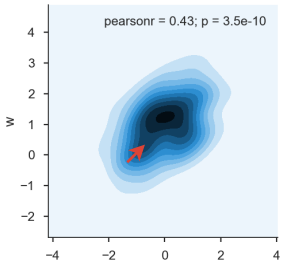
Minimising a cost function



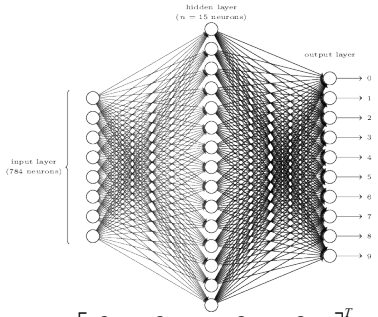
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent





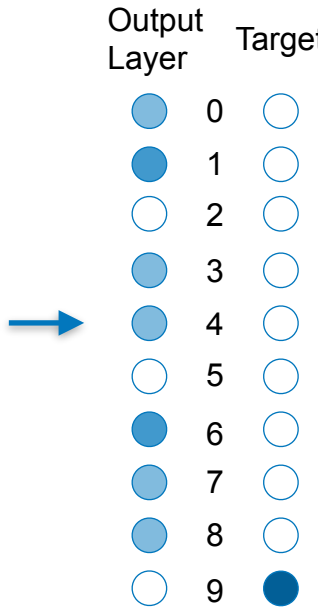
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$

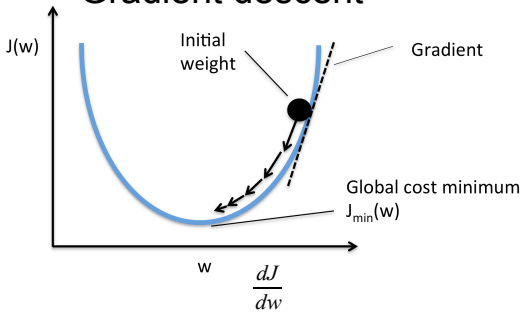
Backpropagation: credit (or blame) assignment in feedforward neural networks

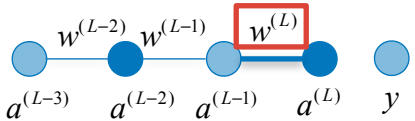
Minimising a cost function

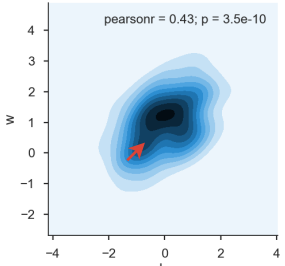


$$J = \sum (a^{(L)} - \vec{y})^2$$

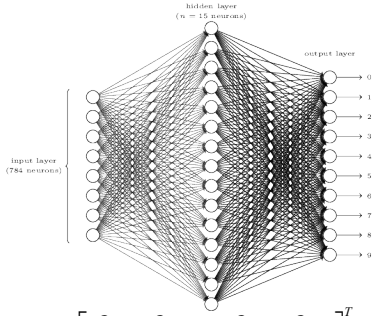
Gradient descent







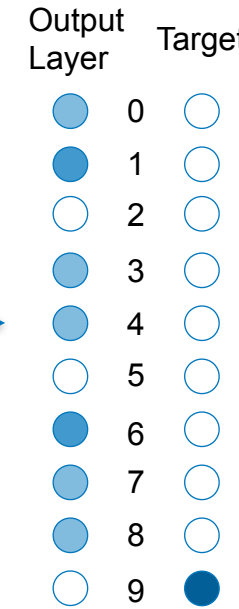
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$

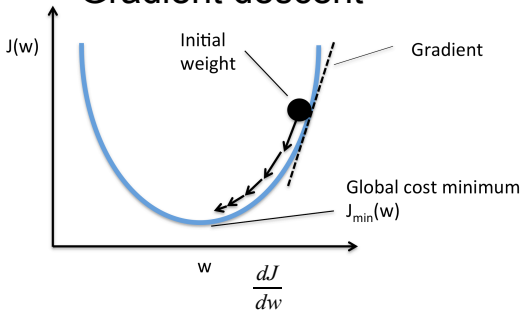
Backpropagation: credit (or blame) assignment in feedforward neural networks

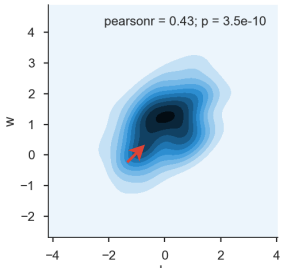
Minimising a cost function



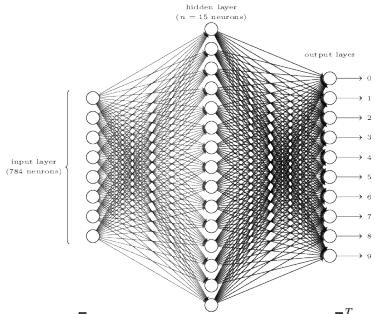
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

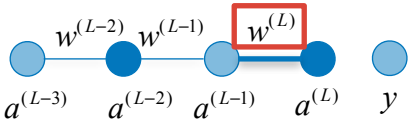




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



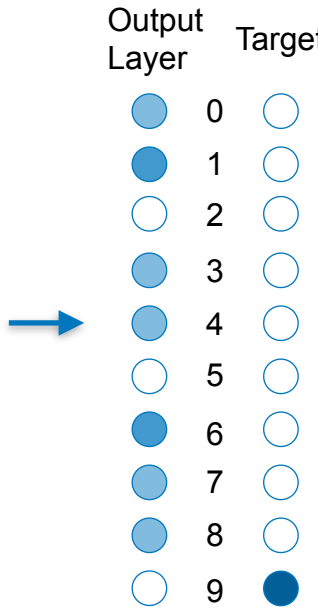
$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



How does the cost depend on changes in a specific weight?

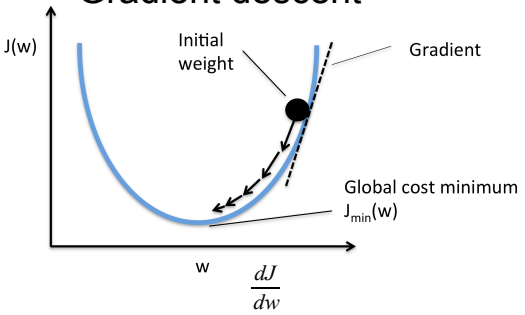
Backpropagation: credit (or blame) assignment in feedforward neural networks

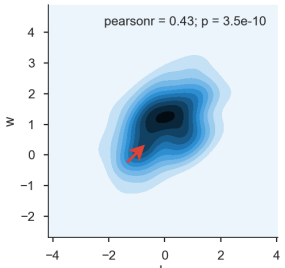
Minimising a cost function



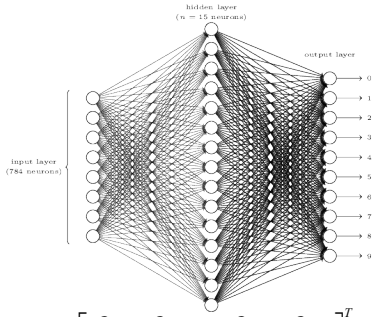
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

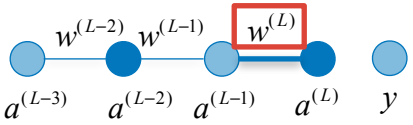




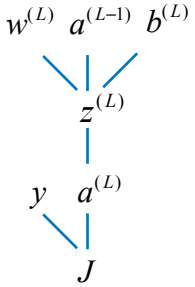
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$

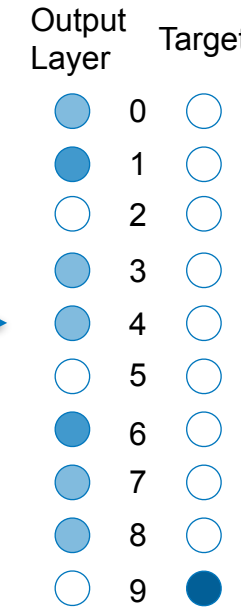


How does the cost depend on changes in a specific weight?



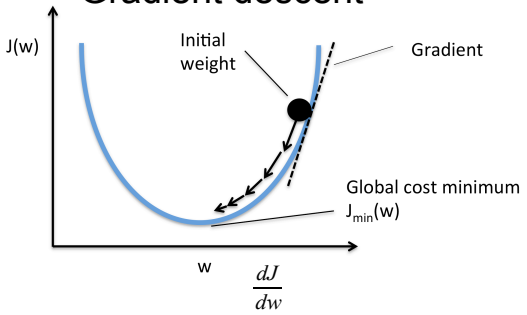
Backpropagation: credit (or blame) assignment in feedforward neural networks

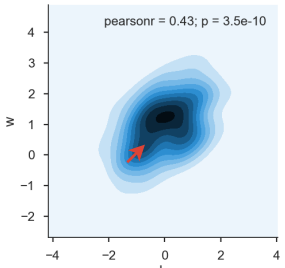
Minimising a cost function



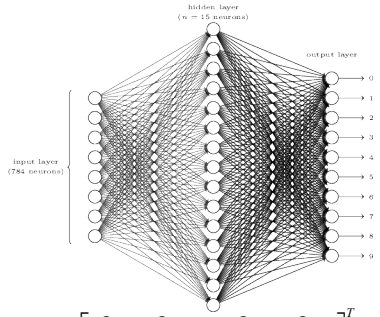
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

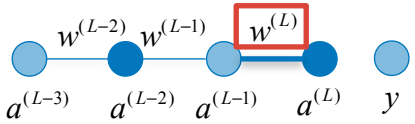




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

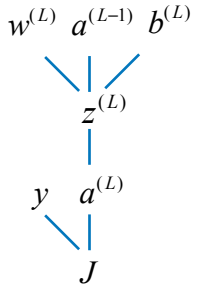


$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



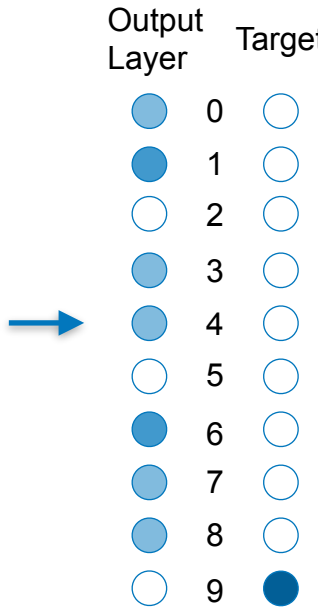
How does the cost depend on changes in a specific weight?

$$\frac{\partial J}{\partial w^{(L)}} =$$



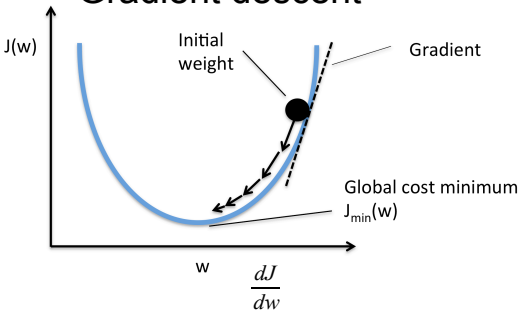
Backpropagation: credit (or blame) assignment in feedforward neural networks

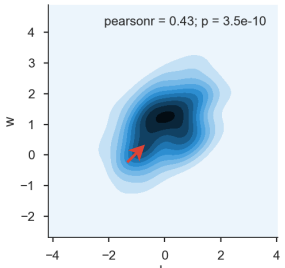
Minimising a cost function



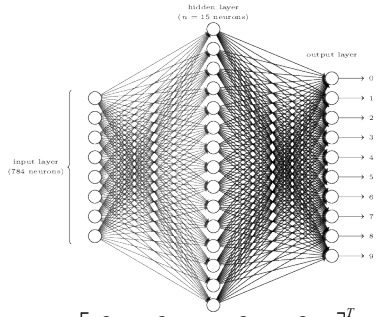
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

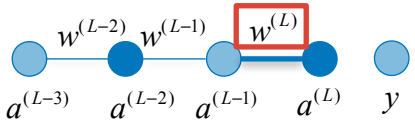




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

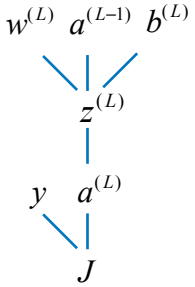


$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



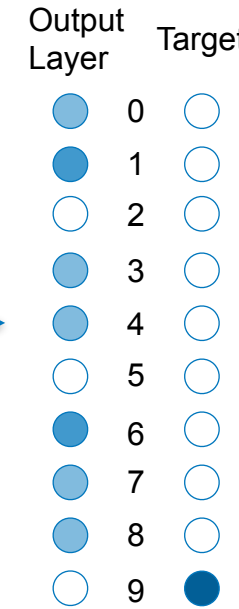
How does the cost depend on changes in a specific weight?

$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}}$$



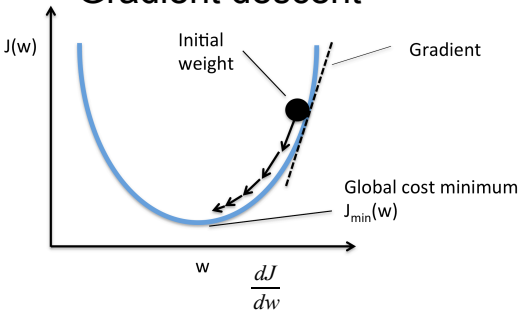
Backpropagation: credit (or blame) assignment in feedforward neural networks

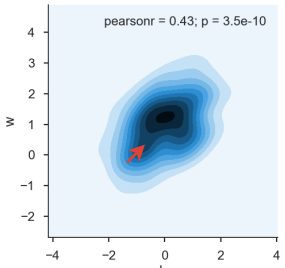
Minimising a cost function



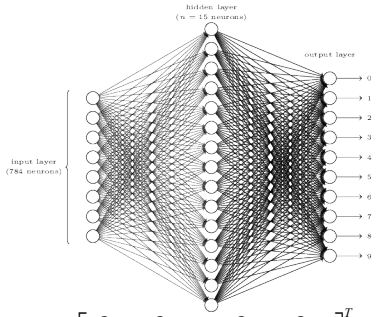
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

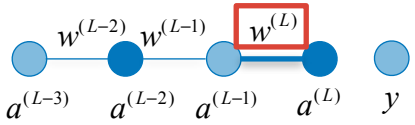




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

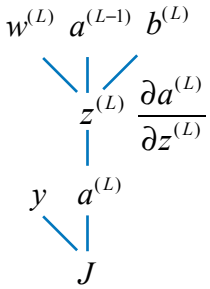


$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



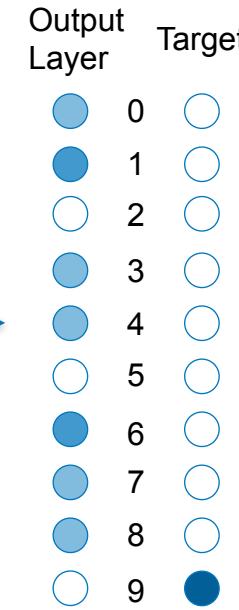
How does the cost depend on changes in a specific weight?

$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}}$$



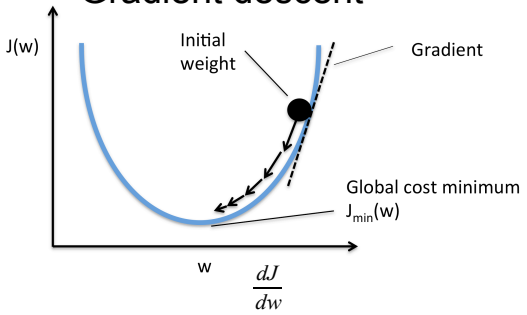
Backpropagation: credit (or blame) assignment in feedforward neural networks

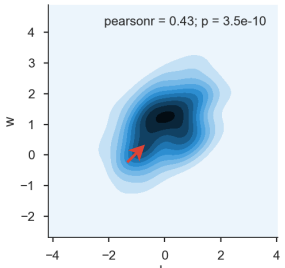
Minimising a cost function



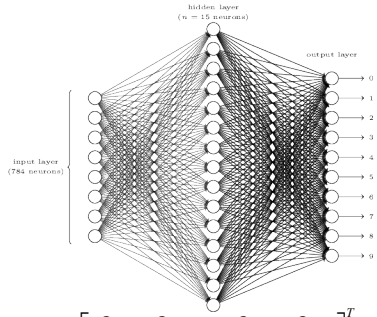
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

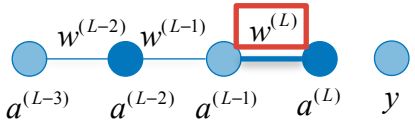




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

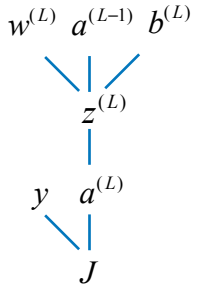


$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



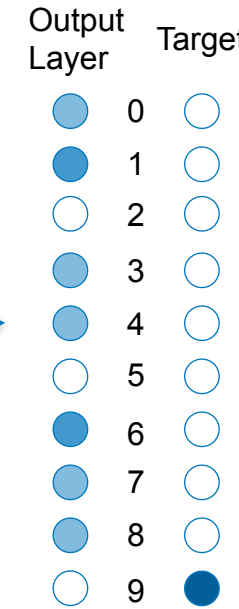
How does the cost depend on changes in a specific weight?

$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}}$$



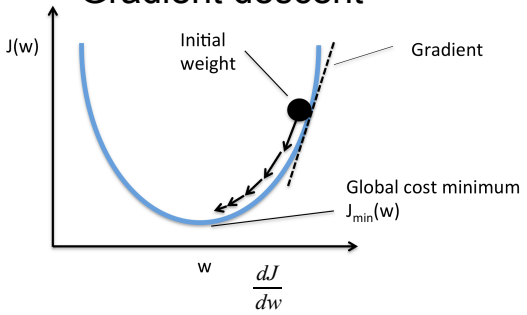
Backpropagation: credit (or blame) assignment in feedforward neural networks

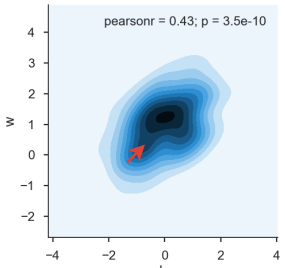
Minimising a cost function



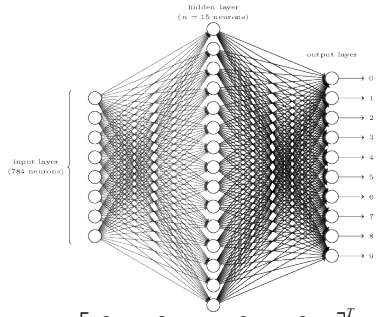
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

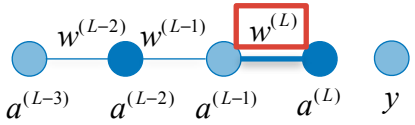




$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$

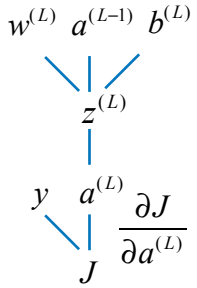


$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



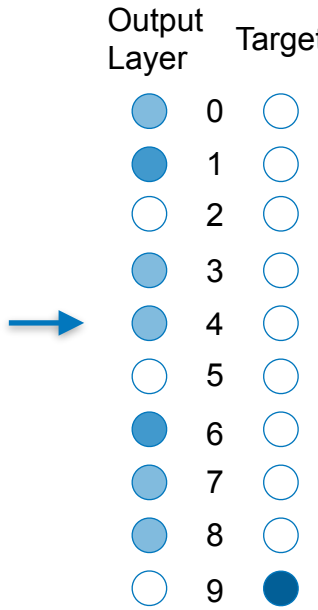
How does the cost depend on changes in a specific weight?

$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}}$$



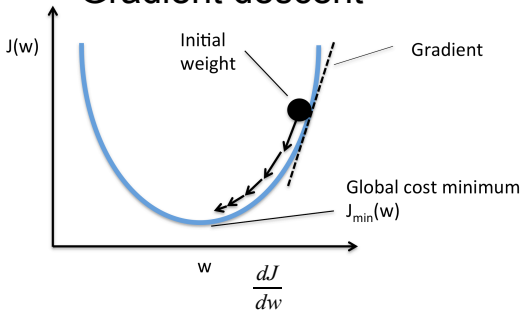
Backpropagation: credit (or blame) assignment in feedforward neural networks

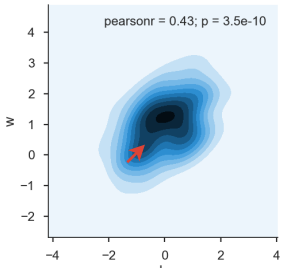
Minimising a cost function



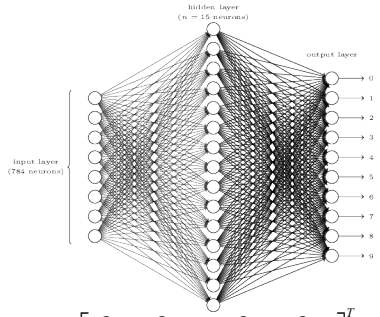
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

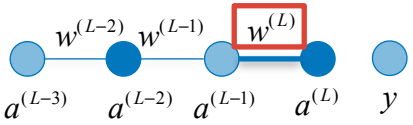




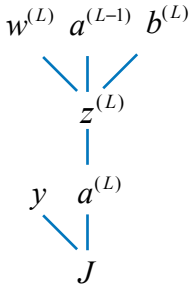
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



How does the cost depend on changes in a specific weight?

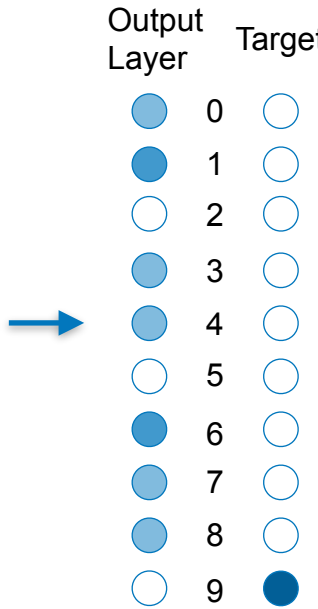


$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial J}{\partial a^{(L)}}$$

Chain rule!

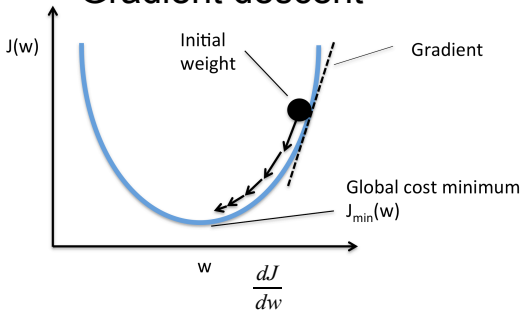
Backpropagation: credit (or blame) assignment in feedforward neural networks

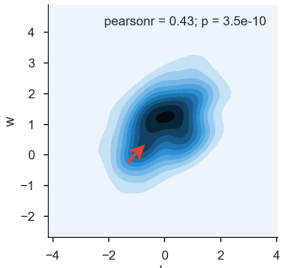
Minimising a cost function



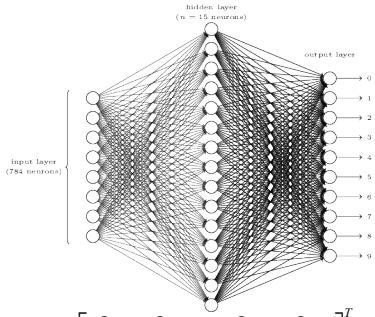
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

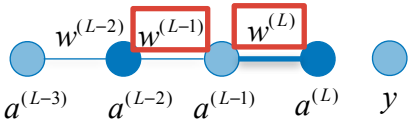




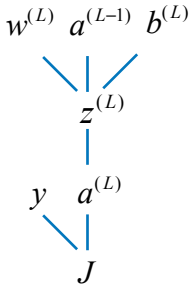
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



How does the cost depend on changes in a specific weight?



$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial J}{\partial a^{(L)}}$$

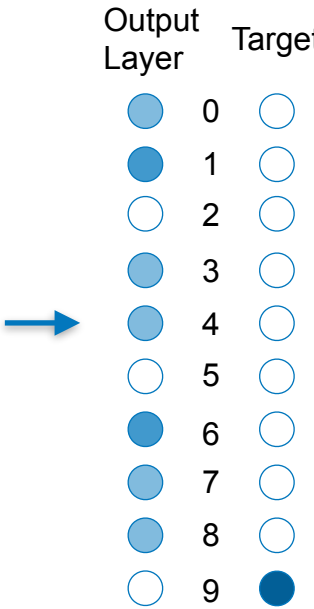
Chain rule!

$$\frac{\partial J}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial J}{\partial a^{(L-1)}}$$

$$= 2w^{(L)}\sigma'(z^{(L)})(a^{(L)} - y)$$

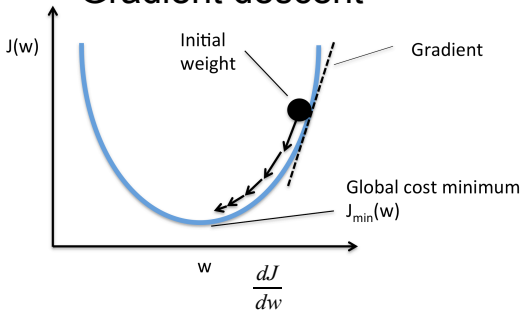
Backpropagation: credit (or blame) assignment in feedforward neural networks

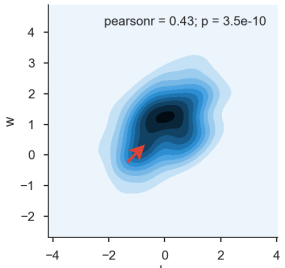
Minimising a cost function



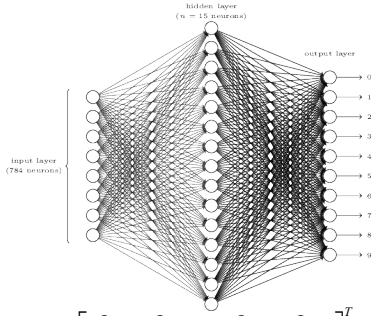
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

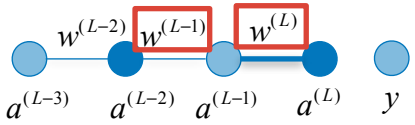




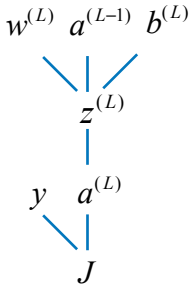
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



How does the cost depend on changes in a specific weight?



$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial J}{\partial a^{(L)}}$$

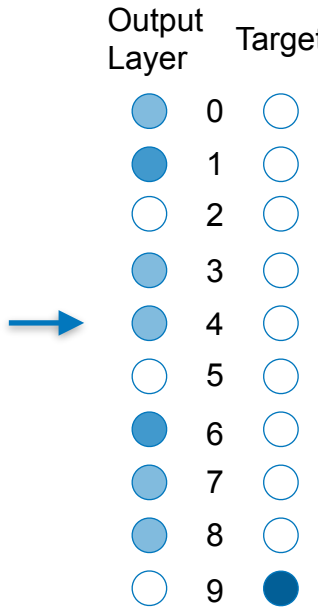
Chain rule!

$$\frac{\partial J}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial J}{\partial a^{(L-1)}}$$

$$= 2 w^{(L)} \sigma'(z^{(L)})(a^{(L)} - y)$$

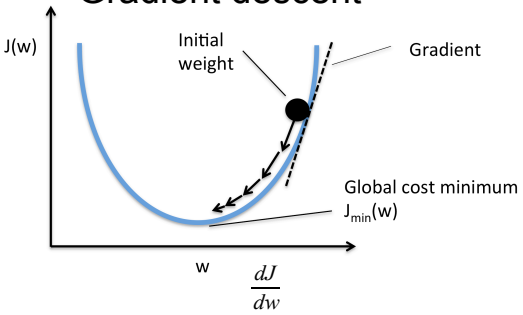
Backpropagation: credit (or blame) assignment in feedforward neural networks

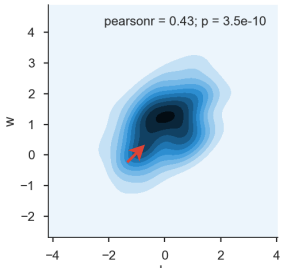
Minimising a cost function



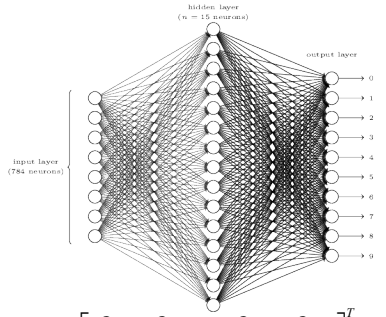
$$J = \sum (a^{(L)} - \vec{y})^2$$

Gradient descent

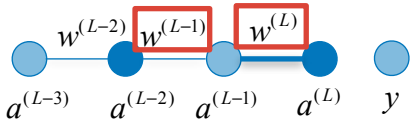




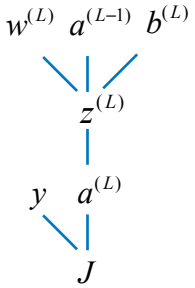
$$\nabla J(w,b) = \left[\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right]^T$$



$$\nabla J = \left[\frac{\partial J}{\partial w^{(1)}}, \frac{\partial J}{\partial b^{(1)}}, \ldots, \frac{\partial J}{\partial w^{(L)}}, \frac{\partial J}{\partial b^{(L)}}\right]^T$$



How does the cost depend on changes in a specific weight?



$$\frac{\partial J}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial J}{\partial a^{(L)}}$$

Chain rule!

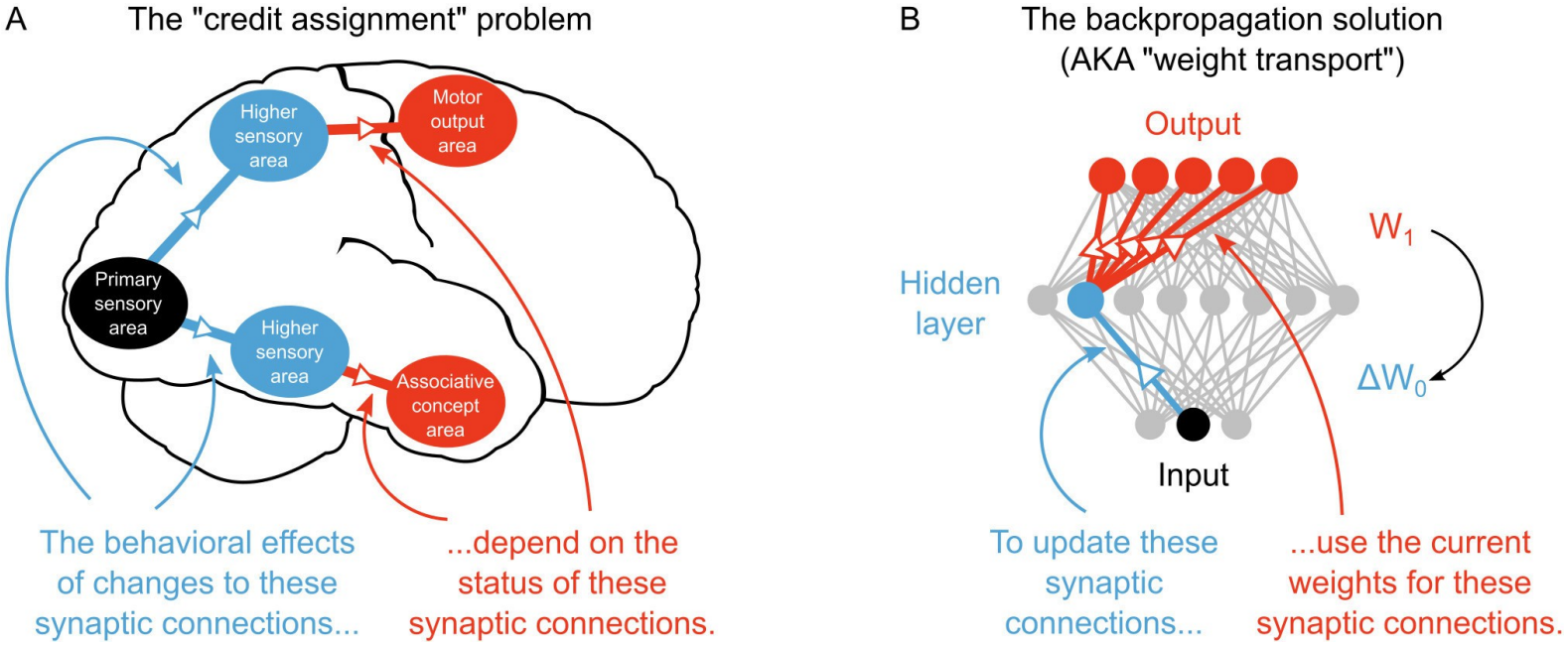
$$\frac{\partial J}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial J}{\partial a^{(L-1)}}$$

$$= 2w^{(L)}\sigma'(z^{(L)})(a^{(L)} - y)$$

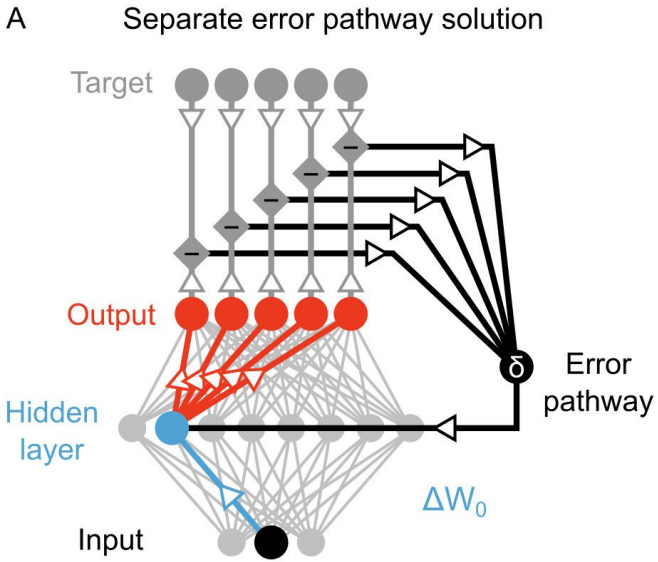
Backpropagation summary:

- Define a cost function
- Initialise weights and biases randomly
- Find gradient using chain rule, working backwards through the network
- Nudge weights and biases in the opposite direction
- Repeat for all training examples
- **Change in shallower layer weights depends on ALL deeper layer weights!**

The credit-assignment problem



Backpropagation in the brain with a separate error pathway is problematic



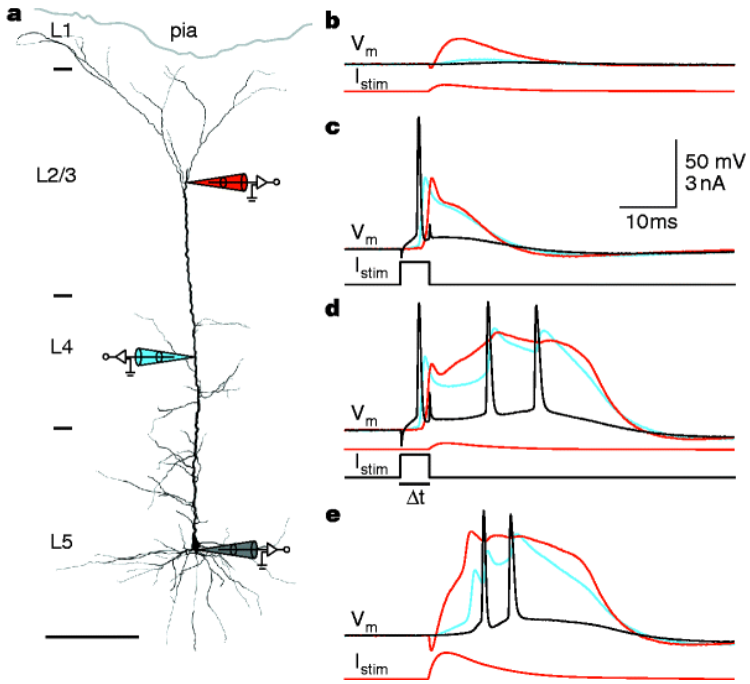
- An error neuron for every hidden layer neuron?
- The error signal is signed. Using a baseline firing rate would lead to noise when the error gets smaller

Lillicrap et al. (2016); Lee et al. (2015); Liao et al. (2015)

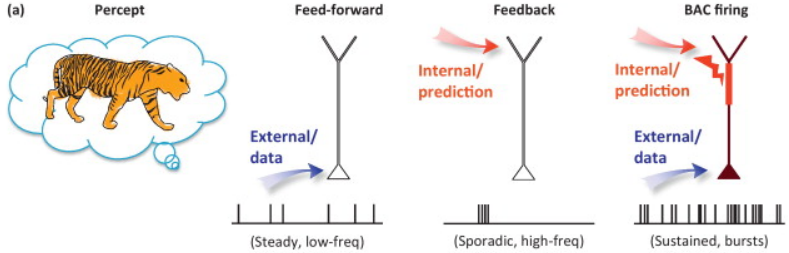
Outline

- Introduction: comparing the brain to a neural network
- The credit assignment problem in learning
- Primer on the machine learning solution: back-propagation
- **Biology's solution? Deep learning with segregated dendrites**
- Results
- Discussion

Calcium-driven dendritic spikes for associating inputs to an internal model

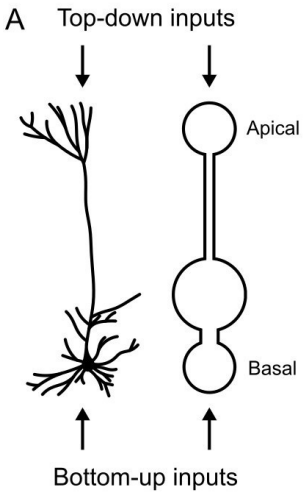


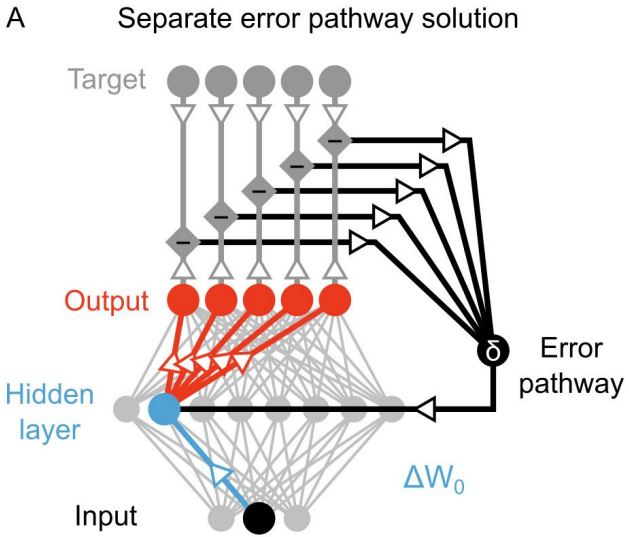
Larkum et al. (1999); Larkum (2013)

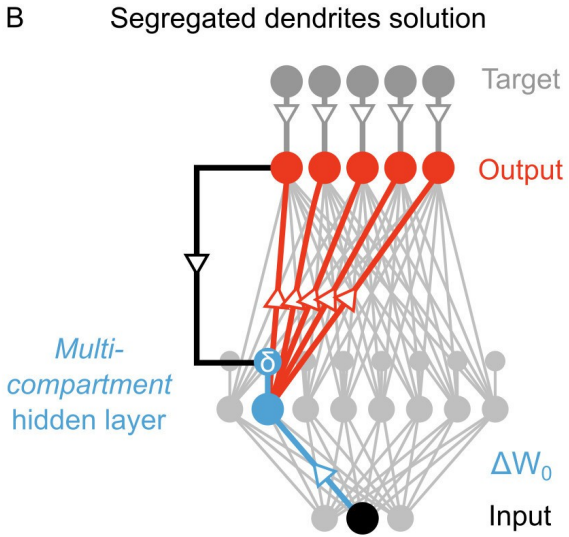


- Real pyramidal neurons have electronically distant apical dendrites
- Apical to soma transduction requires active propagation (through voltage gated Ca^{2+} channels).
- Bottom-up input arrives at basal dendrites, higher order feedback at apical dendrites (to some extent)
- Simultaneous apical and basal stimulation leads to plateau potentials

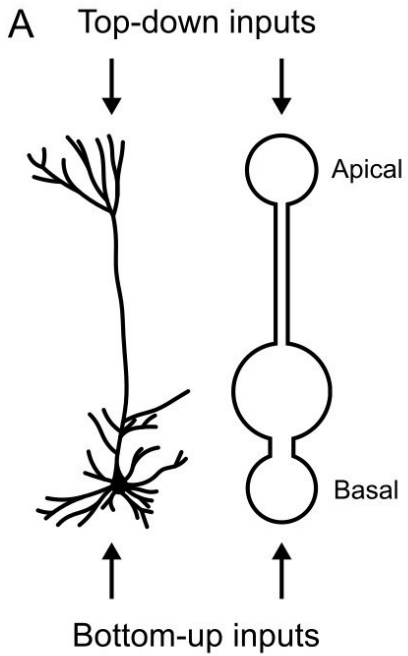
Solving the credit assignment problem with segregated dendrites

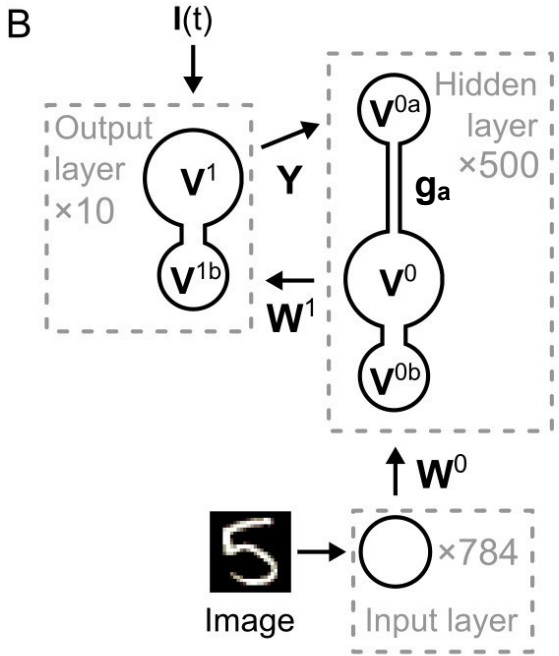


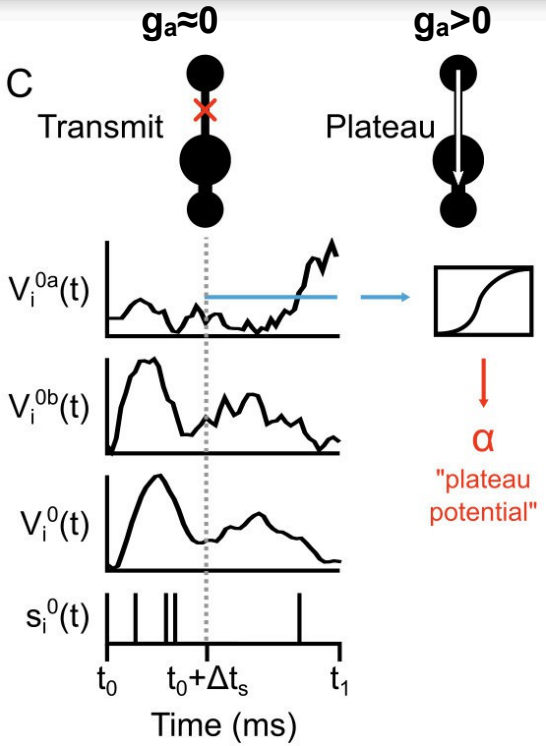




Basic model architecture: neurons with segregated dendritic compartments







Poisson spiking

$$\tau \frac{dV_i^0(t)}{dt} = -V_i^0(t) + \frac{g_b}{g_l}(V_i^{0b}(t) - V_i^0(t)) + \frac{g_a}{g_l}(V_i^{0a}(t) - V_i^0(t))$$

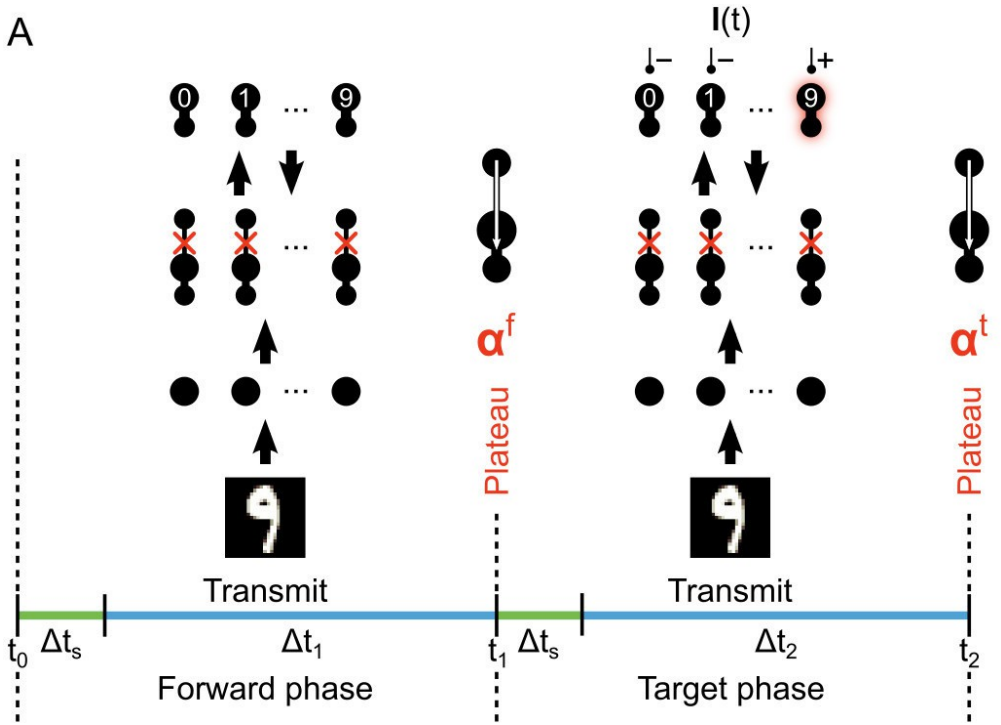
$$V_i^{0b}(t) = \sum_{j=1}^{\ell} W_{ij}^0 s_j^{\text{input}}(t) + b_i^0$$

$$V_i^{0a}(t) = \sum_{j=1}^{n} Y_{ij} s_j^1(t)$$

$$\tau \frac{dV_i^1(t)}{dt} = -V_i^1(t) + \frac{g_d}{g_l}(V_i^{1b}(t) - V_i^1(t)) + I_i(t)$$

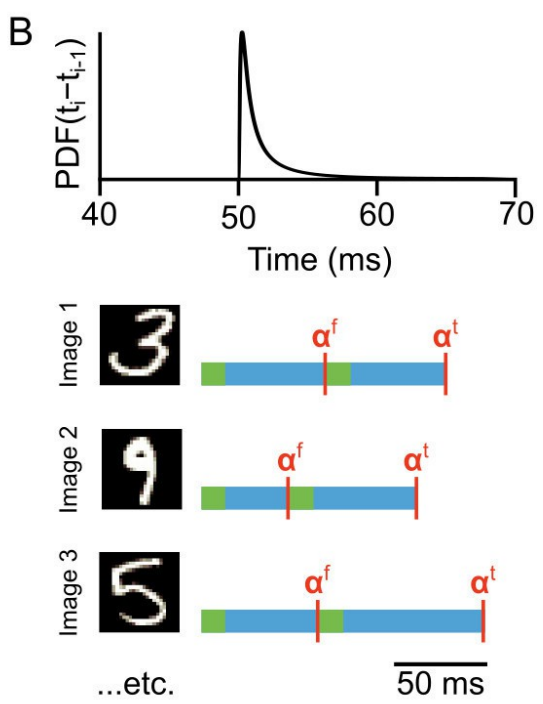
$$V_i^{1b}(t) = \sum_{j=1}^{\ell} W_{ij}^1 s_j^0(t) + b_i^1$$

Network learning in two distinct phases



$$\alpha_i^f = \sigma\left(\frac{1}{\Delta t_1}\int_{t_1-\Delta t_1}^{t_1} V_i^{0a}(t)dt\right)$$

$$\alpha_i^t = \sigma\left(\frac{1}{\Delta t_2}\int_{t_2-\Delta t_2}^{t_2} V_i^{0a}(t)dt\right)$$



Defining the local loss functions

*Actual loss functions more complicated and shown in methods

Defining the local loss functions

Output layer target and loss

$$\begin{aligned}\phi_i^{1*} &= \overline{\phi_i^1}^f \\ &= \frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \phi_i^1(t)dt\end{aligned}$$

$$\begin{aligned}L^1 &\approx ||\boldsymbol{\phi}^{1*} - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^1}^f - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= \left|\left|\frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \boldsymbol{\phi}^1(t)dt - \frac{1}{\Delta t_1}\int_{t_0+\Delta t_s}^{t_1} \boldsymbol{\phi}^1(t)dt\right|\right|_2^2\end{aligned}$$

*Actual loss functions more complicated and shown in methods

Defining the local loss functions

Output layer target and loss

$$\begin{aligned}\phi_i^{1*} &= \overline{\phi_i^1}^t \\ &= \frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \phi_i^1(t)dt\end{aligned}$$

$$\begin{aligned}L^1 &\approx ||\boldsymbol{\phi}^{1*} - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^1}^t - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= \left|\left|\frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \boldsymbol{\phi}^1(t)dt - \frac{1}{\Delta t_1}\int_{t_0+\Delta t_s}^{t_1} \boldsymbol{\phi}^1(t)dt\right|\right|_2^2\end{aligned}$$

Hidden layer target and loss

$$\phi_i^{0*} = \overline{\phi_i^0}^f + \alpha_i^t - \alpha_i^f$$

$$\begin{aligned}L^0 &\approx ||\boldsymbol{\phi}^{0*} - \overline{\boldsymbol{\phi}^0}^f||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^0}^f + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}_i^f - \overline{\boldsymbol{\phi}^0}^f||_2^2 \\ &= ||\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f||_2^2\end{aligned}$$

*Actual loss functions more complicated and shown in methods

Defining the local loss functions

Output layer target and loss

$$\begin{aligned}\phi_i^{1*} &= \overline{\phi_i^1}^t \\ &= \frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \phi_i^1(t)dt\end{aligned}$$

$$\begin{aligned}L^1 &\approx ||\boldsymbol{\phi}^{1*} - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^1}^t - \overline{\boldsymbol{\phi}^1}^f||_2^2 \\ &= \left|\left|\frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \boldsymbol{\phi}^1(t)dt - \frac{1}{\Delta t_1}\int_{t_0+\Delta t_s}^{t_1} \boldsymbol{\phi}^1(t)dt\right|\right|_2^2\end{aligned}$$

$$\Delta W^1 = -\eta_0 \frac{\partial L^1}{\partial W^1}$$

$$\Delta W^0 = -\eta_1 \frac{\partial L^0}{\partial W^0}$$

Hidden layer target and loss

$$\phi_i^{0*} = \overline{\phi_i^0}^f + \alpha_i^t - \alpha_i^f$$

$$\begin{aligned}L^0 &\approx ||\boldsymbol{\phi}^{0*} - \overline{\boldsymbol{\phi}^0}^f||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^0}^f + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}_i^f - \overline{\boldsymbol{\phi}^0}^f||_2^2 \\ &= ||\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f||_2^2\end{aligned}$$

*Actual loss functions more complicated and shown in methods

Defining the local loss functions

Output layer target and loss

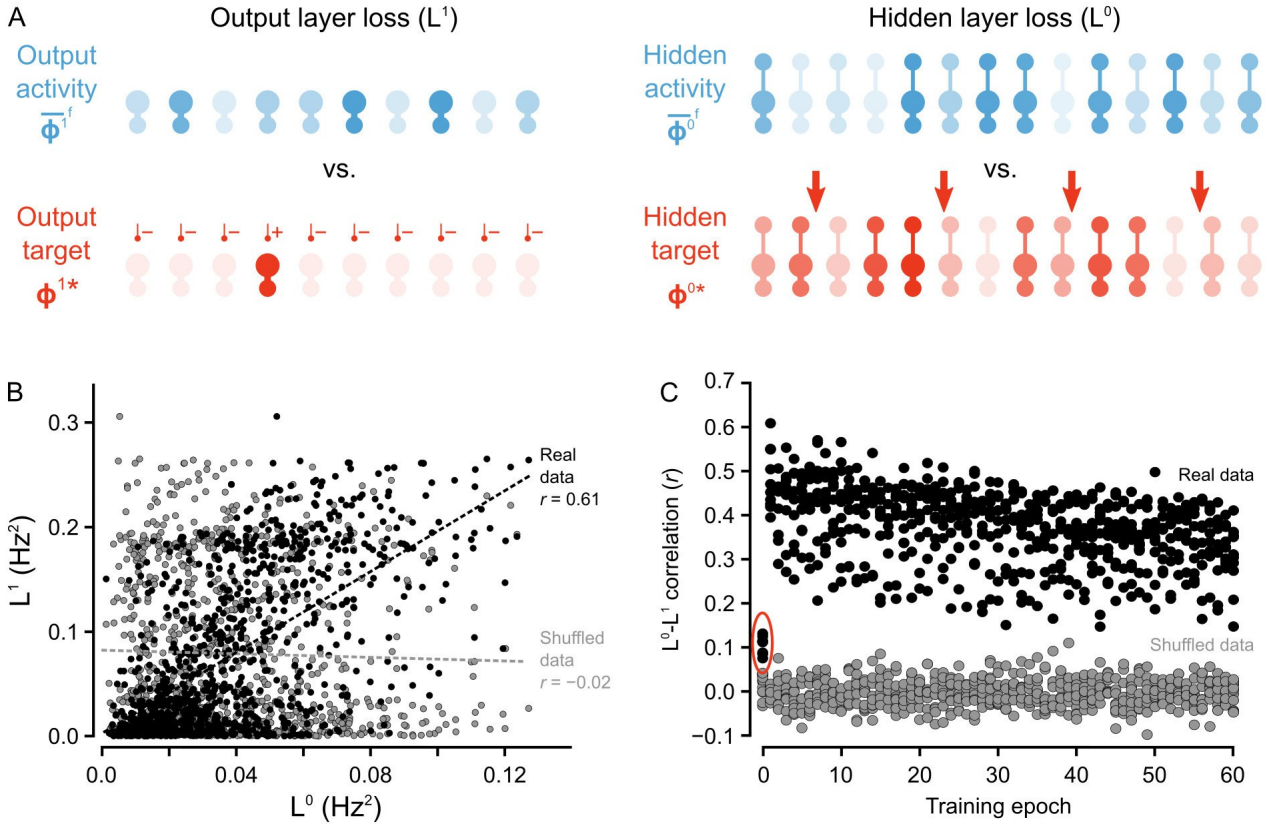
$$\phi_i^{1*} = \overline{\phi_i^1}^t = \frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \phi_i^1(t)dt$$

$$L^1 \approx ||\boldsymbol{\phi}^{1*} - \overline{\boldsymbol{\phi}^1}^t||_2^2 = ||\overline{\boldsymbol{\phi}^1}^t - \overline{\boldsymbol{\phi}^1}^f||_2^2 = \left|\left|\frac{1}{\Delta t_2}\int_{t_1+\Delta t_s}^{t_2} \boldsymbol{\phi}^1(t)dt - \frac{1}{\Delta t_1}\int_{t_0+\Delta t_s}^{t_1} \boldsymbol{\phi}^1(t)dt\right|\right|_2^2$$

Hidden layer target and loss

$$\phi_i^{0*} = \overline{\phi_i^0}^f + \alpha_i^t - \alpha_i^f$$

$$L^0 \approx ||\boldsymbol{\phi}^{0*} - \overline{\boldsymbol{\phi}^0}^f||_2^2 = ||\overline{\boldsymbol{\phi}^0}^f + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}_i^f - \overline{\boldsymbol{\phi}^0}^f||_2^2 = ||\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f||_2^2$$

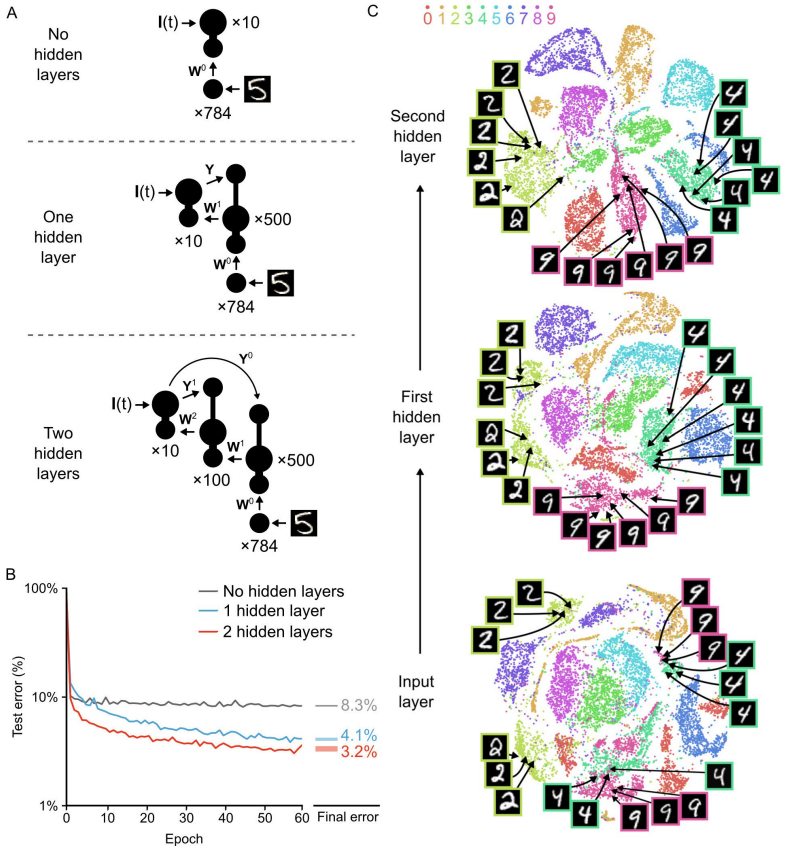


*Actual loss functions more complicated and shown in methods

Outline

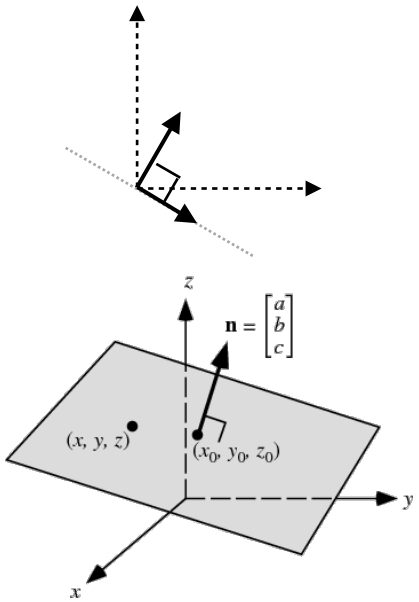
- Introduction: comparing the brain to a neural network
- The credit assignment problem in learning
- Primer on the machine learning solution: back-propagation
- Biology's solution? Deep learning with segregated dendrites
- **Results**
- Discussion

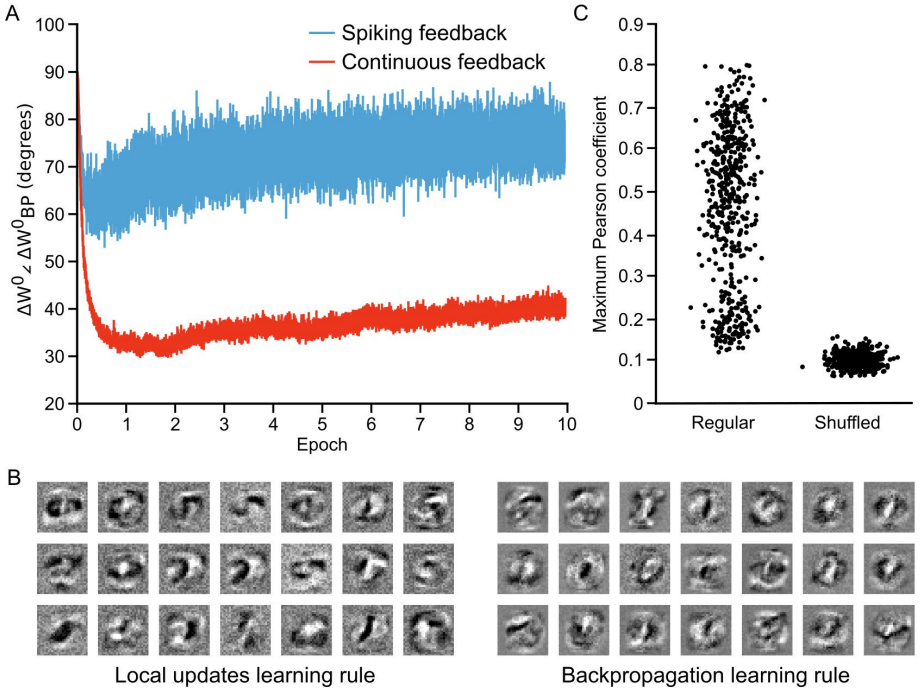
Deep learning: making good use of hidden layers



The algorithm approximates backpropagation in its weight updates

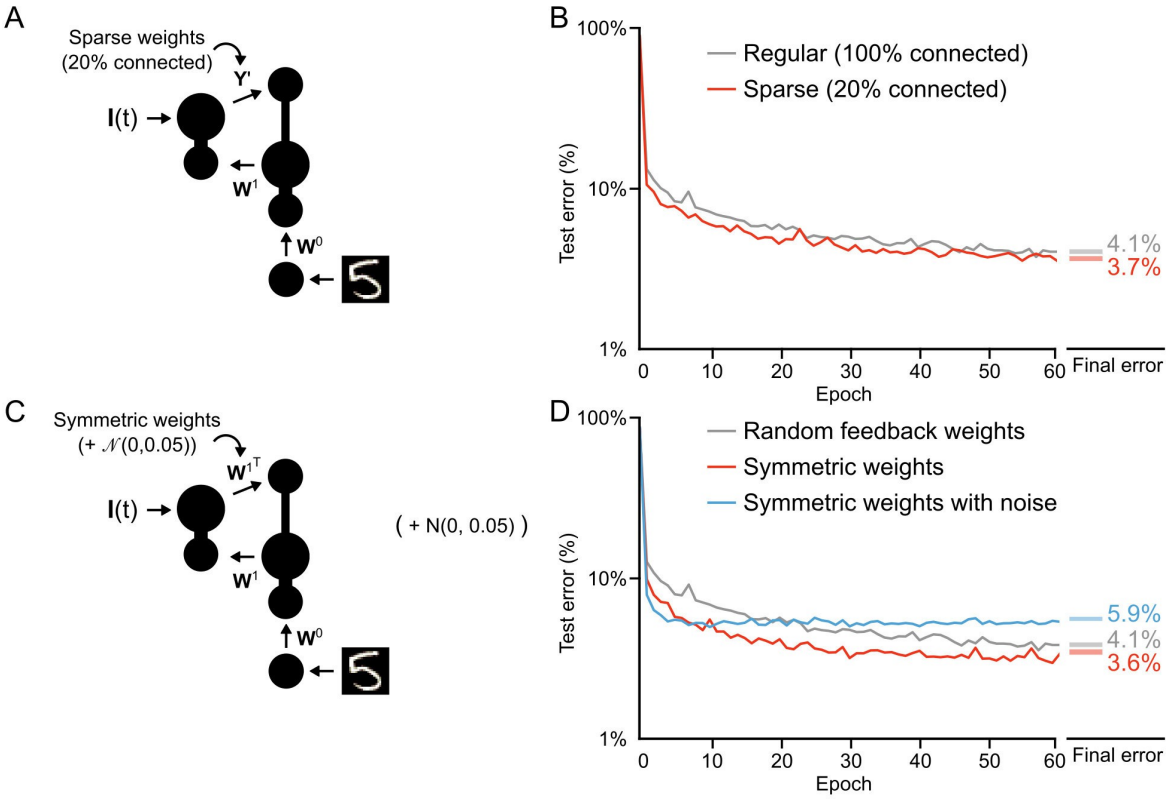
ΔW^0 vs ΔW^0_{BP}



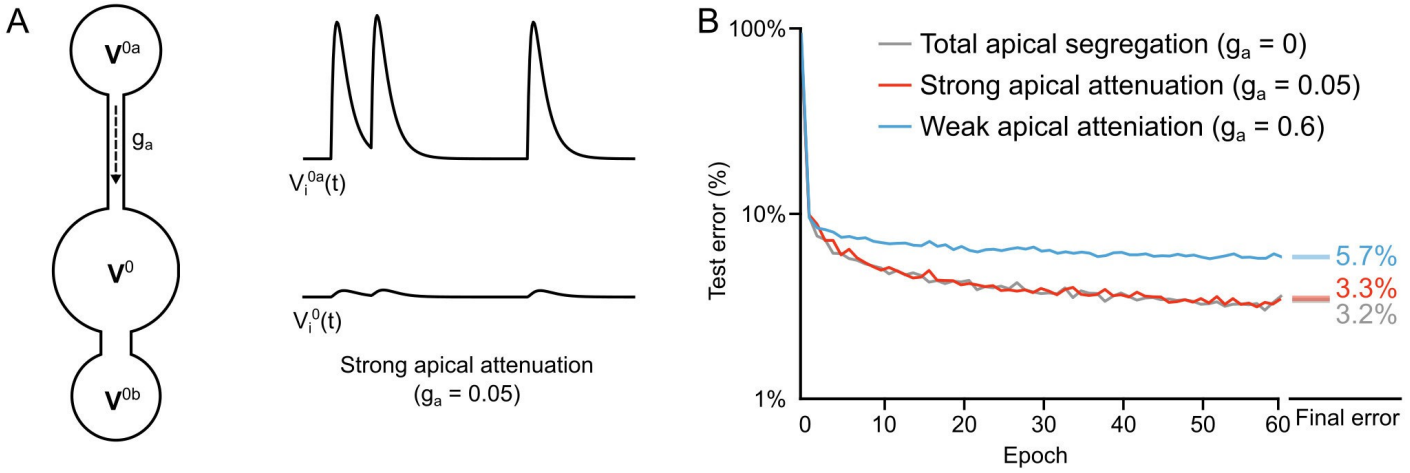


In higher dimensions, any random 2 vectors will be likely orthogonal

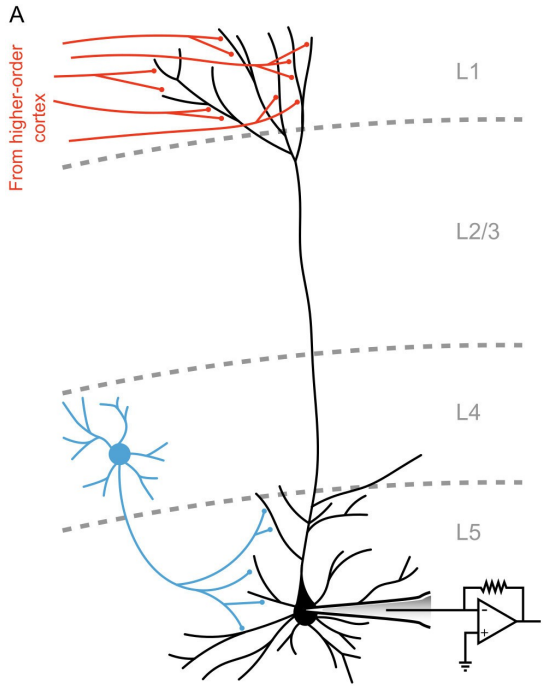
Conditions on feedback synapses

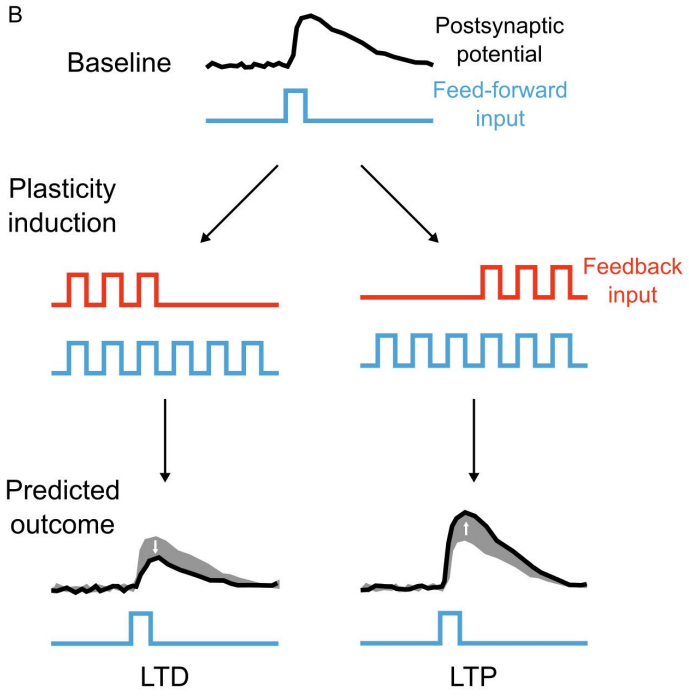


Apical dendritic segregation is important for learning



An experiment to test the central prediction of the model





Hidden layer target and loss

$$\phi_i^{0*} = \overline{\phi_i^{0^f}} + \alpha_i^t - \alpha_i^f$$

$$\begin{aligned} L^0 &\approx ||\boldsymbol{\phi}^{0*} - \overline{\boldsymbol{\phi}^{0^f}}||_2^2 \\ &= ||\overline{\boldsymbol{\phi}^{0^f}} + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}_i^f - \overline{\boldsymbol{\phi}^{0^f}}||_2^2 \\ &= ||\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f||_2^2 \end{aligned}$$

Plateau 1 higher than plateau 2: LTD

Plateau 2 higher than plateau 1: LTP

Summary

- Guergiev et al. solved the credit assignment problem with neurons with segregated dendrites
- Their algorithm approaches back-propagation in terms of the proposed weight updates
- Model of biologically realistic deep learning

Discussion

- Where does the teacher signal come from?
- Distinct phases of processing?
  - Might be governed by oscillations or neuromodulatory signals
- How do you overcome the temporal gap between plateau potentials?
  - Could be resolved by slower time constants

Thanks for listening!