ROB 537:
Learning-Based Control

Week 1, Lecture 2

Multi-Layered Feed Forward Neural Networks

Announcements:

HW 1 Due on 10/5

(write your own Neural Network)

ROB 537
Learning-Based Control

Week 1, Lecture 2

Multi-Layered Feed Forward Neural Networks

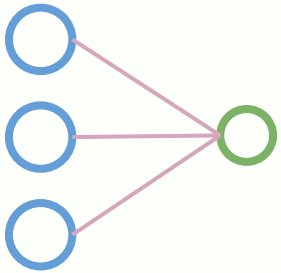
HW 1 due on **10/10 11:59 PM**

Robotics Seminars: 10AM Fridays

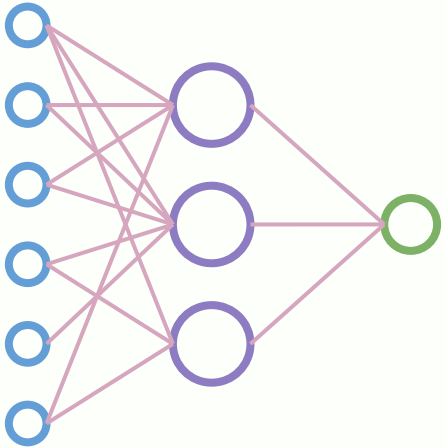
Oregon State University

Neural Network Architectures

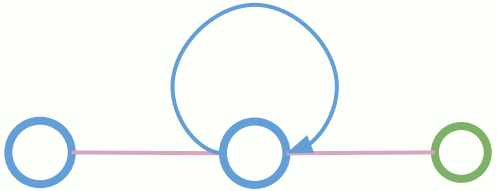
Single Layer Feed Forward



Multi-Layered Feed Forward



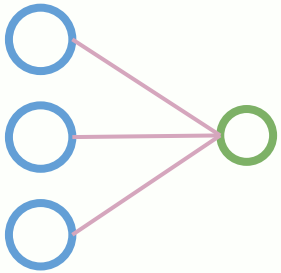
Recurrent Network



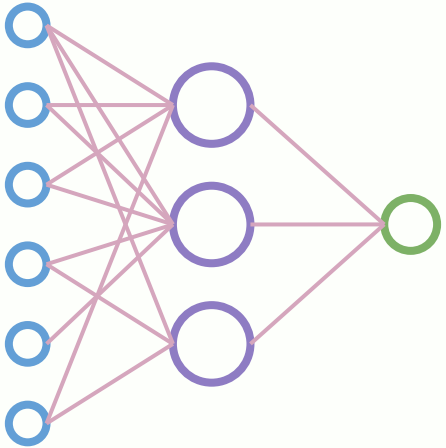
Oregon State University

Neural Network Architectures

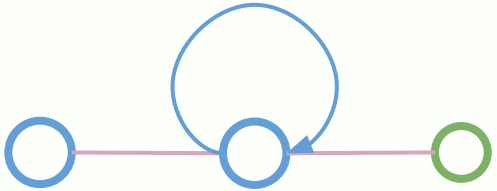
Single Layer Feed Forward



Multi-Layered Feed Forward

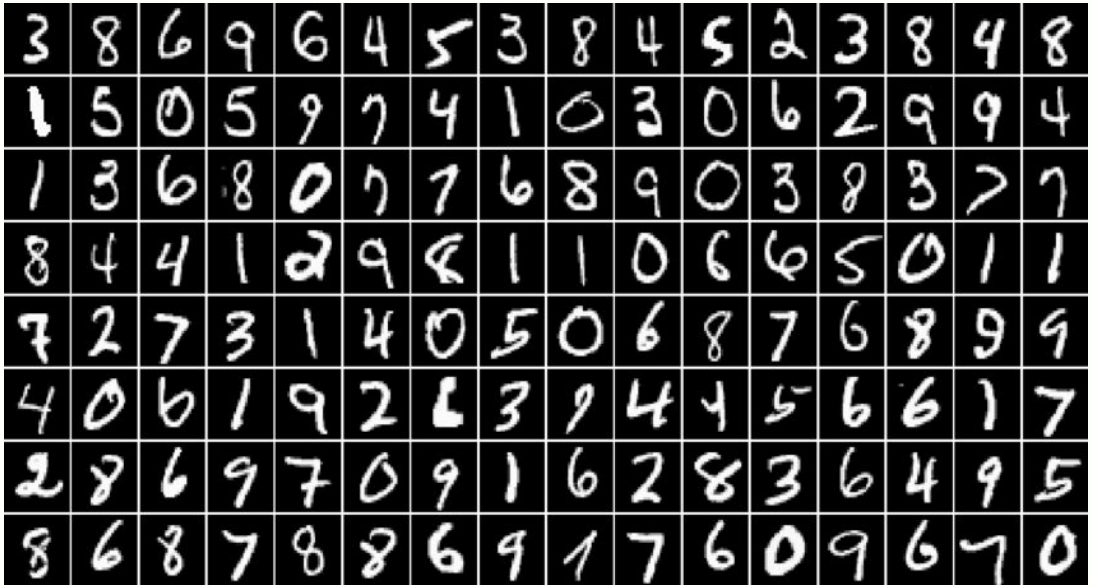


Recurrent Network

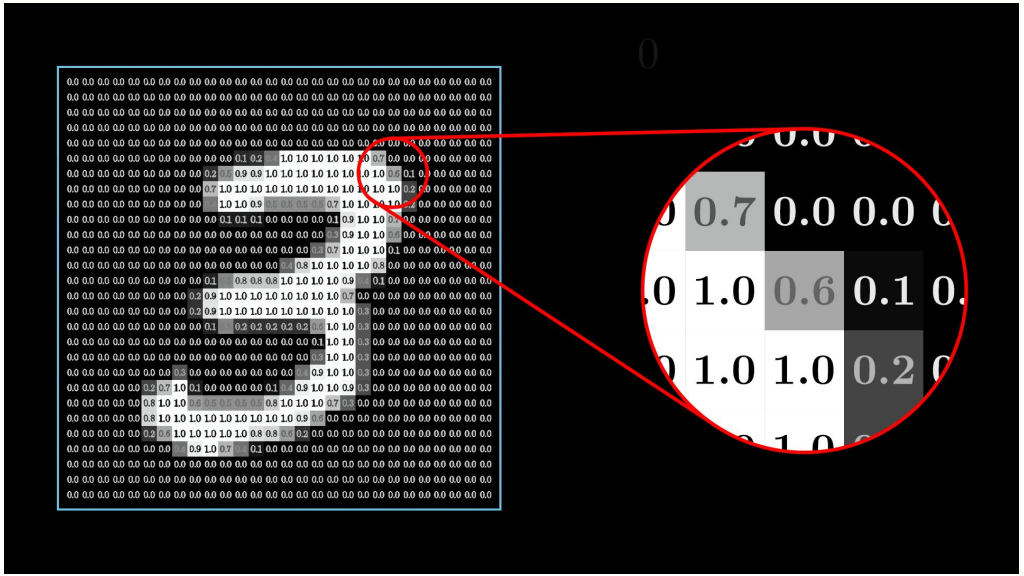


Oregon State University

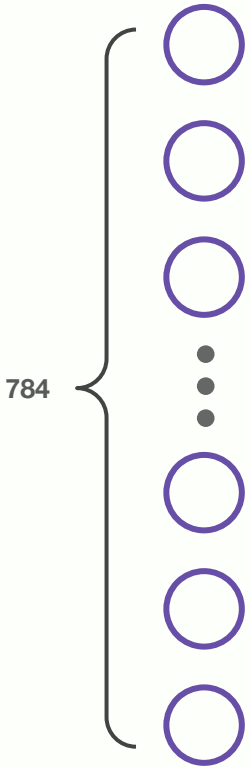
Learning From Data → MNIST



Learning From Data → MNIST

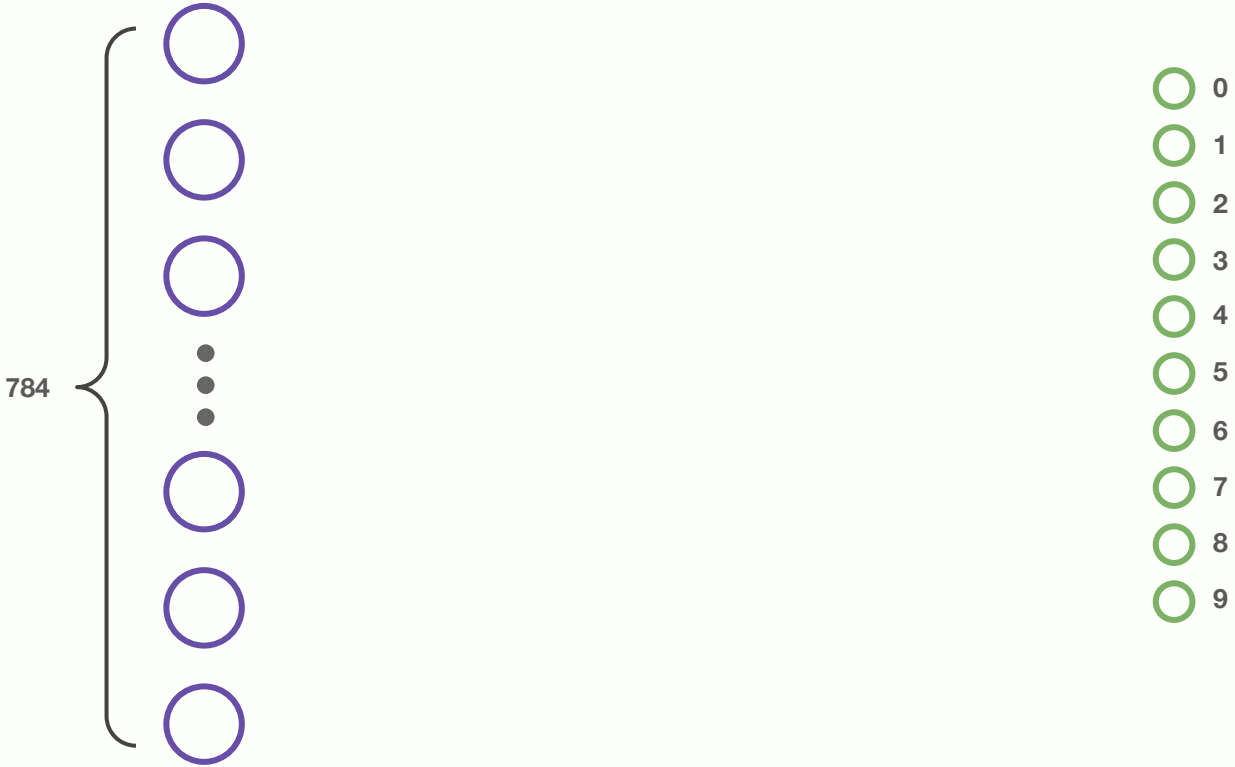


MNIST → Neural Network



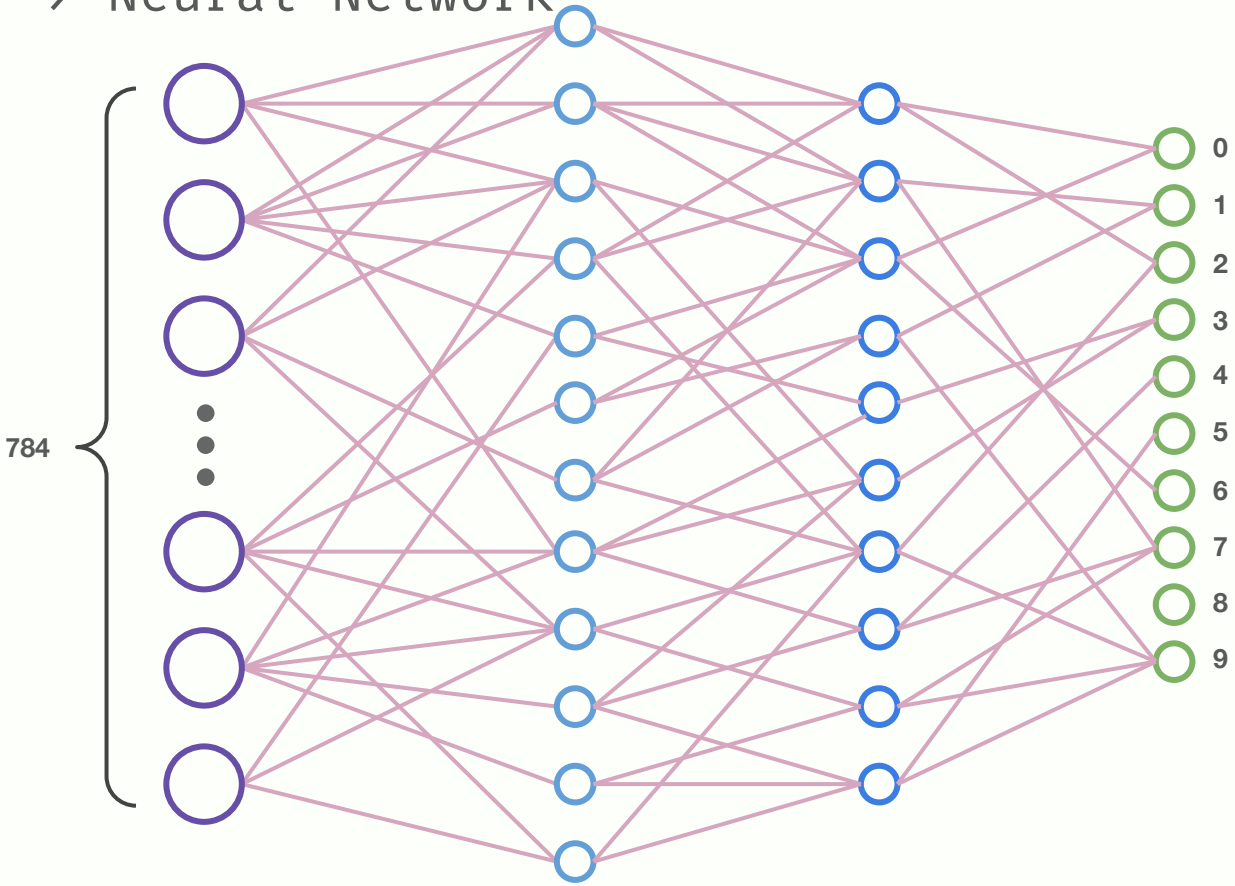
Oregon State University

MNIST → Neural Network



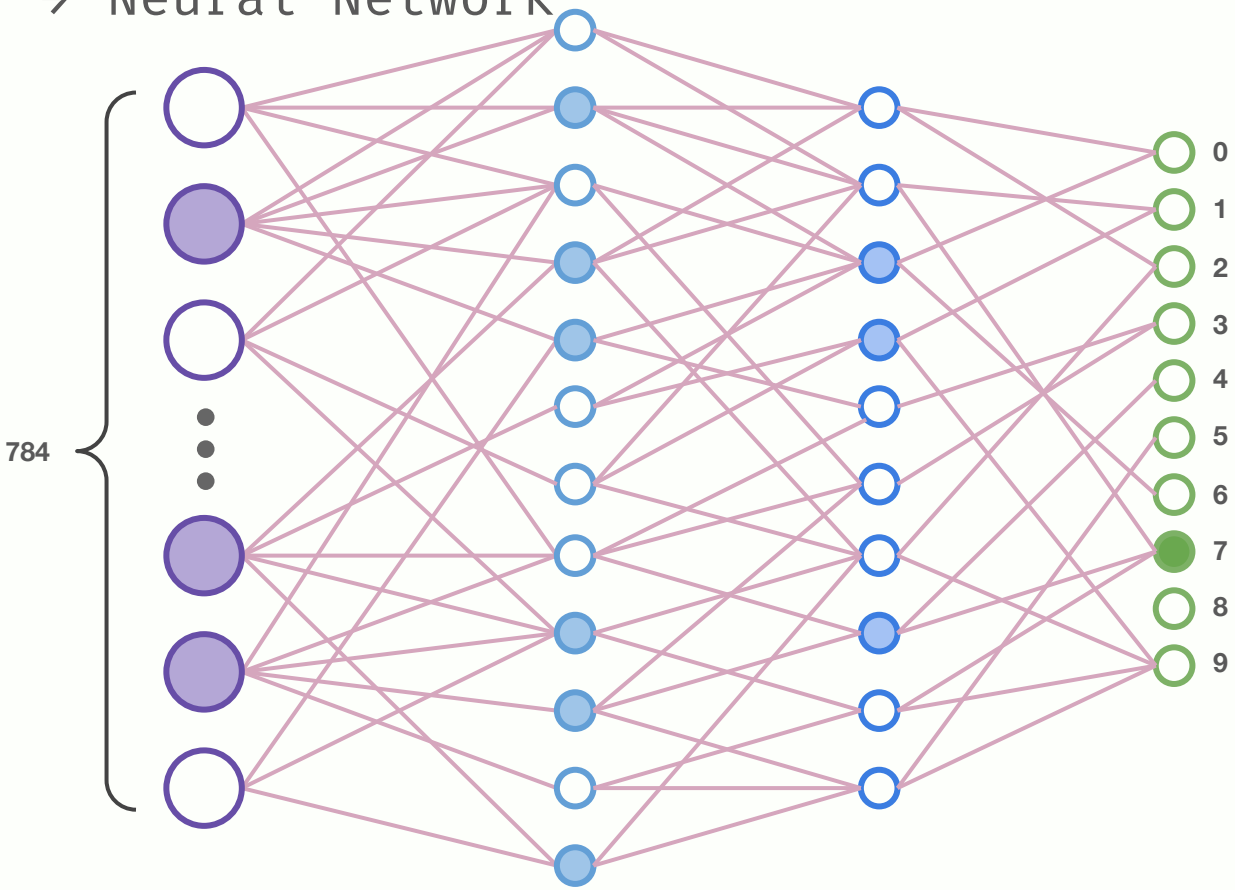
Oregon State University

MNIST → Neural Network



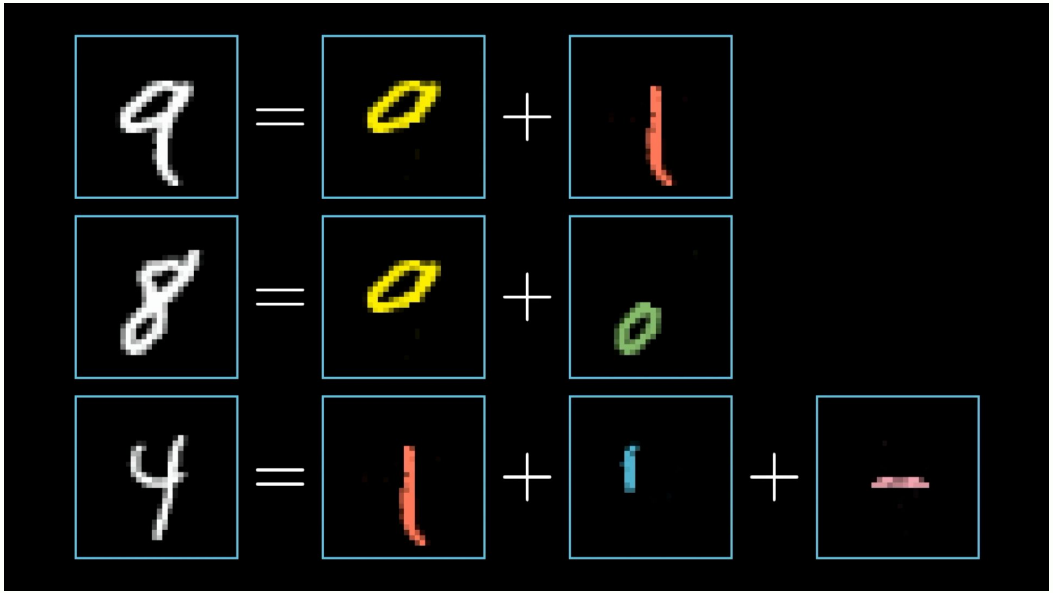
Oregon State University

MNIST → Neural Network

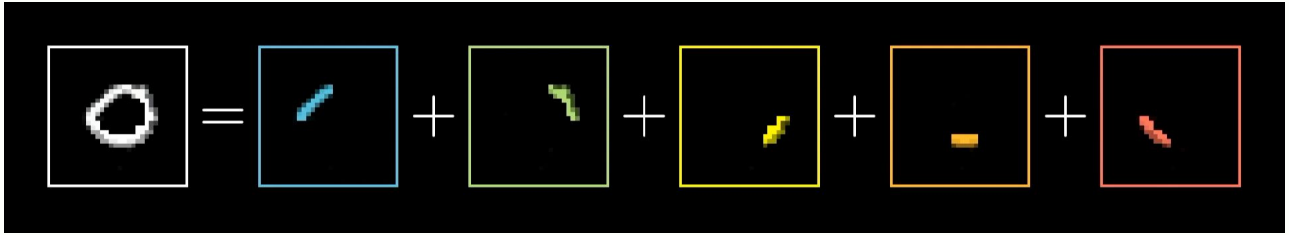


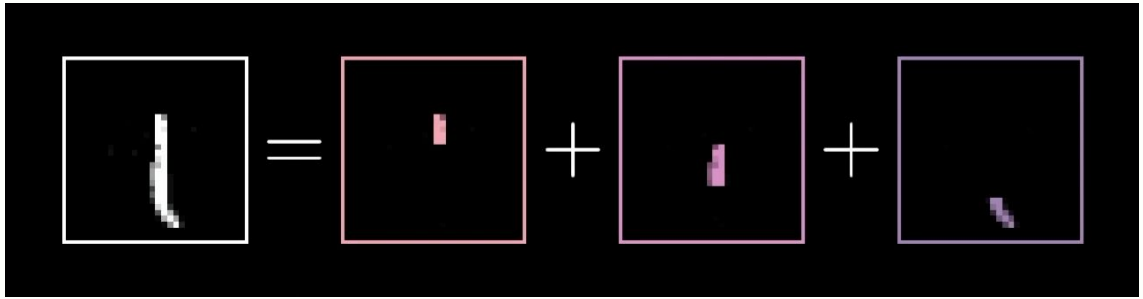
Oregon State University

Hidden Layers → Features

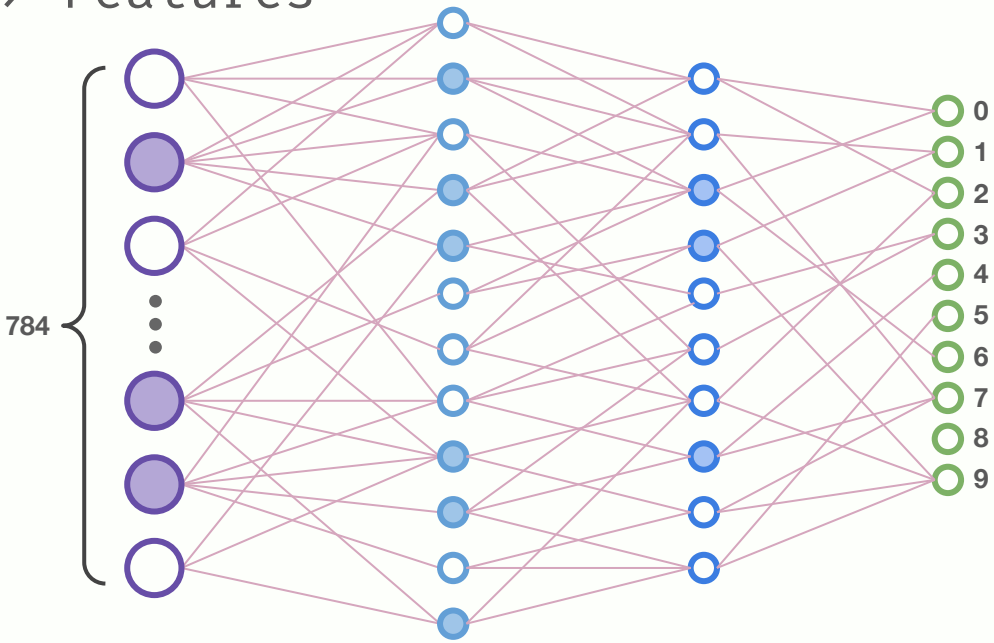


Hidden Layers → Features



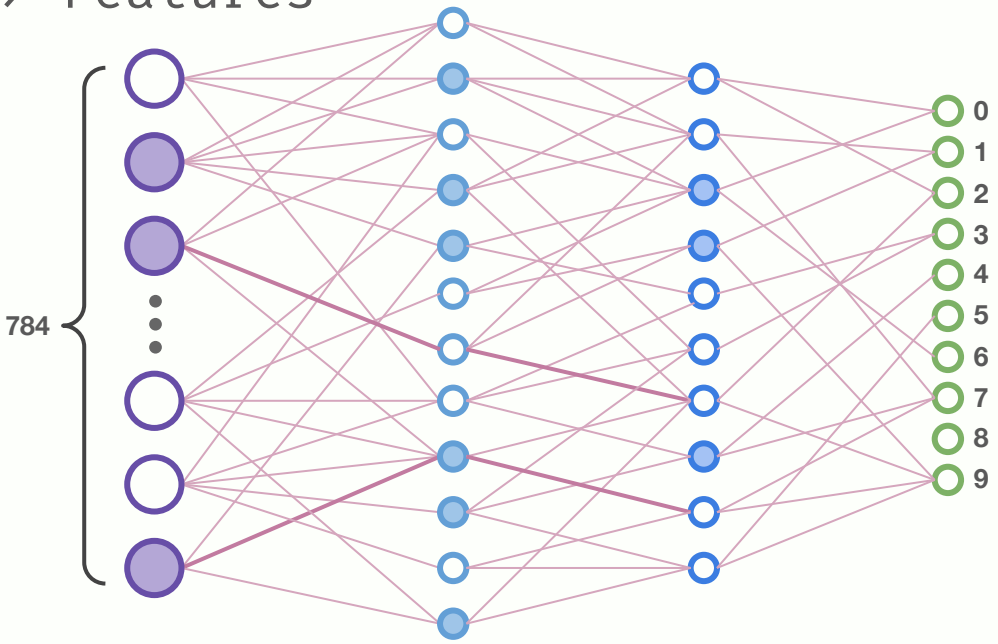


Hidden Layers → Features



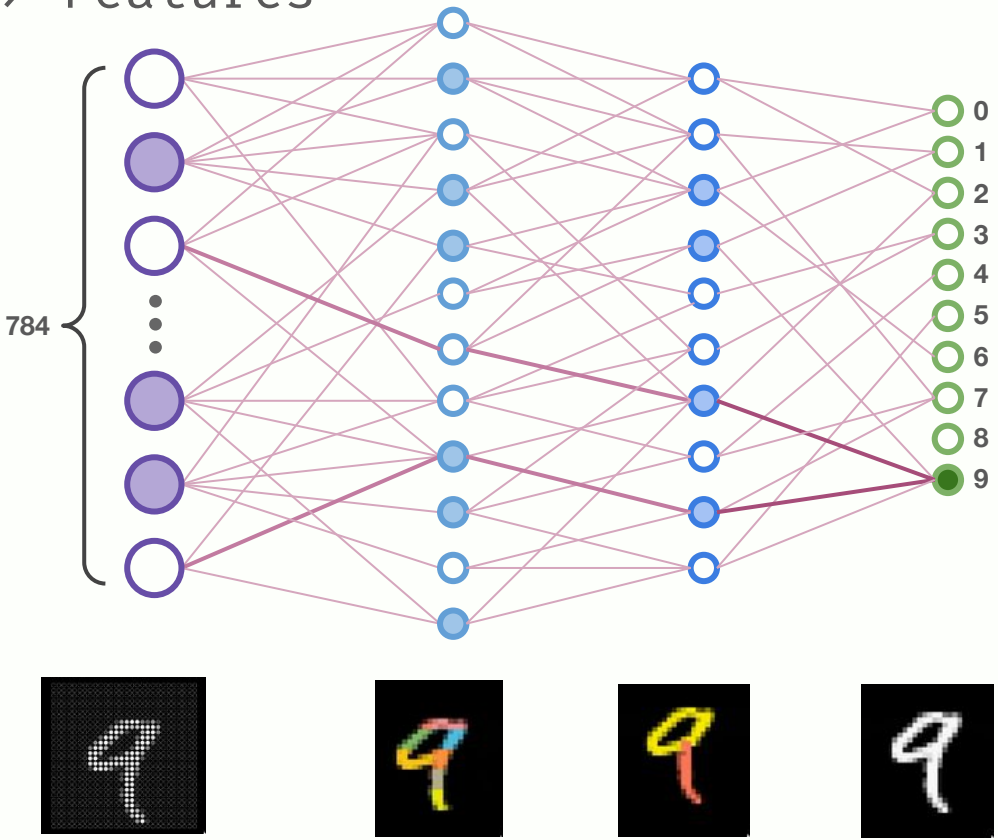
Oregon State University

Hidden Layers → Features



Oregon State University

Hidden Layers → Features



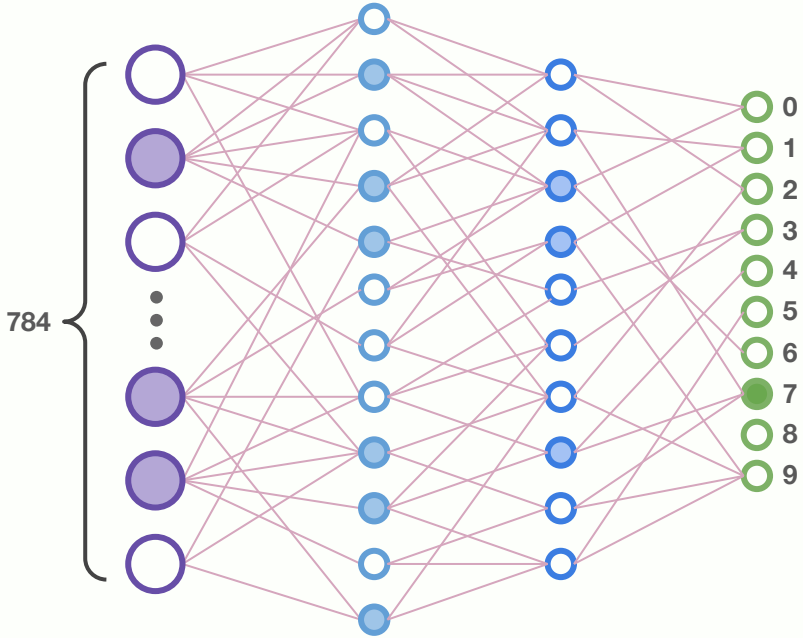
Oregon State University

Hidden Layers → Features

(Roth et al, 2018)

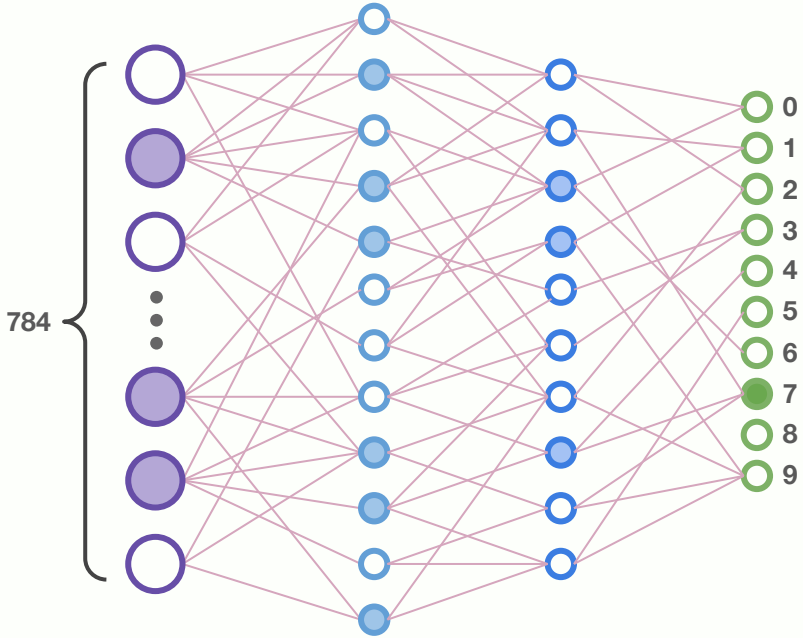
Oregon State University

MNIST → Neural Network



Oregon State University

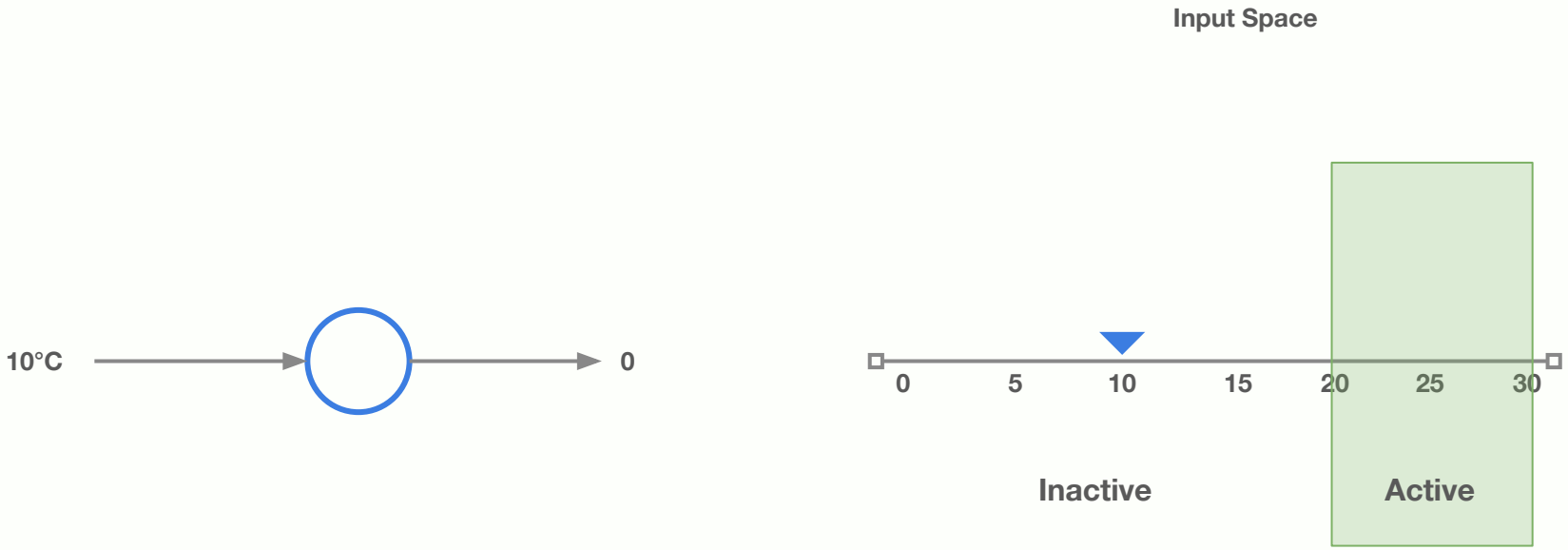
MNIST → Neural Network



$$y = \sigma(\, w_1 a_1 + w_2 a_2 + \ldots + b \,)$$

Oregon State University

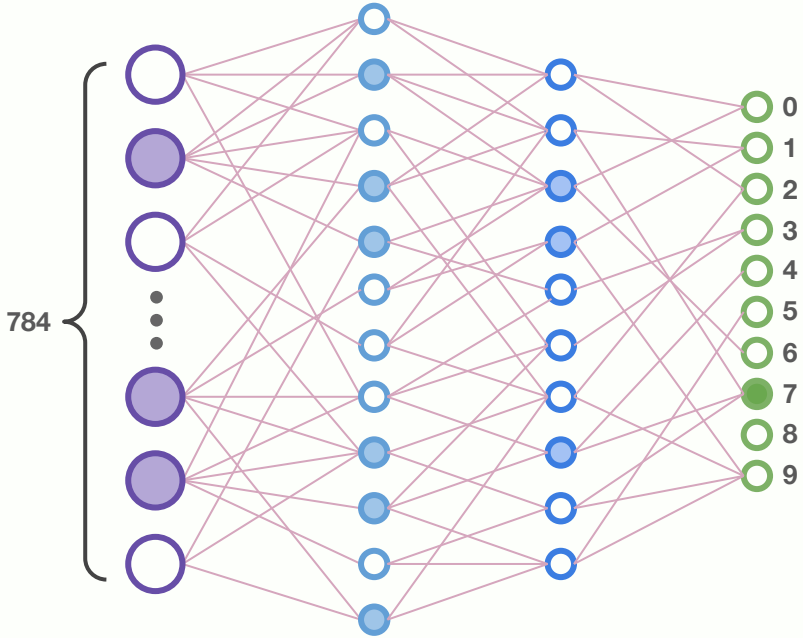
Recap: Perceptron → Bias



$$y = H(x - 20) = \begin{cases} 1 & \text{if } x - 20 \geq 0 \\ 0 & \text{if } x - 20 < 0 \end{cases}$$

Oregon State University

MNIST → Neural Network

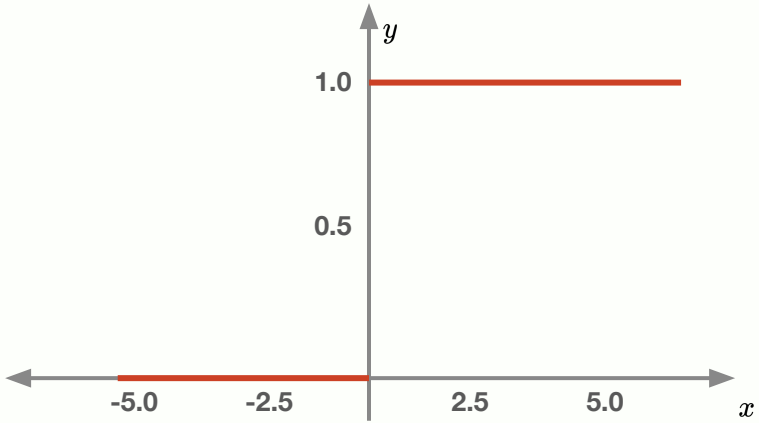


$$y = \sigma(\, w_1 a_1 + w_2 a_2 + \ldots + b \,)$$

Oregon State University

Recap: Activation Functions → Heaviside Step Function

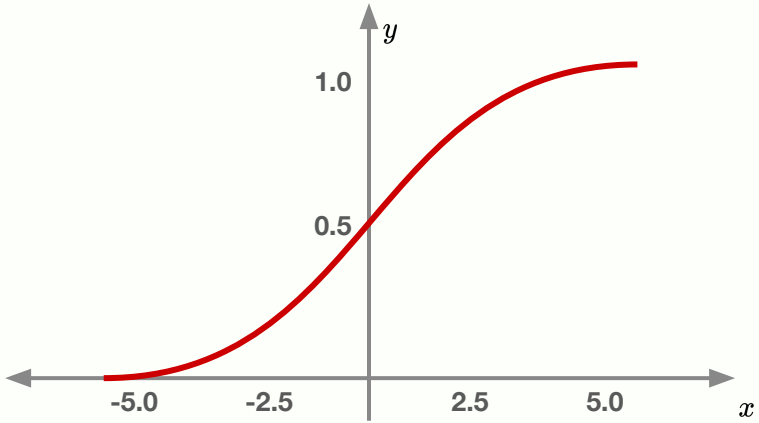
$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



Oregon State University

Recap: Activation Functions → Sigmoid

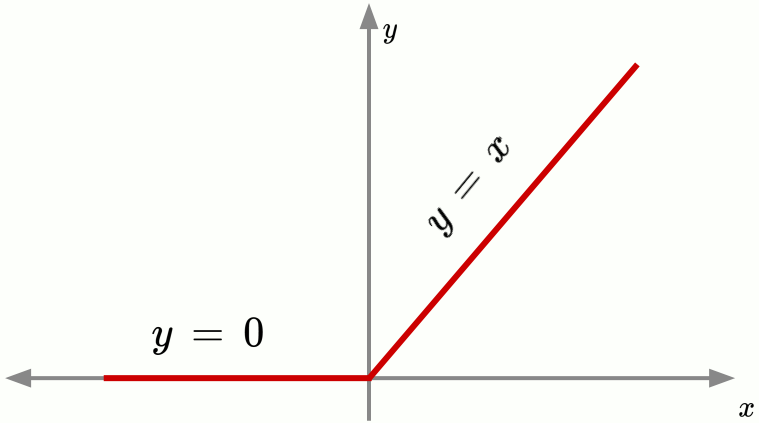
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



Oregon State University

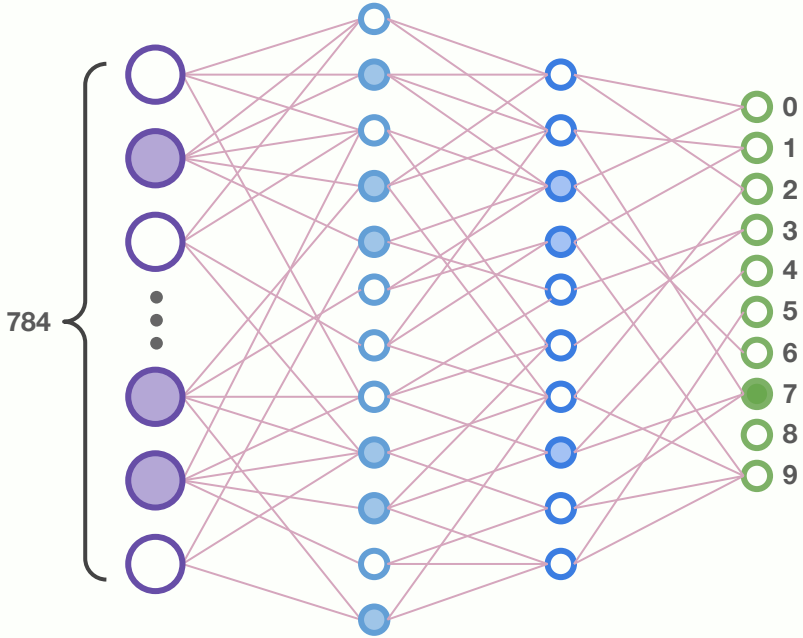
Recap: Activation Functions → ReLU

$$ReLU(x) = \max(0, x) = (x)^+$$



Oregon State University

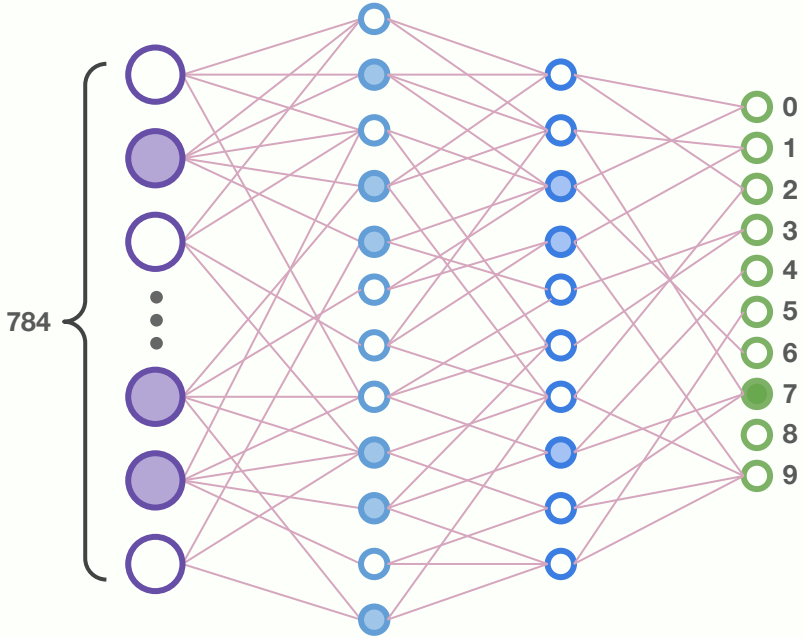
MNIST → Neural Network



$$y = \sigma(\, w_1 a_1 + w_2 a_2 + \ldots + b \,)$$

Oregon State University

MNIST → Neural Network



$$y = \sigma(\, w_1 a_1 + w_2 a_2 + \ldots + b \,)$$

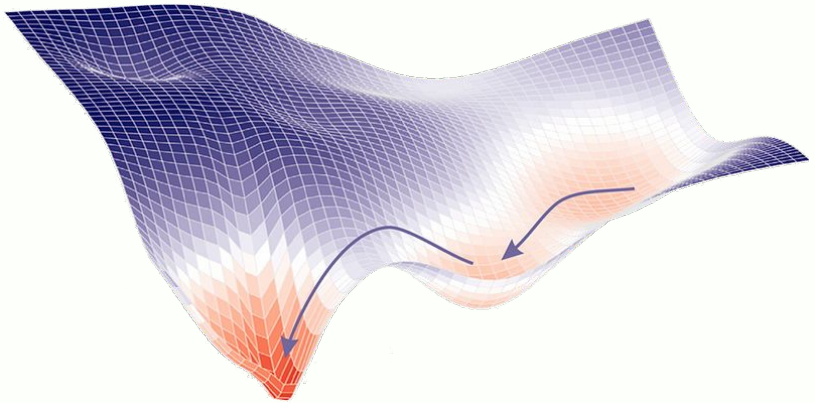
784 *x* **16 + 16** *x* **16 + 16** *x* **10**
weights

16 +16 +10
biases

13,002

Oregon State University

Gradient Descent

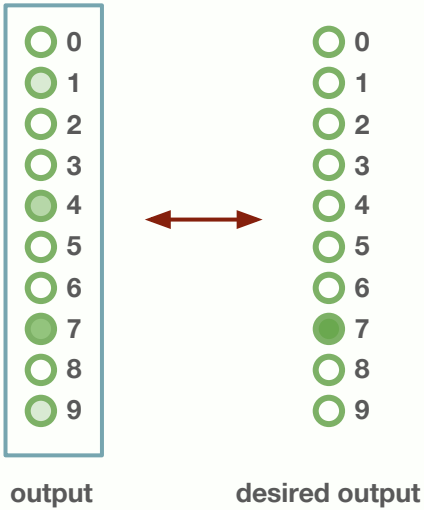


Error

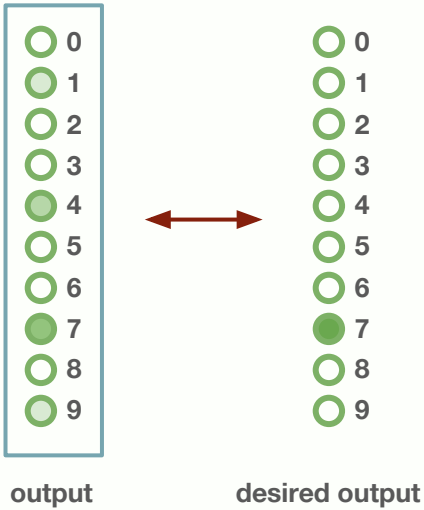
Oregon State University

Error

Error



Error



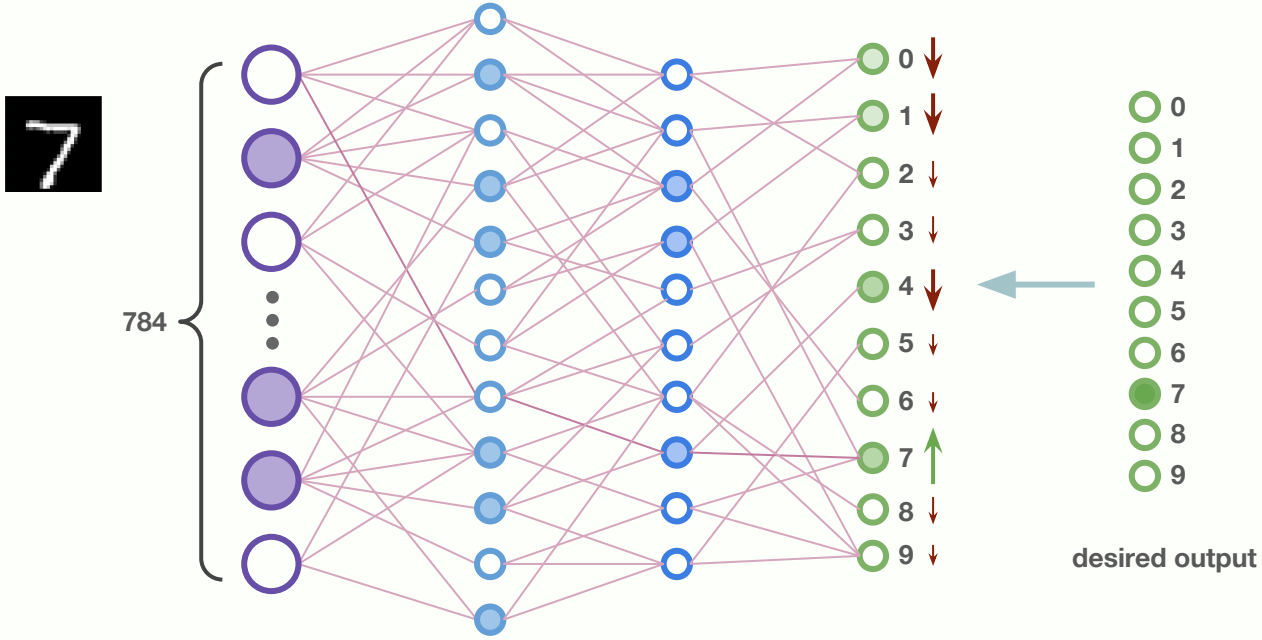
Error in predicting 

$$
\begin{aligned}
&(0.43 - 0.00)^2 + \\
&(0.28 - 0.00)^2 + \\
&(0.19 - 0.00)^2 + \\
&(0.38 - 0.00)^2 + \\
&(0.72 - 0.00)^2 + \\
&(0.01 - 0.00)^2 + \\
&(0.14 - 0.00)^2 + \\
&(0.96 - 1.00)^2 + \\
&(0.29 - 0.00)^2 + \\
&(0.63 - 0.00)^2
\end{aligned}
$$

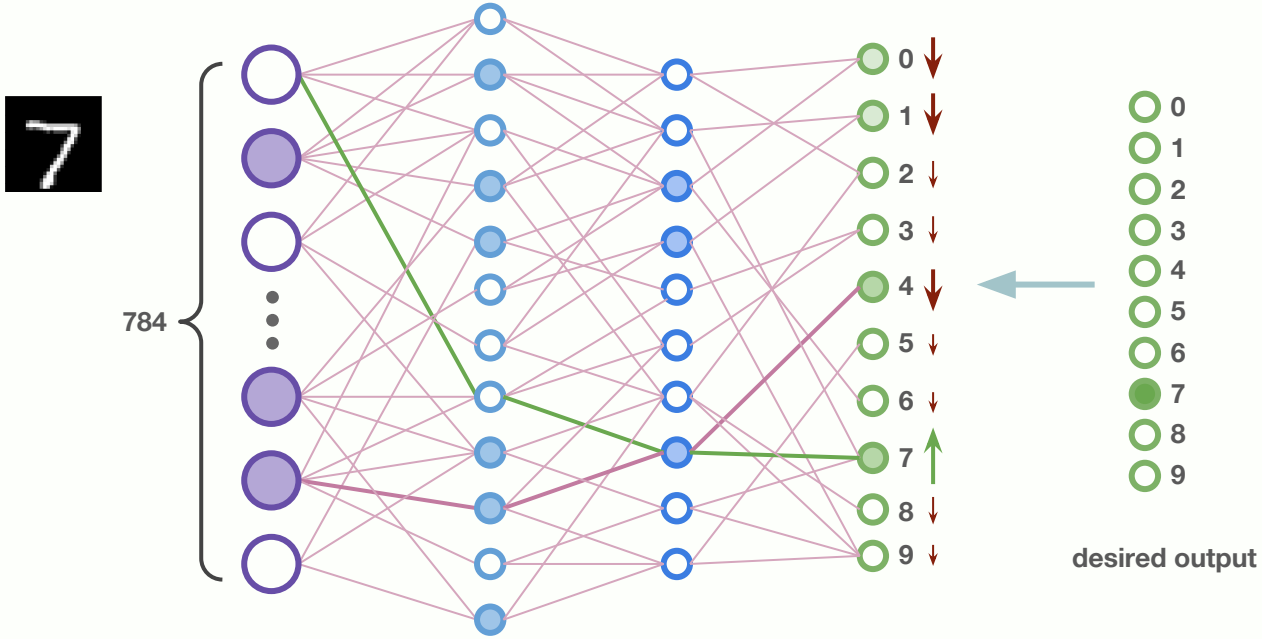
Oregon State University

Back Propagation

Back Propagation



Back Propagation

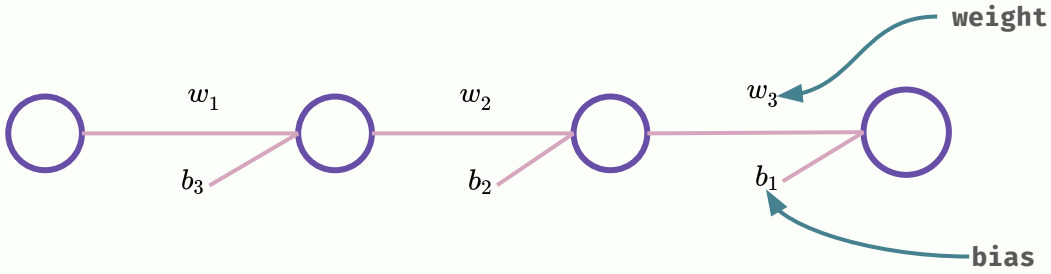


Back Propagation

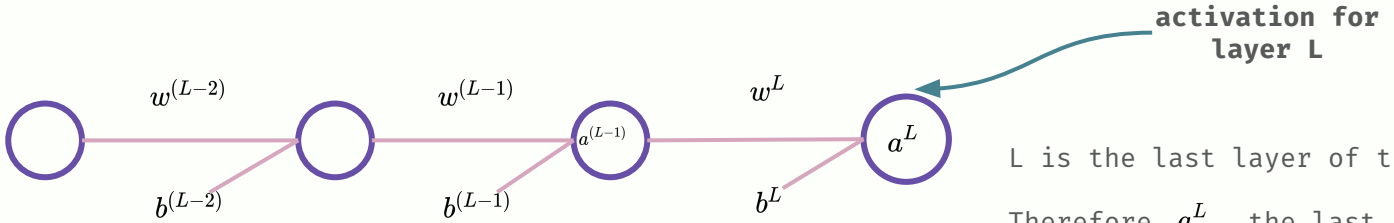
Calculating gradients with backpropagation

Back Propagation

Simple network with three layers, each layer has just one neuron
(the first neuron is the input)



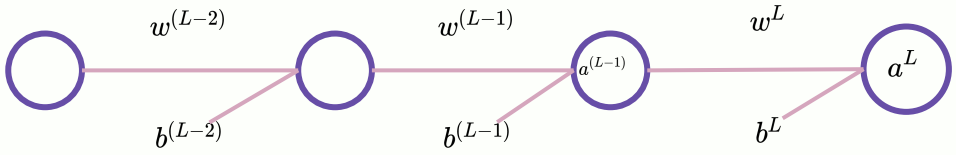
Generalize the notation for a network with L layers:



L is the last layer of the network

Therefore, a^L , the last activation, is the output of this network

Back Propagation



What we know already:

$$E_0 = \left(a^L - y\right)^2$$

$$a^L = \sigma\left(w^L a^{(L-1)} + b^L\right)$$

$$z^L = w^L a^{(L-1)} + b^L$$

$$a^L = \sigma\left(z^L\right)$$

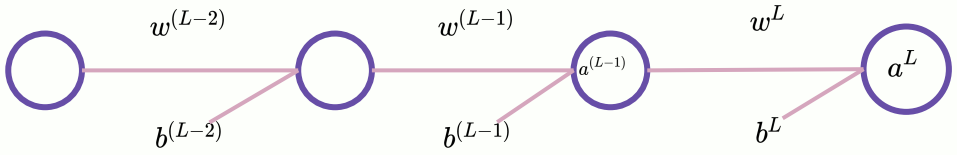
Where → E_0 is the error in predicting output y, for the first sample in the training data

→ σ is the activation function

→ z^L is a shorthand for the weighted sum that is fed to the activation function

Oregon State University

Back Propagation



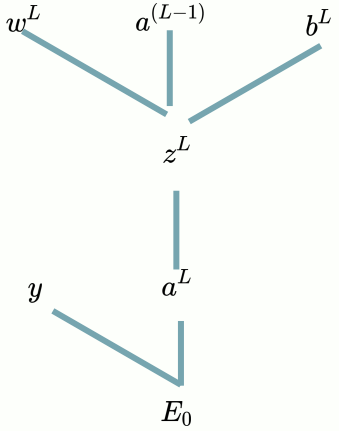
What we know already:

$$E_0 = \left(a^L - y\right)^2$$

$$a^L = \sigma\left(w^L a^{(L-1)} + b^L\right)$$

$$z^L = w^L a^{(L-1)} + b^L$$

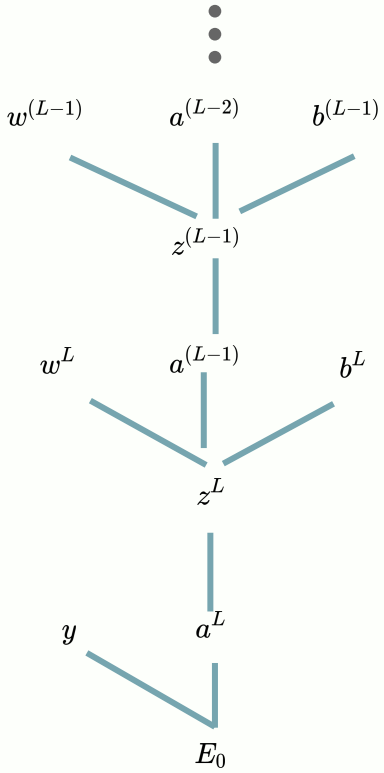
$$a^L = \sigma\left(z^L\right)$$



Value Dependency Tree

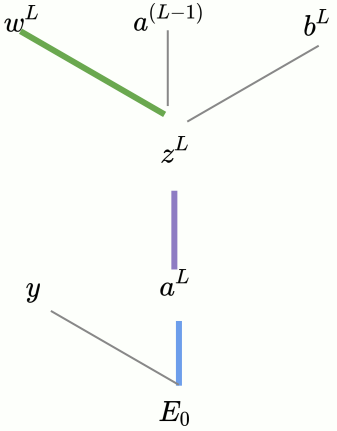
Value Dependency Tree: Recursive

The tree of dependencies goes all the way to the input neuron, the first activation, a^0



Value Dependency Tree

How much does a nudge to the weight w^L change error E_0 ?



Value Dependency Tree

Chain rule to find sensitivity of Error E_0 to changes in the weight of the last layer w^L :

$$\frac{\partial E_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}$$

How much does a nudge to weight w^L change z^L ?

How much does a nudge to the weighted sum z^L change a^L ?

How much does a nudge to the activation a^L change E_0 ?

Constituent Derivatives

$$\frac{\partial E_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}$$

What we know already

Computing constituent derivatives

$$z^L = w^L a^{(L-1)} + b^L \quad \rightarrow \quad \frac{\partial z^L}{\partial w^L} = a^{(L-1)}$$

$$a^L = \sigma(z^L) \quad \rightarrow \quad \frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

$$E_0 = (a^L - y)^2 \quad \rightarrow \quad \frac{\partial E_0}{\partial a^L} = 2(a^L - y)$$

Oregon State University

Constituent Derivatives

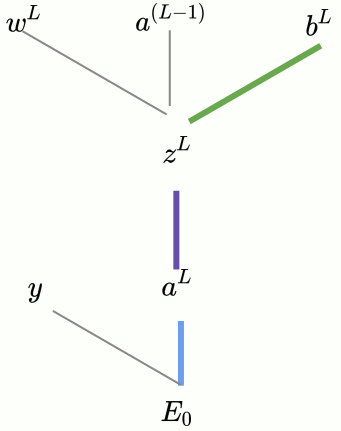
$$\frac{\partial E_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}$$

$$\frac{\partial E_0}{\partial w^L} = 2a^{(L-1)}\sigma'\left(z^L\right)\left(a^L - y\right)$$

$$\Delta w^L = \eta \frac{\partial E_0}{\partial w^L}$$

Oregon State University

How much does a nudge to the bias b^L change error E_0 ?



Value Dependency Tree

Chain rule to find sensitivity of Error E_0 to changes in the bias of the last layer b^L :

$$\frac{\partial E_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L} \frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}$$

How much does a nudge to bias b^L change z^L ?

How much does a nudge to the weighted sum z^L change a^L ?

How much does a nudge to the activation a^L change E_0 ?

$$\frac{\partial z^L}{\partial b^L} = 1$$

(from previous: $z^L = w^L a^{(L-1)} + b^L$)

Weight and Bias Updates for the last layer L

Weight update:

$$\frac{\partial E_0}{\partial w^L} = 2a^{(L-1)}\sigma'\left(z^L\right)\left(a^L - y\right)$$

$$\Delta w^L = \eta\frac{\partial E_0}{\partial w^L}$$

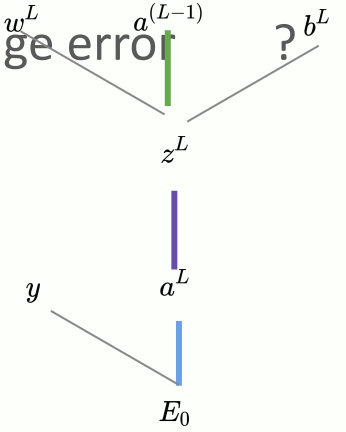
Bias update:

$$\frac{\partial E_0}{\partial b^L} = \frac{\partial a^L}{\partial z^L}\frac{\partial E_0}{\partial a^L} = 2\sigma'\left(z^L\right)\left(a^L - y\right)$$

$$\Delta b^L = \eta\frac{\partial E_0}{\partial b^L}$$

Oregon State University

How much does a nudge to the previous activation $a^{(L-1)}$ change error E_0 ?



Value Dependency Tree

Chain rule to find sensitivity of Error E_0 to changes in the previous activation $a^{(L-1)}$:

$$\frac{\partial E_0}{\partial a^{(L-1)}} = \frac{\partial z^L}{\partial a^{(L-1)}} \frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}$$

How much does a nudge to bias $a^{(L-1)}$ change z^L ?

How much does a nudge to the weighted sum z^L change a^L ?

How much does a nudge to the activation a^L change E_0 ?

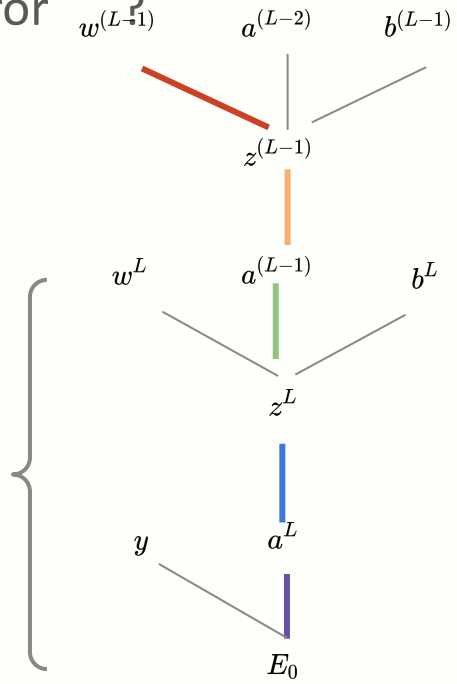
$$\frac{\partial E_0}{\partial a^{(L-1)}} = 2w^L \sigma'(z^L)(a^L - y)$$

How much does a nudge to the weight $w^{(L-1)}$ change error E_0?

Follow the dependency tree as a guide to write the constituent derivatives:

$$\frac{\partial E_0}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^L}{\partial a^{(L-1)}} \underbrace{\frac{\partial a^L}{\partial z^L} \frac{\partial E_0}{\partial a^L}}_{\frac{\partial E_0}{\partial a^{(L-1)}}}$$

$$\frac{\partial E_0}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial E_0}{\partial a^{(L-1)}}$$



Value Dependency Tree

Weight Updates for any layer L (follow the same exercise for bias updates)

Weight update:

$$\frac{\partial E_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L}\frac{\partial a^L}{\partial z^L}\frac{\partial E_0}{\partial a^L} \quad \text{——— 1}$$

$$\frac{\partial E_0}{\partial a^{(L-1)}} = \frac{\partial z^L}{\partial a^{(L-1)}}\frac{\partial a^L}{\partial z^L}\frac{\partial E_0}{\partial a^L} \quad \text{——— 2}$$

For any layer L, update the weight w^L using **equation 1**.

If L is the last layer, $\frac{\partial E_0}{\partial a^L}$ in **equation 1** resolves to $2(a^L - y)$.

If L is not the last layer, $\frac{\partial E_0}{\partial a^L}$ in **equation 1** resolves to **equation 2**

$$\Delta w^L = \eta \frac{\partial E_0}{\partial w^L}$$

Oregon State University

Mini-Batch Updates

Equations so far were derived for error in a single training sample $E_{s=0}$.

We typically calculate error over an entire mini-batch of size **S,** by averaging individual errors.

For instance, the gradient for weight w^L across an entire mini-batch is given by:

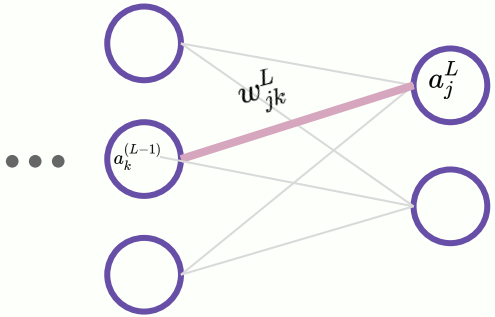
$$\frac{\partial E}{\partial w^L} = \frac{1}{S}\sum_{s=0}^{S}\frac{\partial E_s}{\partial w^L}$$

The weight is then updated:

$$\Delta w^L = \eta\frac{\partial E}{\partial w^L}$$

Oregon State University

Neural Networks: Layers With Multiple Neurons



L is the last layer.

Neurons in layer **(L-1)** are indexed with **k**.

Neurons in layer **L** are indexed with **j**.

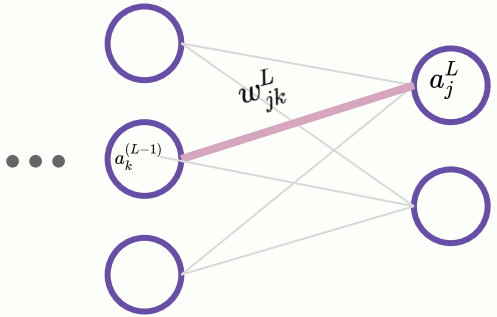
Updates largely stay the same, except we now explicitly take into account neuron indices

Oregon State University

Neural Networks: Layers With Multiple Neurons

$$E_0 = \sum_{j=0}^{n_L - 1} \left(a_j^L - y_j\right)^2$$

Size of the output vector **y** is equal to the number of neurons n_L in the output layer **L**



The weighted sum z_j^L for neuron **j** in layer **L** is the weighted sum of activations over all neurons in the previous layer **(L-1)**:

$$z_j^L = w_{j0}^L a_0^{(L-1)} + w_{j1}^L a_1^{(L-1)} + w_{j2}^L a_2^{(L-1)} + b_j^L$$

Generally, for neuron **j** in any layer **L**:

$$z_j^L = \left(\sum_{k=0}^{n_{(L-1)} - 1} w_{jk}^L a_k^{(L-1)} \right) + b_j^L$$

Oregon State University

Neural Networks: Layers With Multiple Neurons → Weights and Biases

Gradients for weight and bias updates look exactly the same. Note the indices.

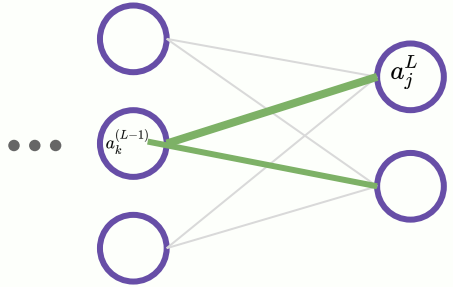
$$\frac{\partial E_0}{\partial w_{jk}^L} = \frac{\partial z_j^L}{\partial w_{jk}^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial E_0}{\partial a_j^L}$$

$$\frac{\partial E_0}{\partial b_j^L} = \frac{\partial z_j^L}{\partial b_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial E_0}{\partial a_j^L}$$

An activation at layer **(L-1)** affects error through multiple paths.

Gradient for activation $\frac{\partial E_0}{\partial a_k^{(L-1)}}$ incorporates this by summing over all the paths it connects to in its next layer **L**.

$$\frac{\partial E_0}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L-1} \frac{\partial z_j^L}{\partial a_k^{(L-1)}} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial E_0}{\partial a_j^L}$$



Activation $a_k^{(L-1)}$ affects error through two paths.

Oregon State University