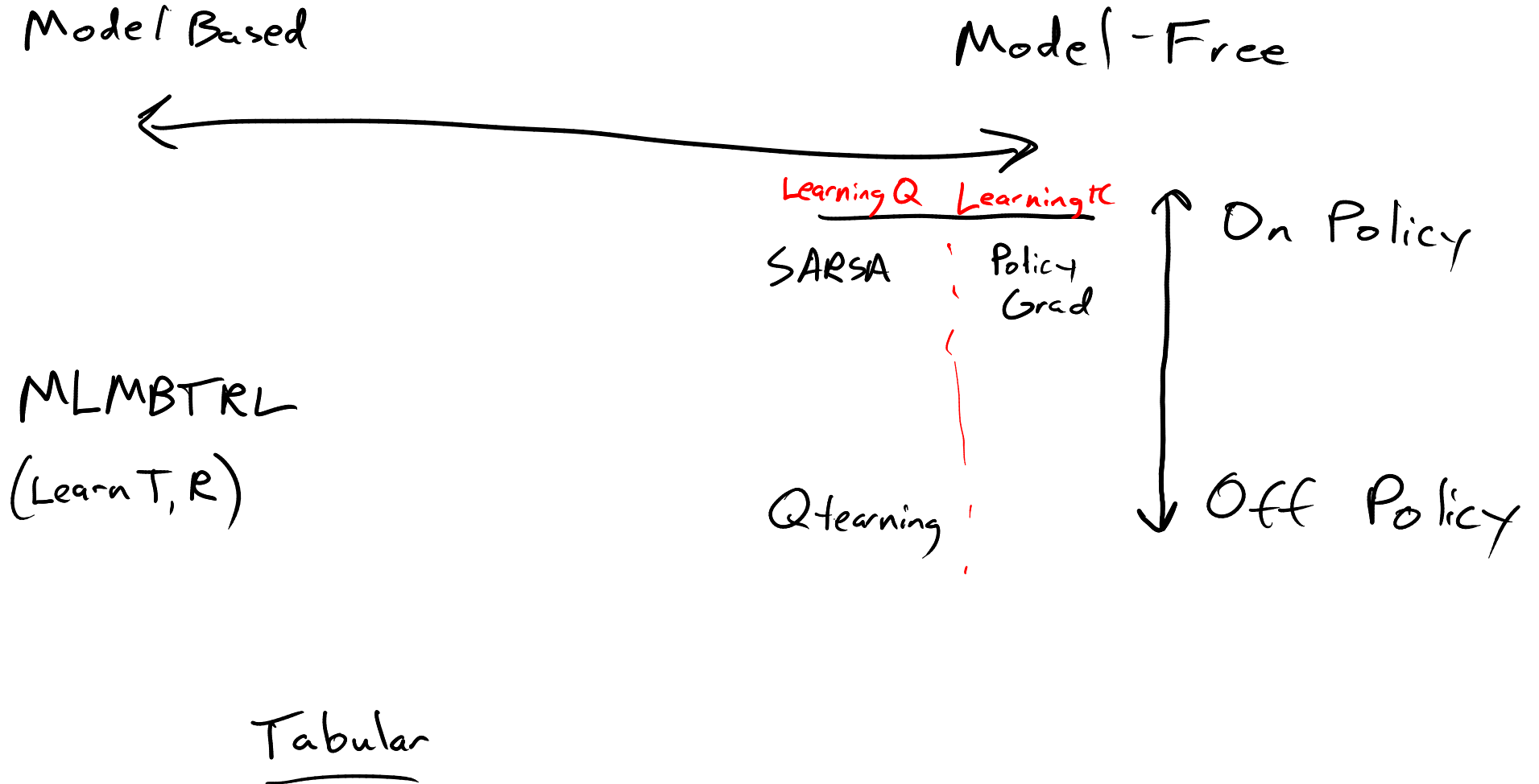


# Map of RL Algorithms



# This Time

Challenges in Reinforcement Learning:


- Exploration vs Exploitation
- Credit Assignment
- Generalization ←

# Function Approximation

# Function Approximation

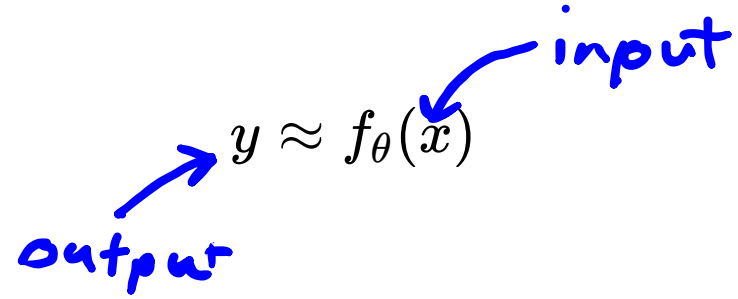
$$y \approx f_{\theta}(x)$$

# Function Approximation

$$y \approx f_{\theta}(x)$$


input

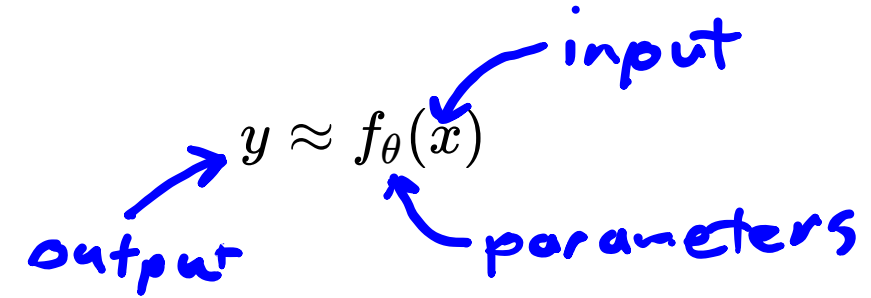
# Function Approximation



A diagram illustrating the function approximation equation  $y \approx f_{\theta}(x)$ . The equation is written in black. Two blue arrows point to the variables: one from the word "output" to  $y$ , and another from the word "input" to  $x$ . The words "output" and "input" are written in blue, lowercase, handwritten-style font.

$$y \approx f_{\theta}(x)$$

# Function Approximation



A diagram illustrating the function approximation equation  $y \approx f_{\theta}(x)$ . The equation is centered, with three handwritten blue arrows pointing to its components: one from the word "output" to  $y$ , one from the word "input" to  $x$ , and one from the word "parameters" to  $\theta$ .

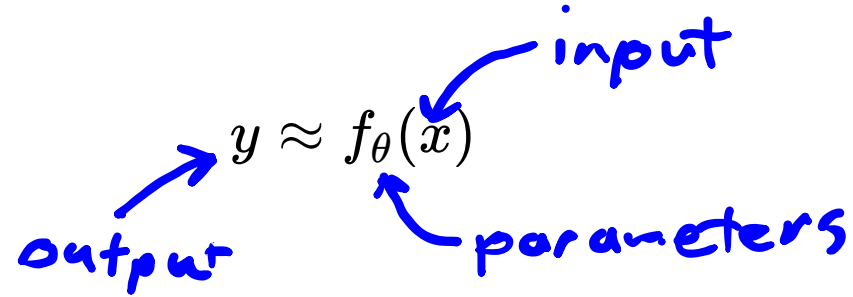
$$y \approx f_{\theta}(x)$$

output

input

parameters

# Function Approximation



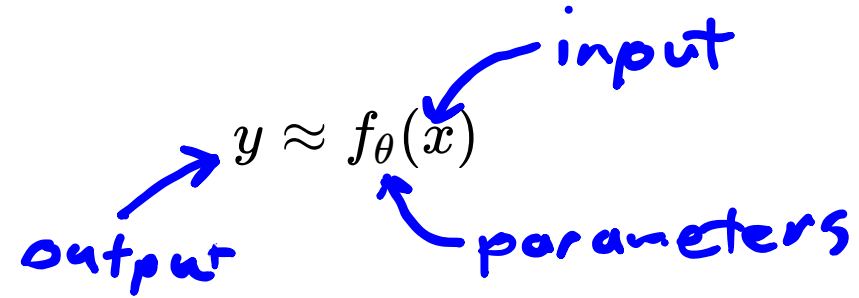
A diagram illustrating the function approximation equation  $y \approx f_{\theta}(x)$ . The equation is centered, with three handwritten blue annotations: an arrow pointing from the word "output" to  $y$ , an arrow pointing from the word "input" to  $x$ , and an arrow pointing from the word "parameters" to  $\theta$ .

Previously, Linear:

$$f_{\theta}(x) = \theta^{\top} \beta(x)$$

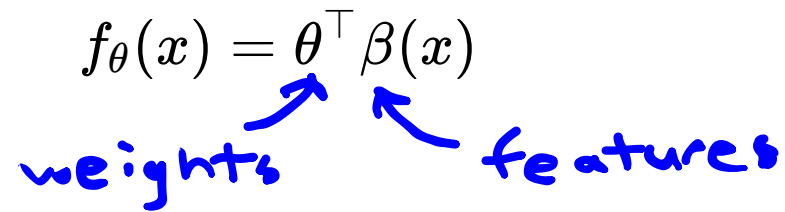


# Function Approximation

$$y \approx f_{\theta}(x)$$


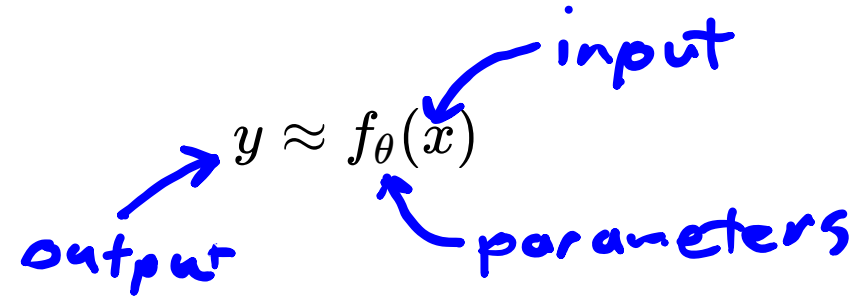
A diagram showing the equation  $y \approx f_{\theta}(x)$  with three handwritten blue arrows. One arrow points from the word "input" to the variable  $x$  in the function argument. Another arrow points from the word "output" to the variable  $y$ . A third arrow points from the word "parameters" to the symbol  $\theta$  in the subscript of the function.

Previously, Linear:

$$f_{\theta}(x) = \theta^{\top} \beta(x)$$


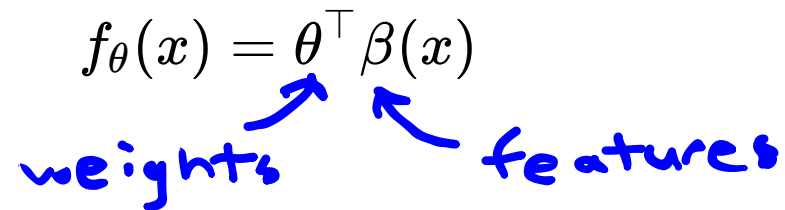
A diagram showing the equation  $f_{\theta}(x) = \theta^{\top} \beta(x)$  with two handwritten blue arrows. One arrow points from the word "weights" to the symbol  $\theta$ . The other arrow points from the word "features" to the symbol  $\beta(x)$ .

# Function Approximation

$$y \approx f_{\theta}(x)$$


A diagram showing the equation  $y \approx f_{\theta}(x)$  with three handwritten blue arrows. One arrow points from the word "input" to the variable  $x$  in the function argument. Another arrow points from the word "output" to the variable  $y$ . A third arrow points from the word "parameters" to the symbol  $\theta$  in the subscript of the function.

Previously, Linear:

$$f_{\theta}(x) = \theta^{\top} \beta(x)$$


A diagram showing the equation  $f_{\theta}(x) = \theta^{\top} \beta(x)$  with two handwritten blue arrows. One arrow points from the word "weights" to the symbol  $\theta$ . The other arrow points from the word "features" to the symbol  $\beta(x)$ .

e.g.  $\beta_i(x) = \sin(i \pi x)$

~~AI = Neural Nets~~

$\pi_{\theta}$


$Q_{\theta}$

gathering data

~~Neural Nets are  
just another  
function  
approximator~~

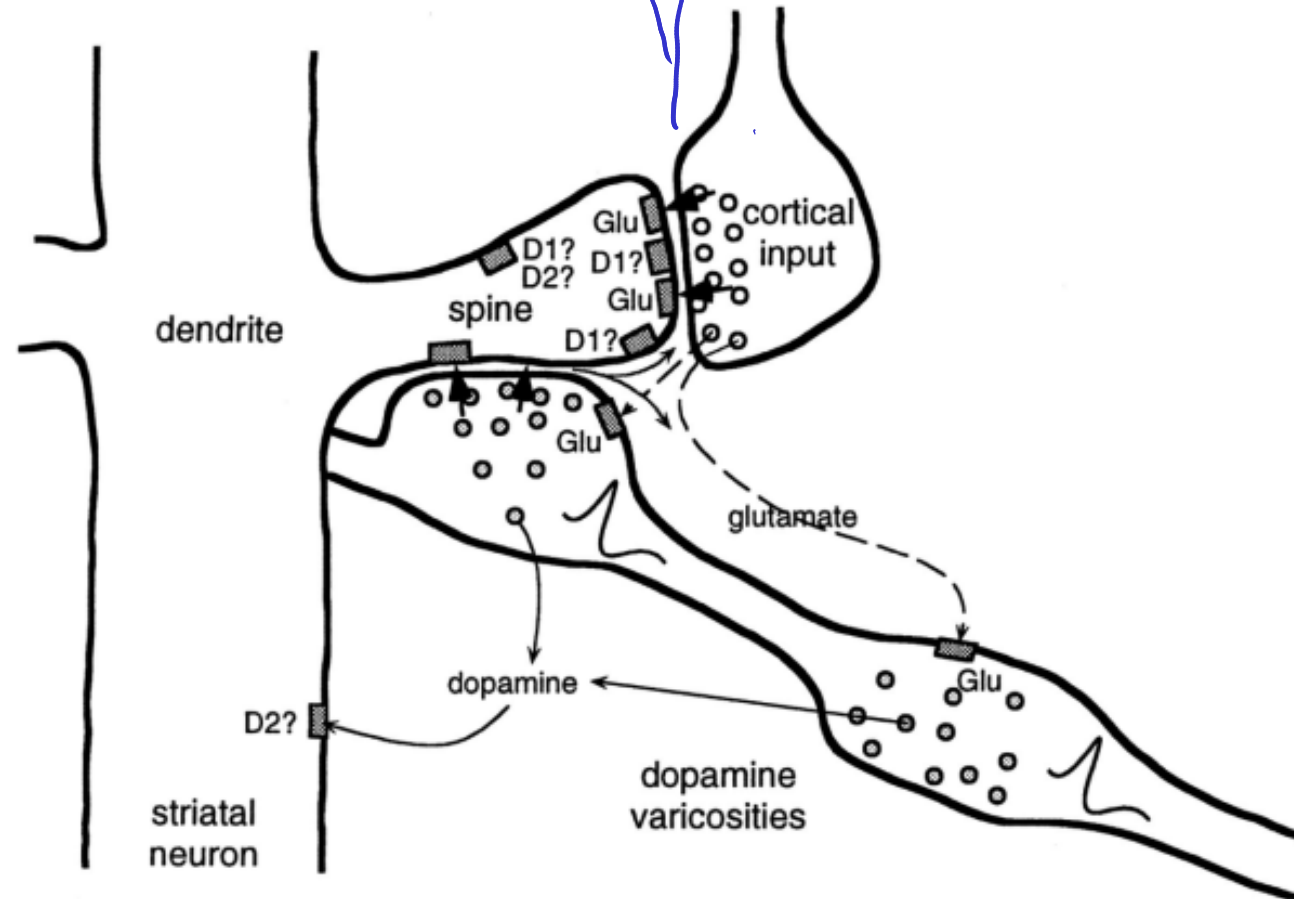
# Neural Network

# Neural Network

$$h(x) = \sigma(Wx + b)$$


# Neural Network

$$h(x) = \sigma(Wx + b)$$



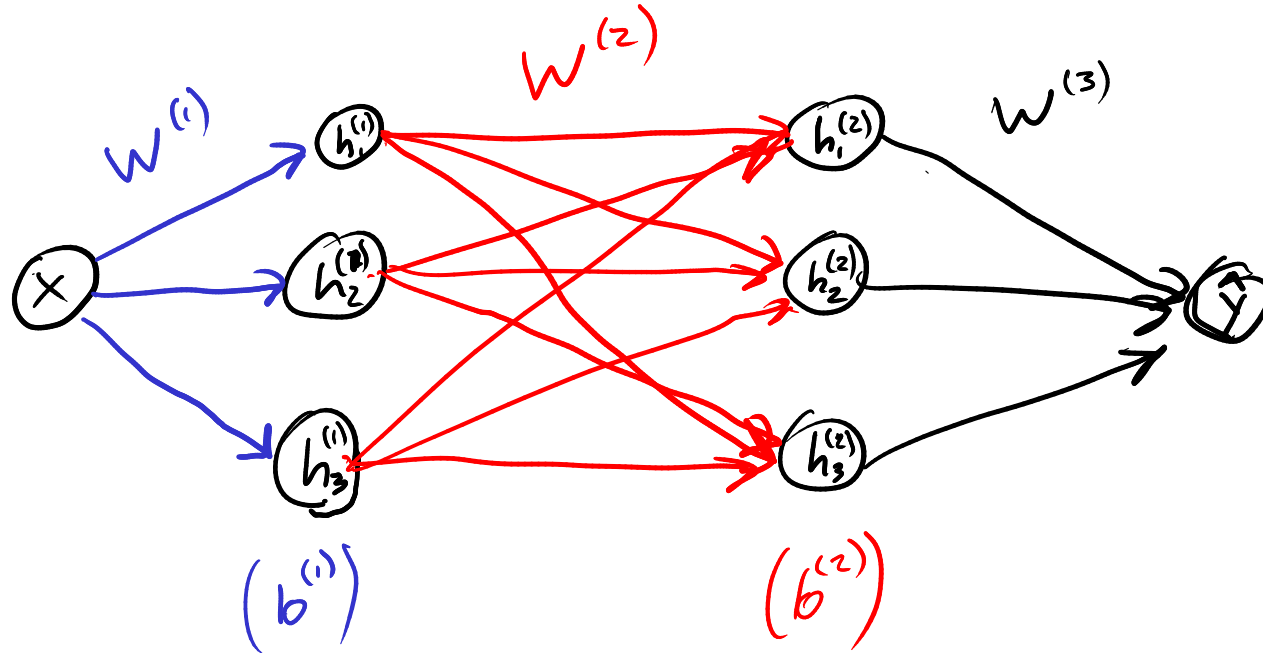
# Neural Network

# Neural Network

$$h(x) = \sigma(Wx + b)$$

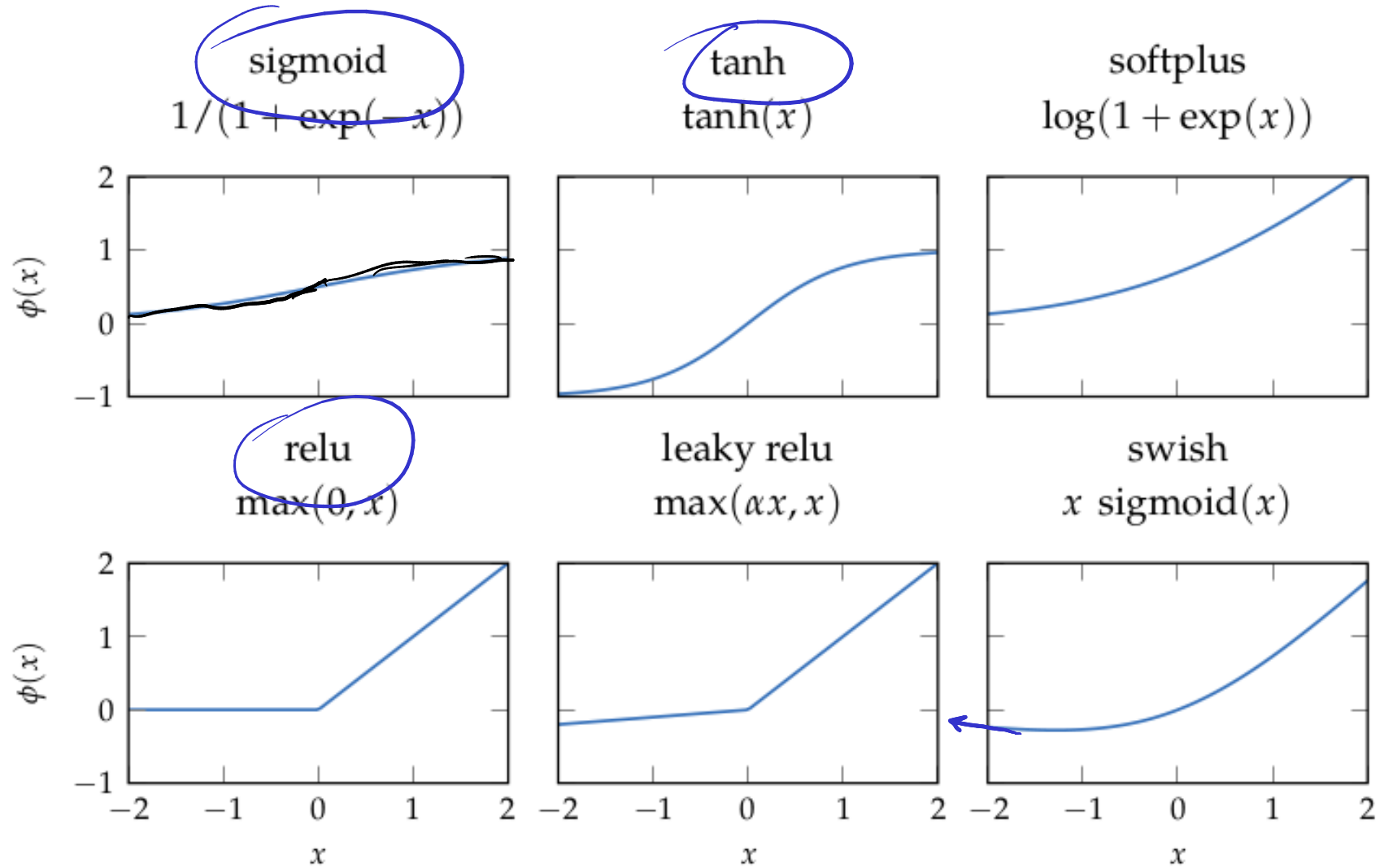
$$f_{\theta}(x) = h^{(2)}(h^{(1)}(x)) = \sigma^{(2)}(W^{(2)} \sigma^{(1)}(W^{(1)}x + b^{(1)}) + b^{(2)})$$

$$\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$$



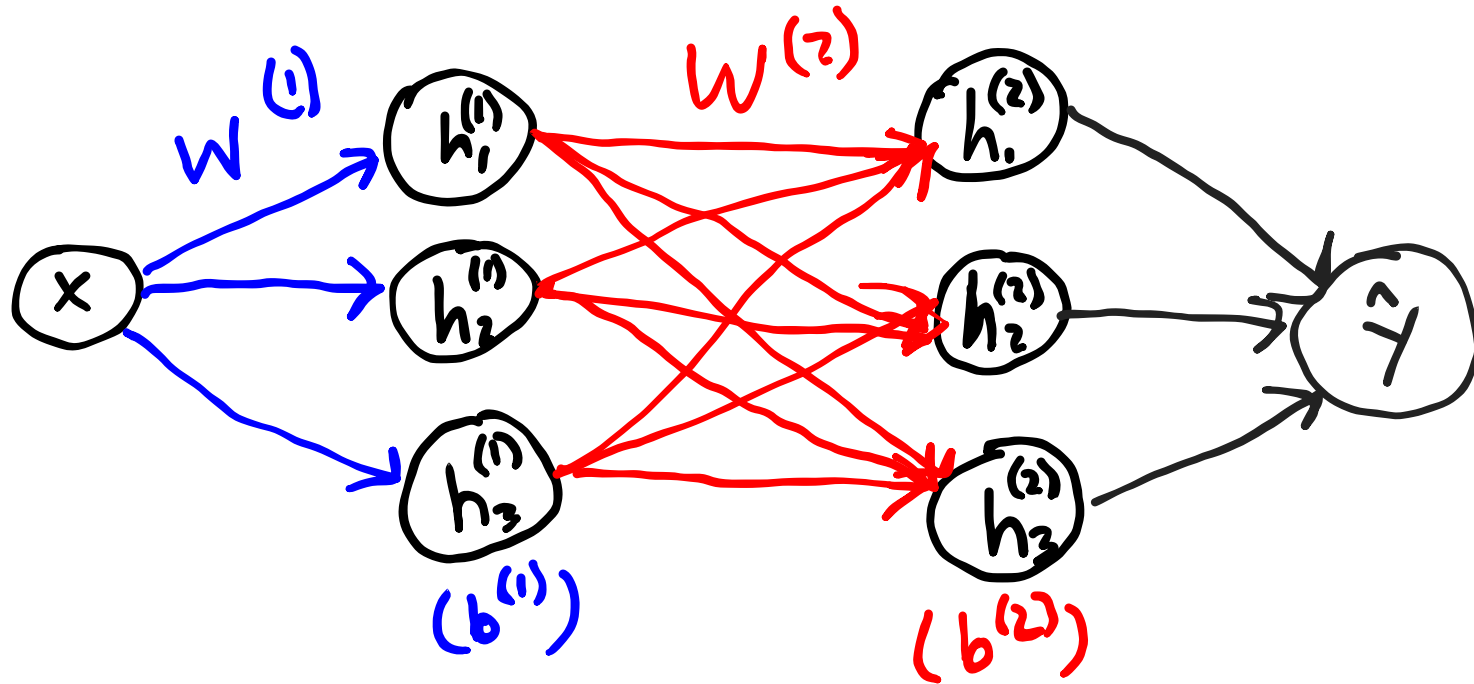


# Nonlinearities

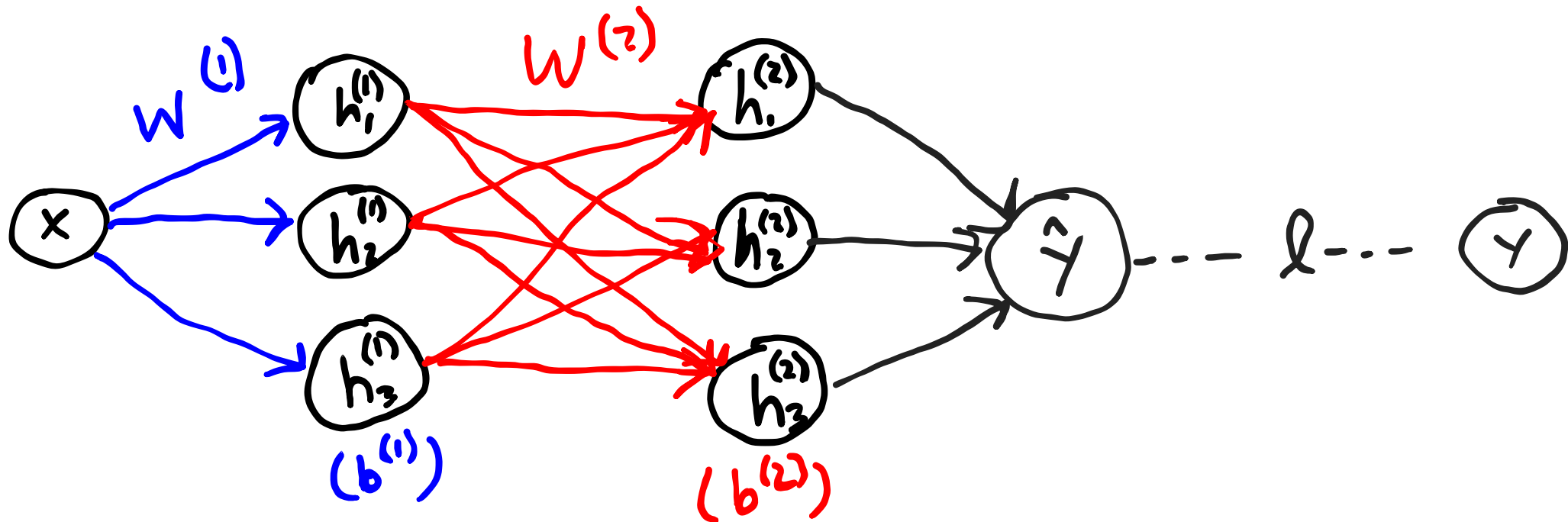


# Training

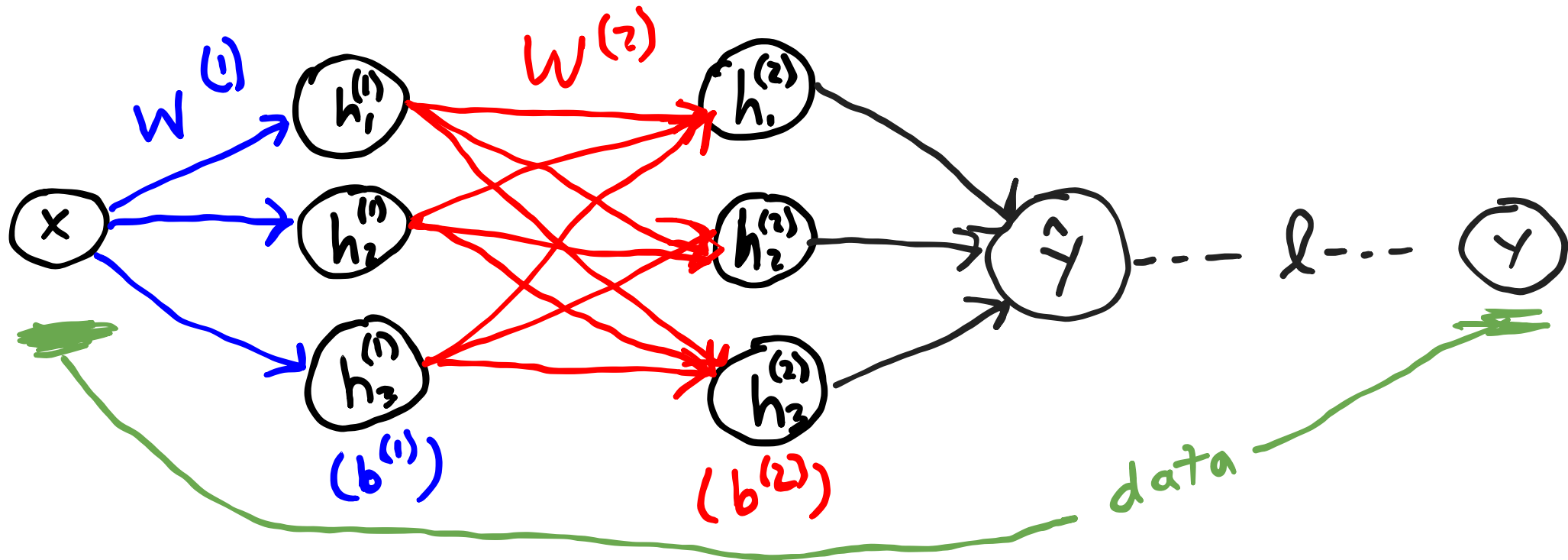
# Training



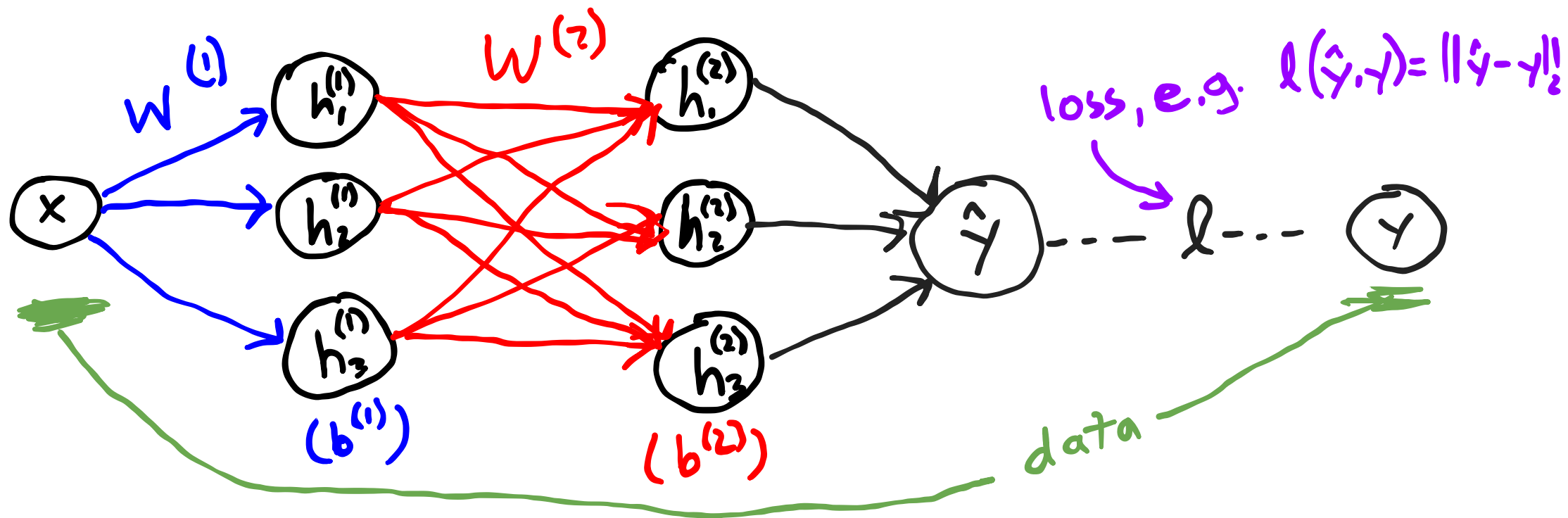
# Training



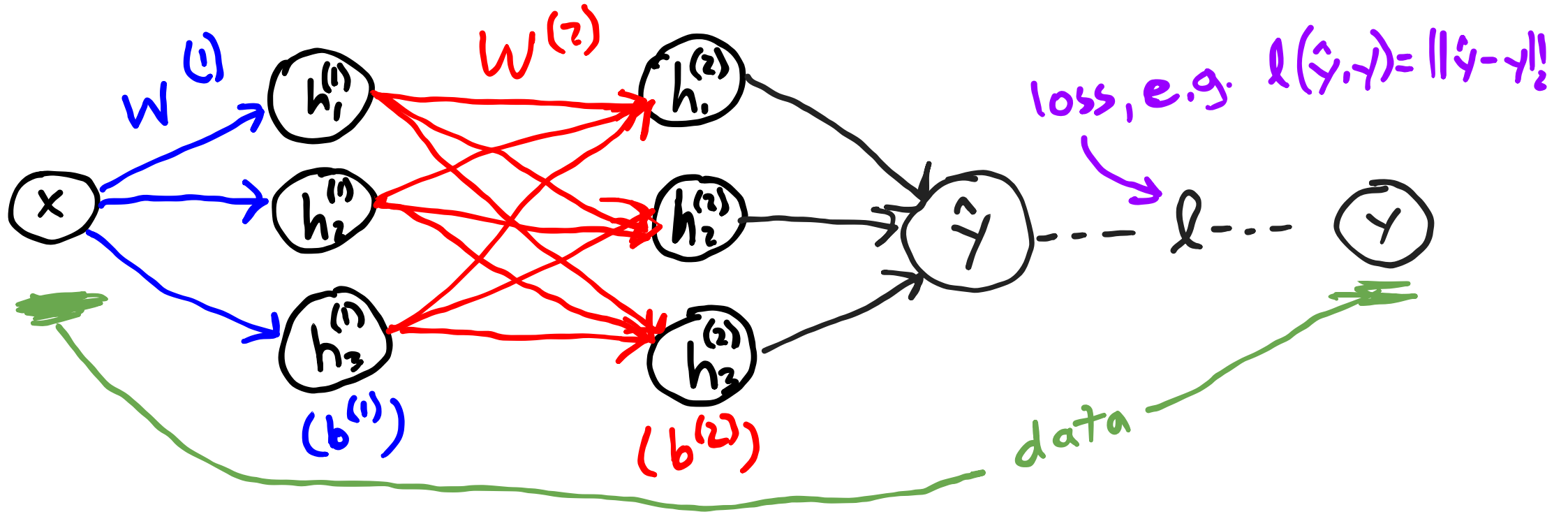
# Training



# Training

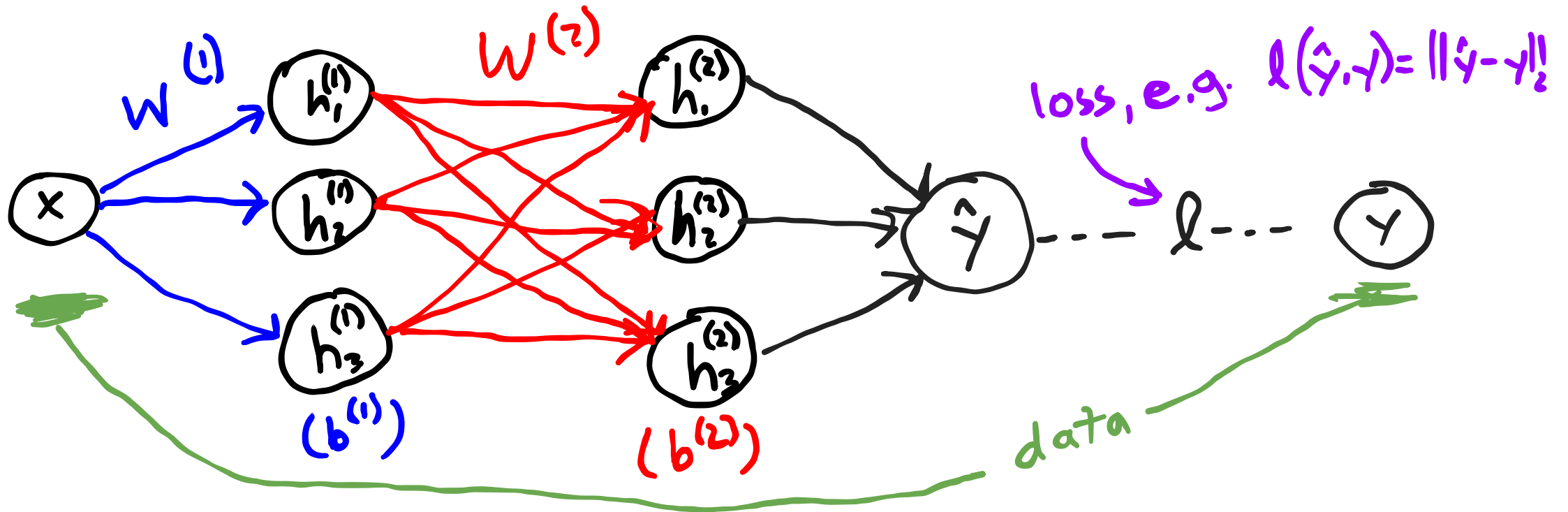


# Training



$$\theta^* = \arg \min_{\theta} \sum_{(x,y) \in \mathcal{D}} l(f_{\theta}(x), y)$$

# Training



$$\theta^* = \arg \min_{\theta} \sum_{(x,y) \in \mathcal{D}} l(f_{\theta}(x), y)$$

Stochastic Gradient Descent:  $\theta \leftarrow \theta - \alpha \nabla_{\theta} l(f_{\theta}(x), y)$



$f \cdot g \cdot h$

$f(g(h(x)))$

# Chain Rule

$$\left. \frac{\partial f(g(h(x)))}{\partial x} \right|_x = \left. \frac{\partial f(g(h))}{\partial h} \right|_h \left. \frac{\partial h(x)}{\partial x} \right|_x = \underbrace{\left. \frac{\partial f(g)}{\partial g} \right|_g}_{\text{red}} \underbrace{\left. \frac{\partial g(h)}{\partial h} \right|_h}_{\text{red}} \underbrace{\left. \frac{\partial h(x)}{\partial x} \right|_x}_{\text{red}}$$

$$l(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$\frac{\partial l}{\partial \hat{y}_i} = -\frac{2}{n} (y_i - \hat{y}_i)$$

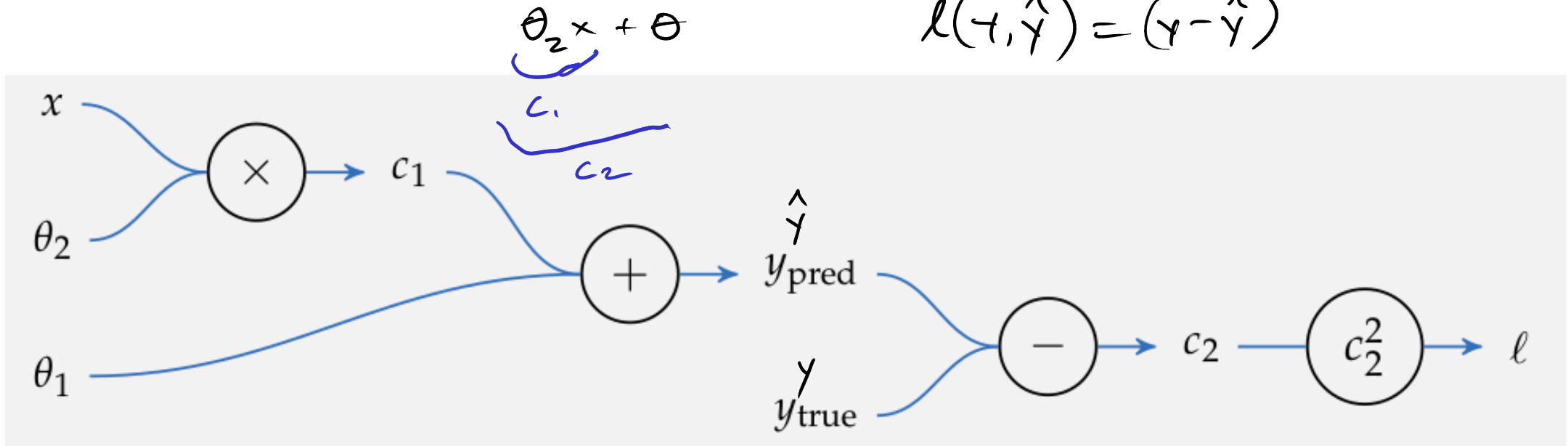
$$\hat{y} = \underline{W^{(2)}} \sigma(\underline{W^{(1)}} x + \underline{b^{(1)}}) + \underline{b^{(2)}}$$
$$\frac{\partial l}{\partial W^{(2)}} = \frac{\partial l}{\partial \hat{y}} \left( \frac{\partial \hat{y}}{\partial W^{(2)}} \right) = \frac{\partial l}{\partial \hat{y}} \sigma(W^{(1)} x + b^{(1)})^T$$

$$\underline{W^{(2)}} \leftarrow \underline{W^{(2)}} - \alpha \frac{\partial l}{\partial W^{(2)}}$$

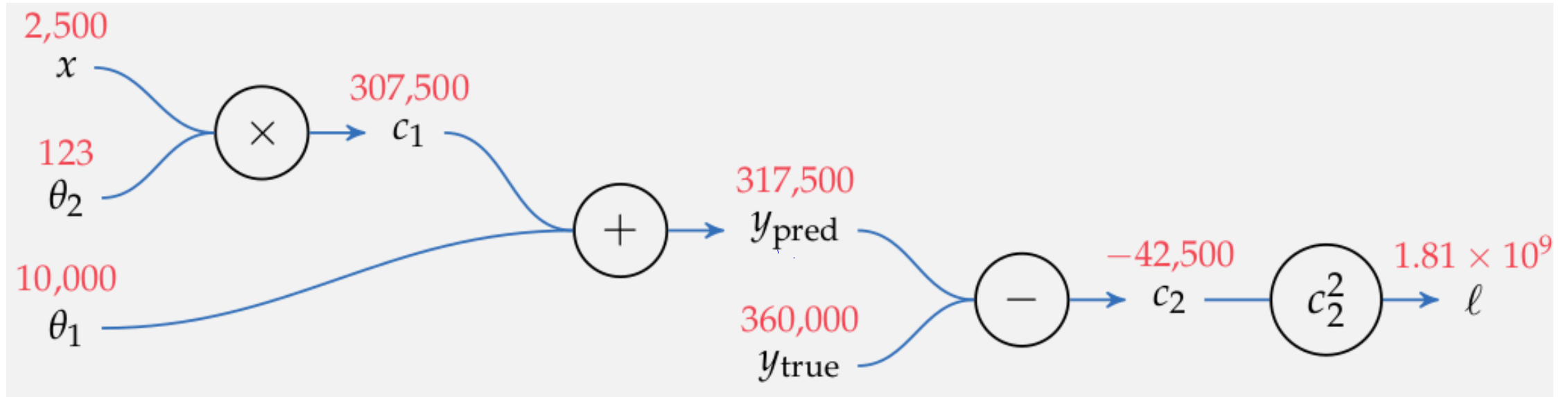
# Backprop

# Backprop

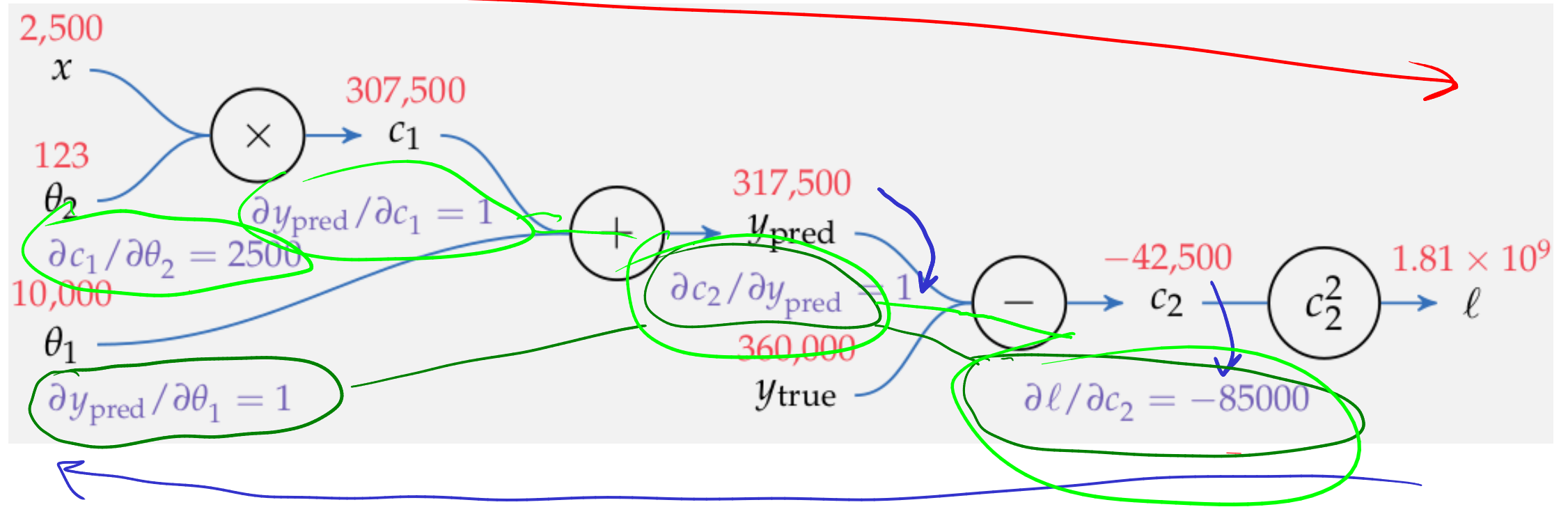
$$\ell(y, \hat{y}) = (y - \hat{y})^2$$



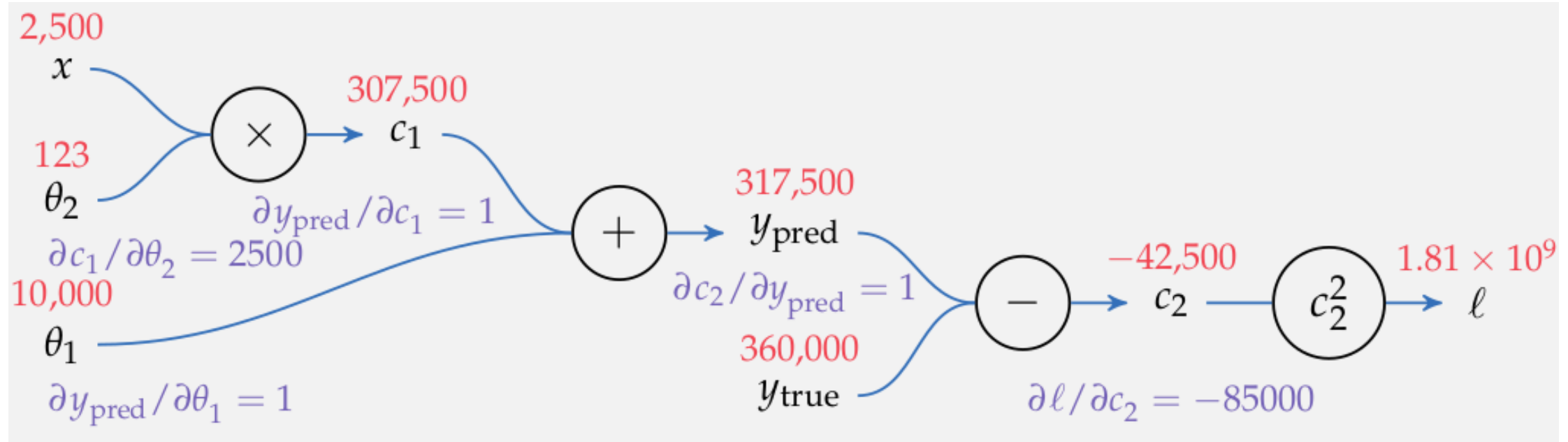
# Backprop



# Backprop



# Backprop



$$\frac{\partial \ell}{\partial \theta_1} = \frac{\partial \ell}{\partial c_2} \frac{\partial c_2}{\partial y_{\text{pred}}} \frac{\partial y_{\text{pred}}}{\partial \theta_1} = -85,000 \cdot 1 \cdot 1 = -85,000$$

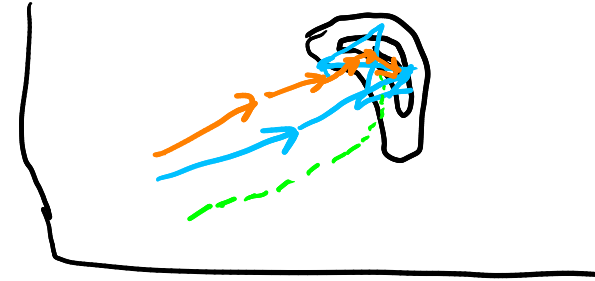
$$\frac{\partial \ell}{\partial \theta_2} = \frac{\partial \ell}{\partial c_2} \frac{\partial c_2}{\partial y_{\text{pred}}} \frac{\partial y_{\text{pred}}}{\partial c_1} \frac{\partial c_1}{\partial \theta_2} = -85,000 \cdot 1 \cdot 1 \cdot 2,500 = -2.125 \times 10^8$$

a “fast and furious” approach to training neural networks does not work and only leads to suffering. Now, suffering is a perfectly natural part of getting a neural network to work well, but it can be mitigated by being thorough, defensive, paranoid, and obsessed with visualizations of basically every possible thing. The qualities that in my experience correlate most strongly to success in deep learning are patience and attention to detail.

- Andrej Karpathy

# Adaptive Step Size: RMSProp

$$\theta = \theta + \alpha \widehat{\nabla_{\theta} f_{\theta}(x)}$$



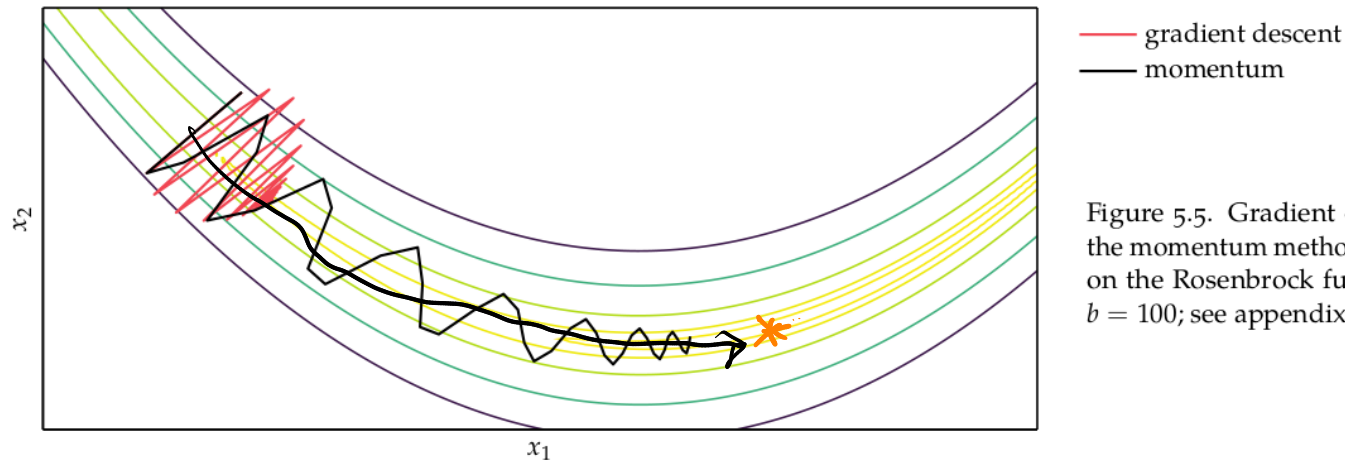
$$\hat{s}^{(k+1)} = \gamma \hat{s}^{(k)} + (1-\gamma)(g^{(k)} \odot g^{(k)})$$

↖ elementwise product

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\alpha}{\epsilon + \sqrt{s_i^{(k+1)}}} g_i^{(k)}$$



# Adaptive Step Size: ADAM



biased decaying moment

biased decaying sq. gradient

corrected d. m.

corrected sq. gradient

$$\hat{v}^{(k+1)} = \gamma_v v^{(k)} + (1 - \gamma_v) g^{(k)}$$

$$\hat{s}^{(k+1)} = \gamma_s s^{(k)} + (1 - \gamma_s) (g^{(k)} \odot g^{(k)})$$

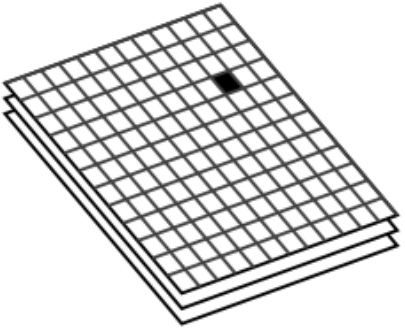
$$\hat{v}^{(k+1)} = v^{(k+1)} / (1 - \gamma_v^{k+1})$$

$$\hat{s}^{(k+1)} = s^{(k+1)} / (1 - \gamma_s^{k+1})$$

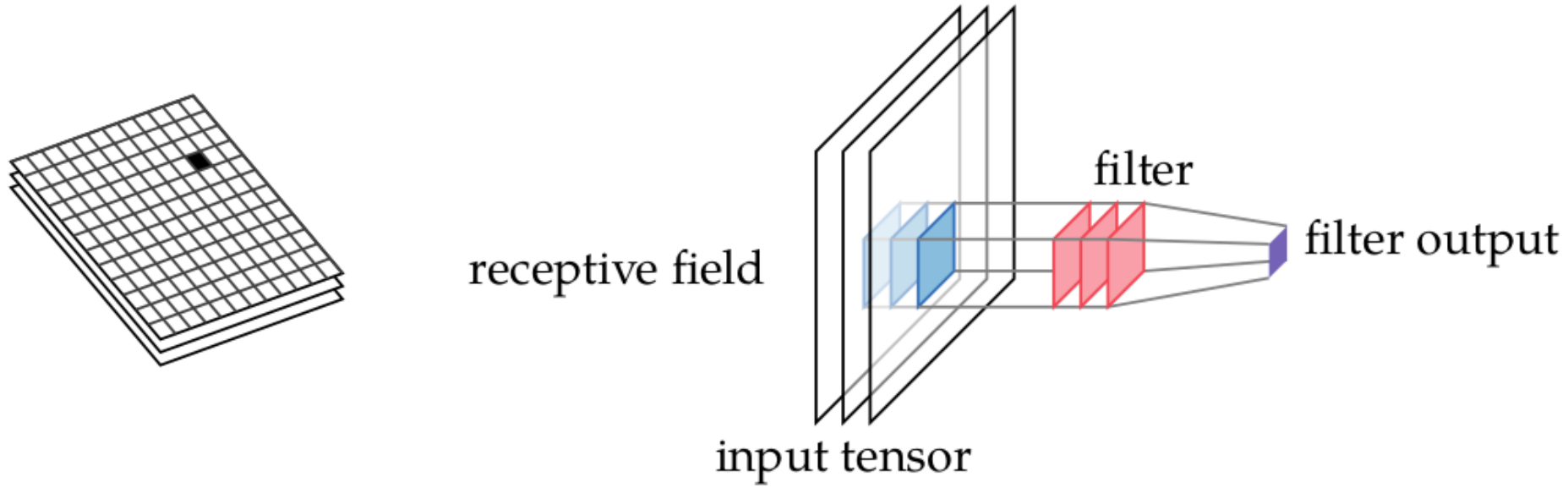
$$x^{(k+1)} = x^{(k)} + \alpha \hat{v}^{(k+1)} / (\epsilon + \sqrt{\hat{s}^{(k+1)}})$$

# On Your Radar: ConvNets

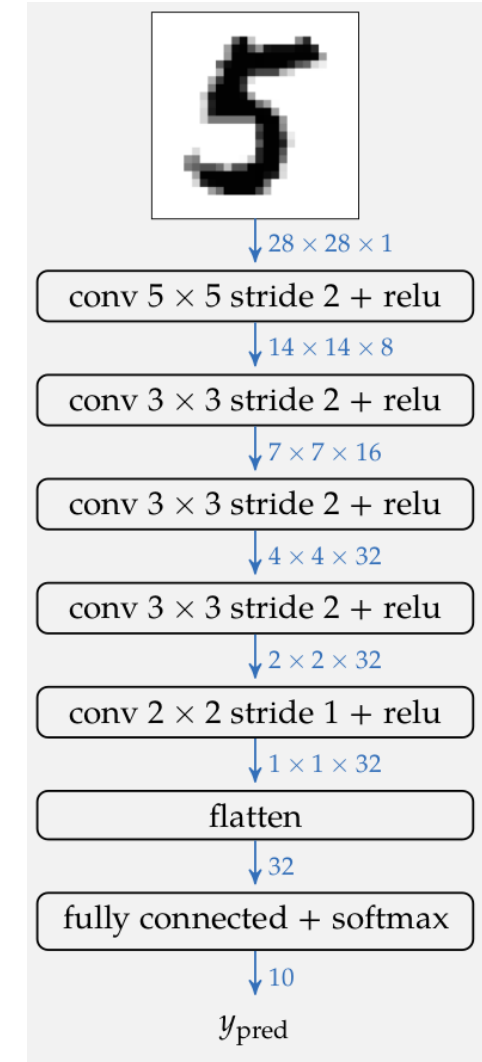
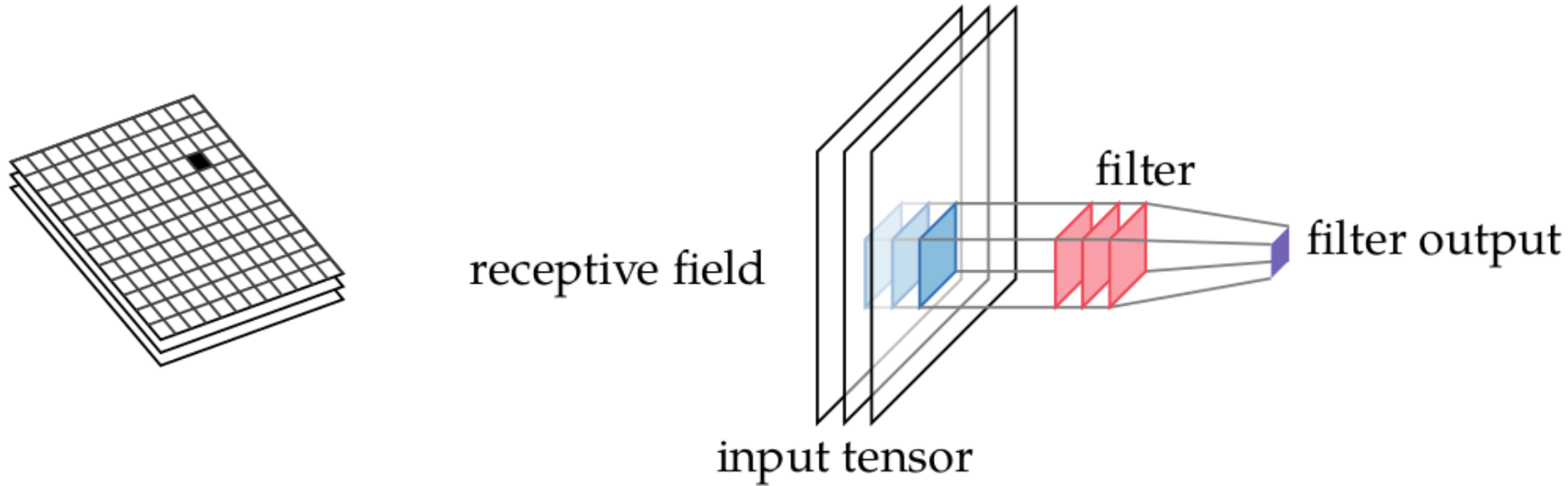
# On Your Radar: ConvNets



# On Your Radar: ConvNets



# On Your Radar: ConvNets



# On Your Radar: Regularization

# On Your Radar: Regularization

$$\arg \min_{\boldsymbol{\theta}} \sum_{(x,y) \in \mathbf{D}} \ell(f_{\boldsymbol{\theta}}(x), y) - \beta \|\boldsymbol{\theta}\|^2$$

# On Your Radar: Regularization

$$\arg \min_{\boldsymbol{\theta}} \sum_{(x,y) \in \mathbf{D}} \ell(f_{\boldsymbol{\theta}}(x), y) - \beta \|\boldsymbol{\theta}\|^2$$

e.g. Batch norm, dropout