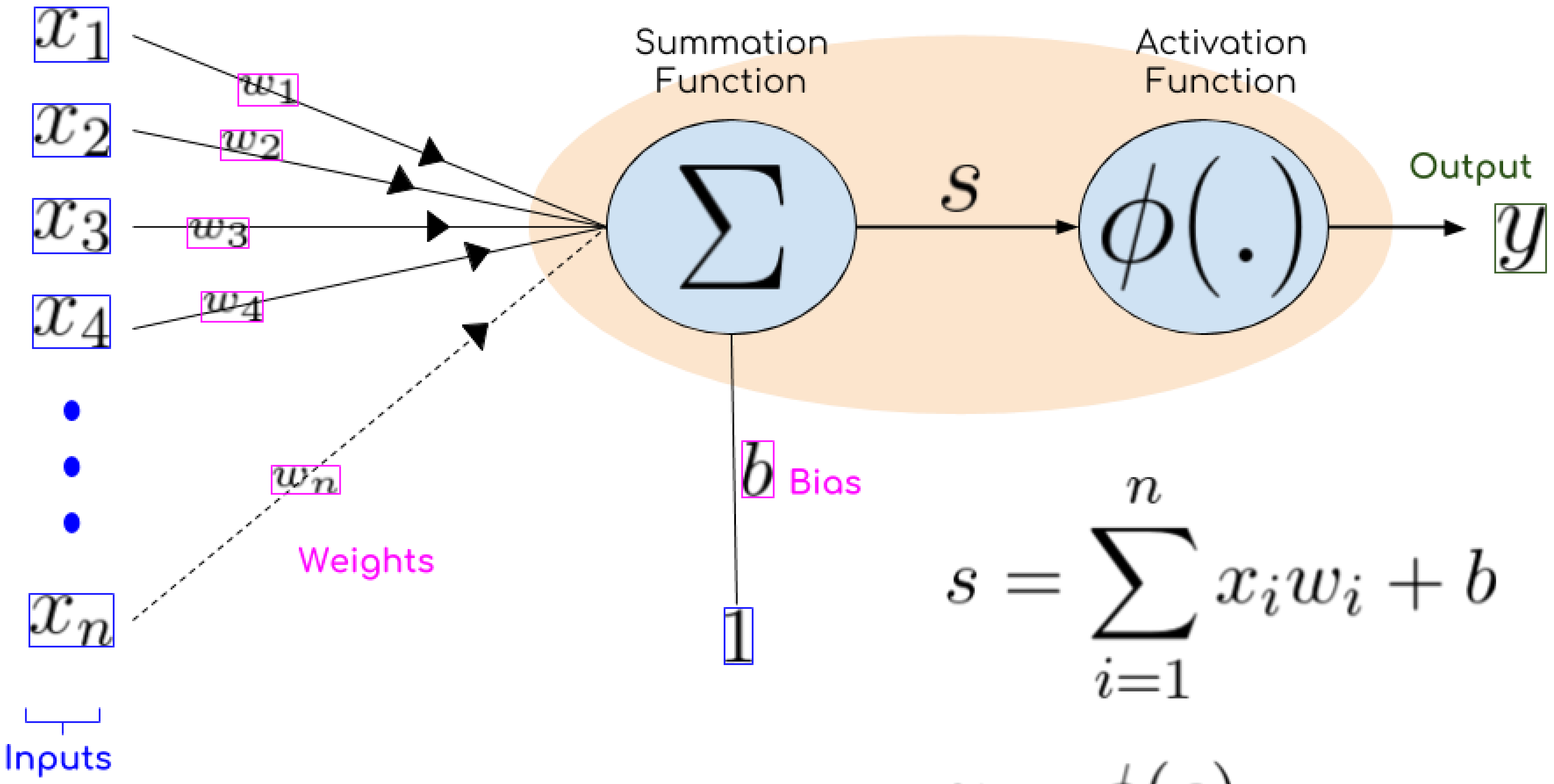
# Neural Networks

## Backpropagation, General Considerations

Ravi Kiran
CVIT, IIIT Hyderabad

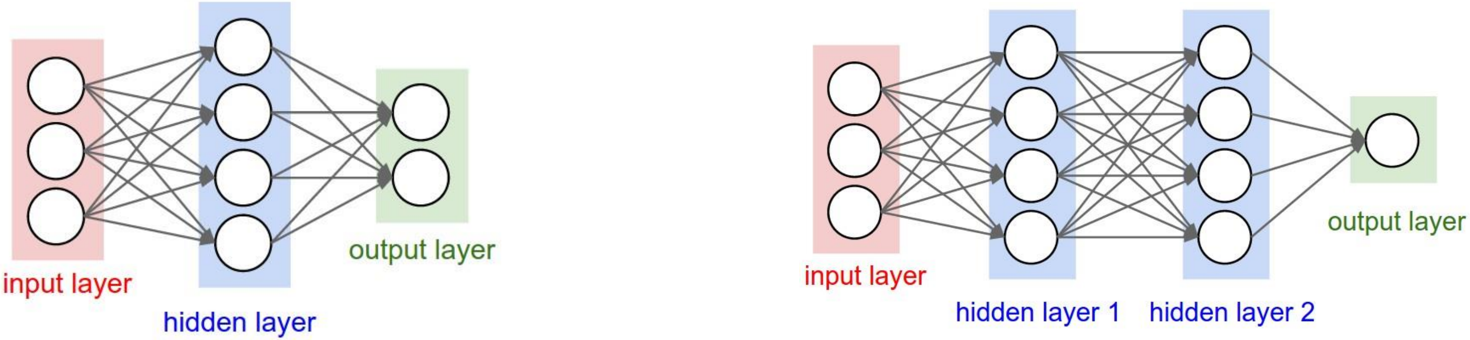
19.02.2019

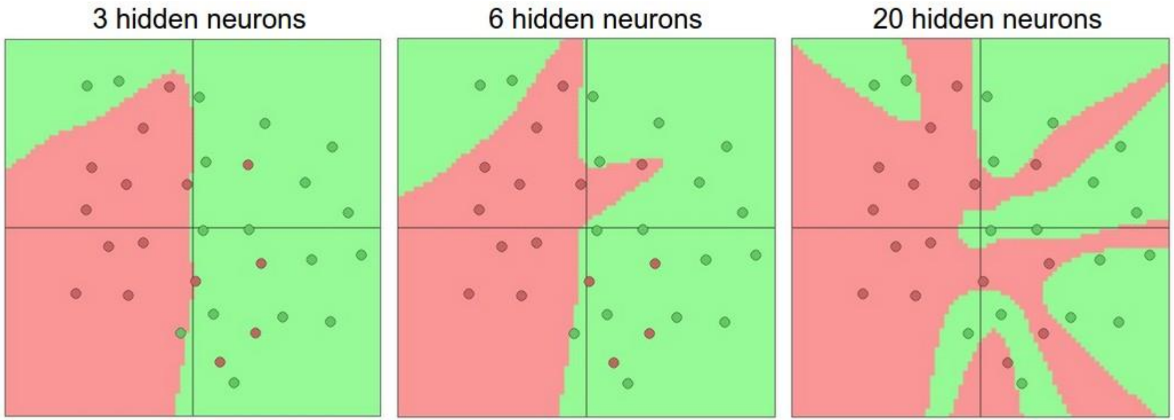



$$s = \sum_{i=1}^{n} x_i w_i + b$$

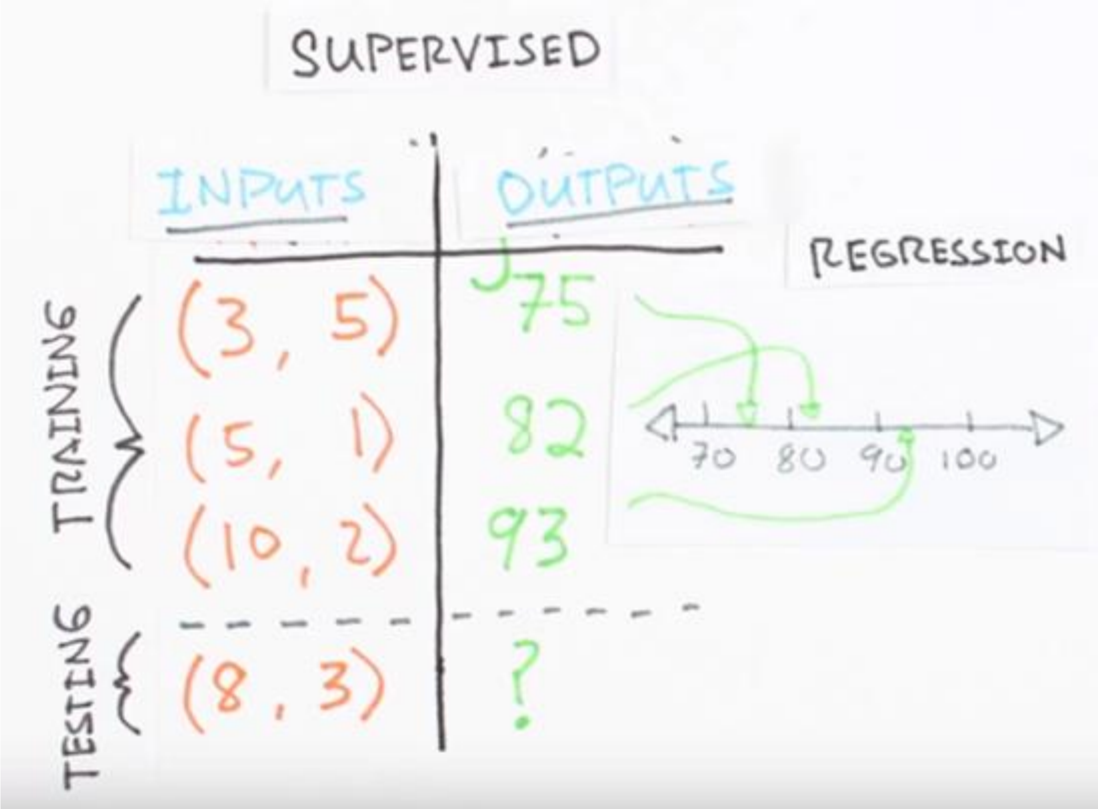
$$y = \phi(s)$$

# Why Use Only One Neuron ?

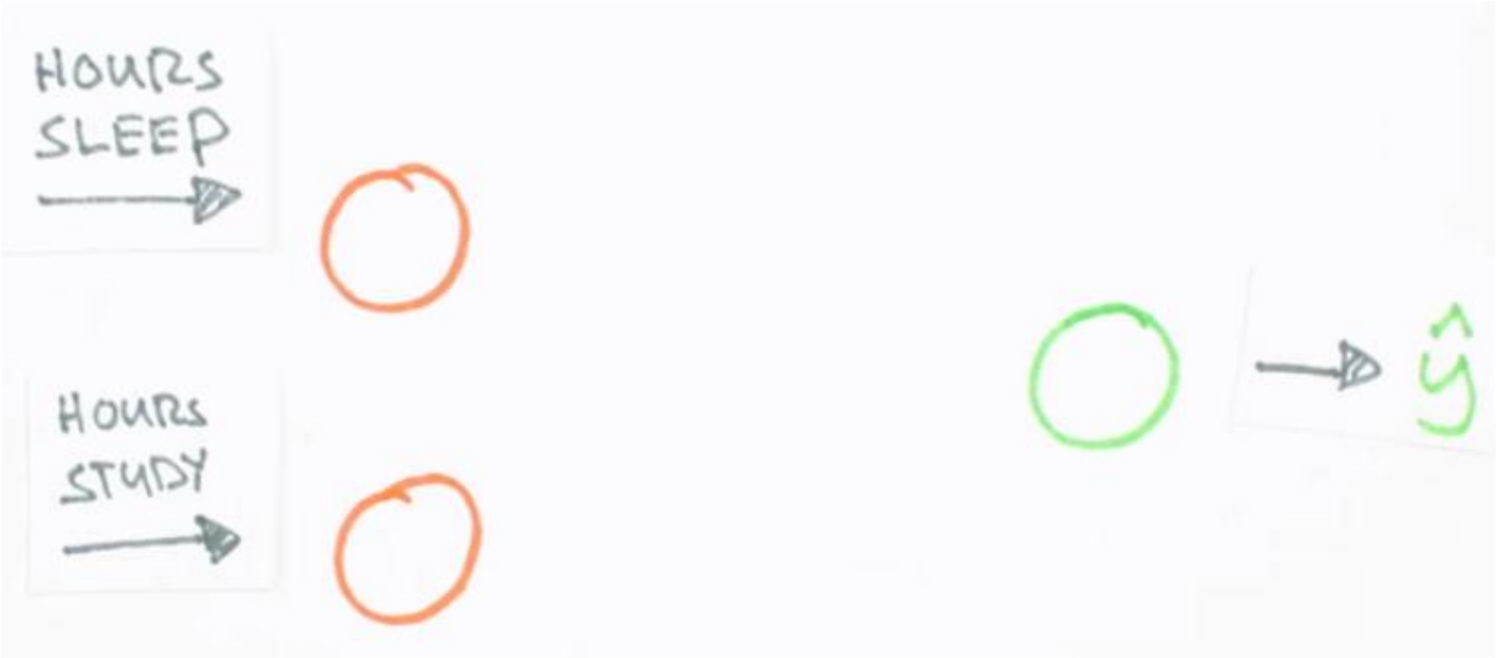


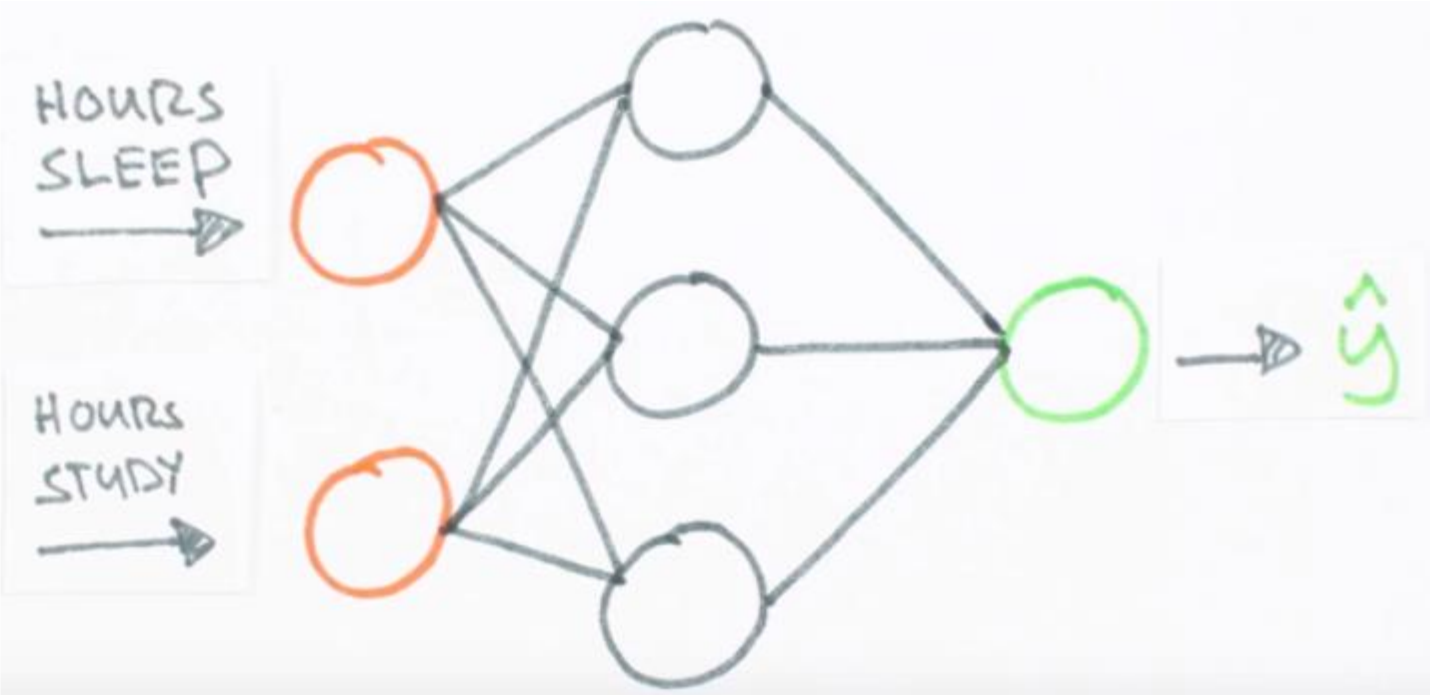
# Multi-Neuron Networks



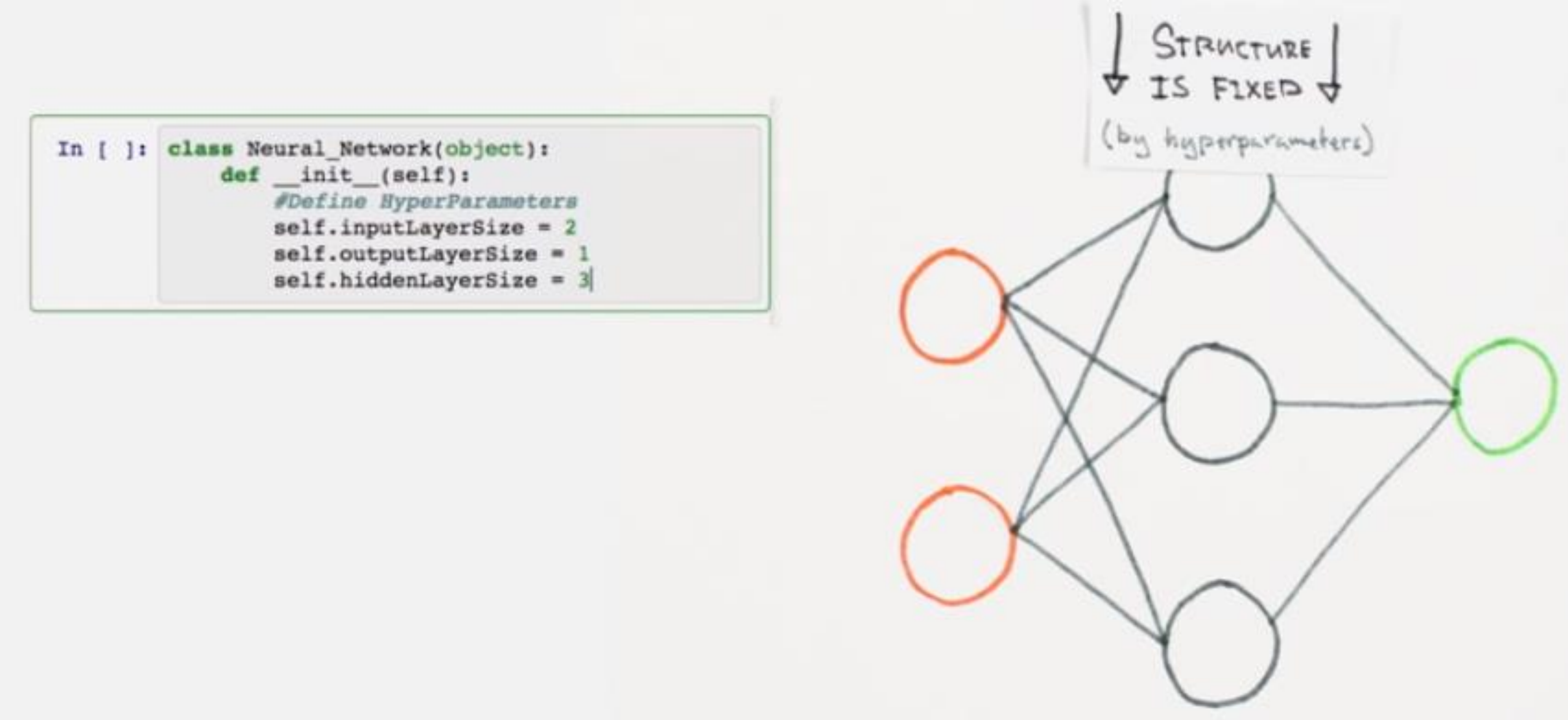
# Multi-Neuron Networks :: Architecture



# Multi-Neuron Networks :: Architecture



# Multi-Neuron Networks :: Architecture



# Gradient Descent

1. Initialize the parameters **w** to some guess (usually all zeros, or random values)
2. Update the parameters:
   $\mathbf{w} = \mathbf{w} - \eta \nabla L(\mathbf{w})$
3. Update the learning rate $\eta$
4. Repeat steps 2-3 until $\nabla L(\mathbf{w})$ is close to zero.

# Stopping Criteria

Stop when the norm of the gradient is below some threshold, θ:

$$||\nabla L(\mathbf{w})|| < \theta$$

Common values of θ are around .01, but if it is taking too long, you can make the threshold larger.

# Multi-Neuron Networks :: training

Initialize network with random weights

While [not converged]

Do forward prop

Do backprop and determine change in weights

Update all weights in all layers

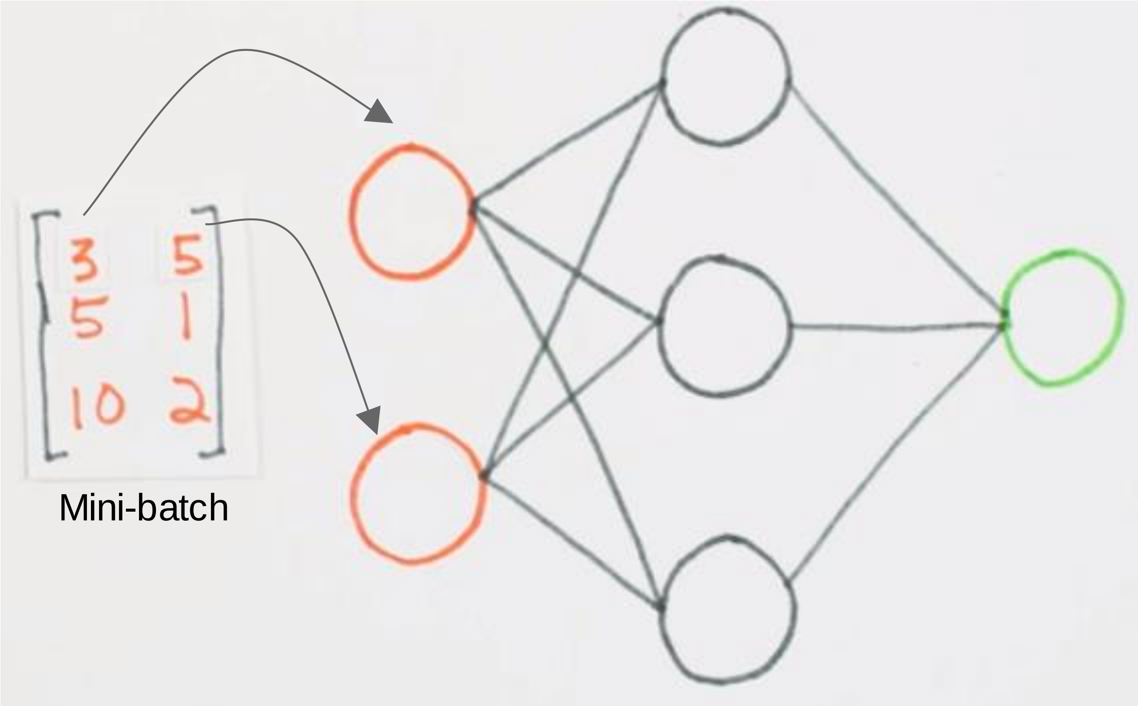
```
Initialize weights w ; // Random or 0

Until [all examples correctly classified]
        For each training sample (x,y)
                Compute yt := xTw
                 if y == yt // Correctly classified
                        continue ;
                else // Update weights
                        dw = (y - yt) * x ;
                        w   = w + dw;
                EndIf
        EndFor
EndUntil
```

# Multi-Neuron Networks :: Forward Propagation

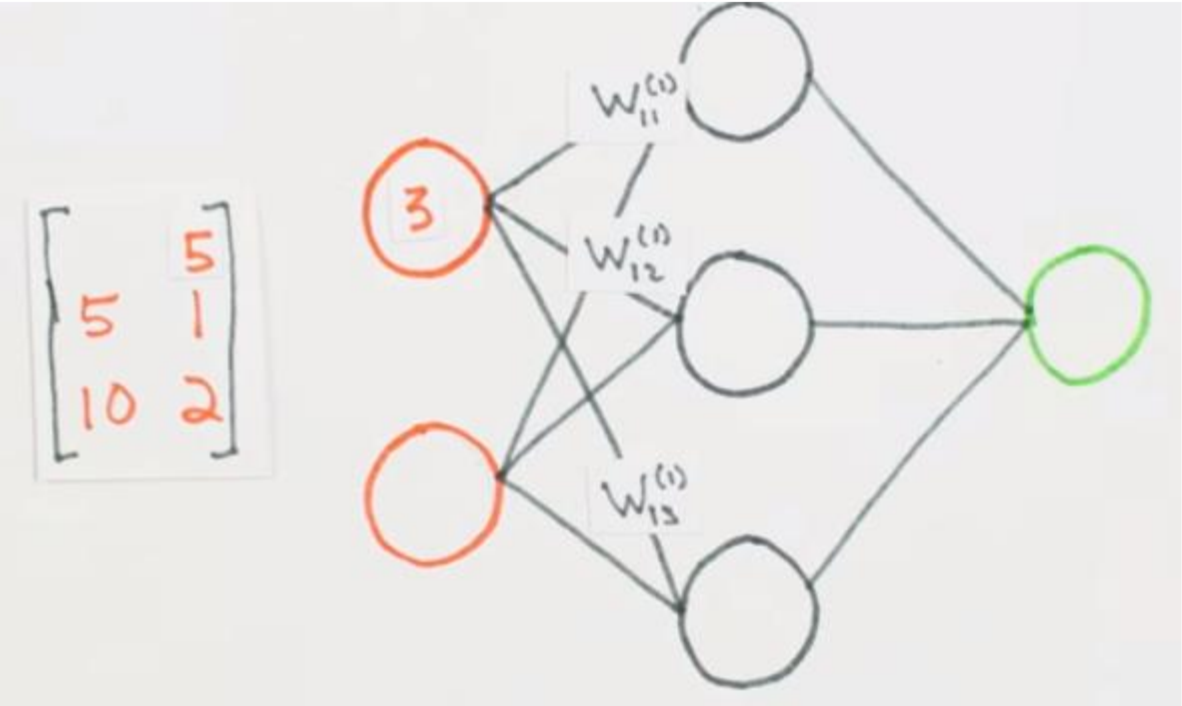
```
In [1]: class Neural_Network(object):
            def __init__(self):
                #Define HyperParameters
                self.inputLayerSize = 2
                self.outputLayerSize = 1
                self.hiddenLayerSize = 3

            def forward(self, X):
                #Propagate inputs through network
```

MATLAB NumPy

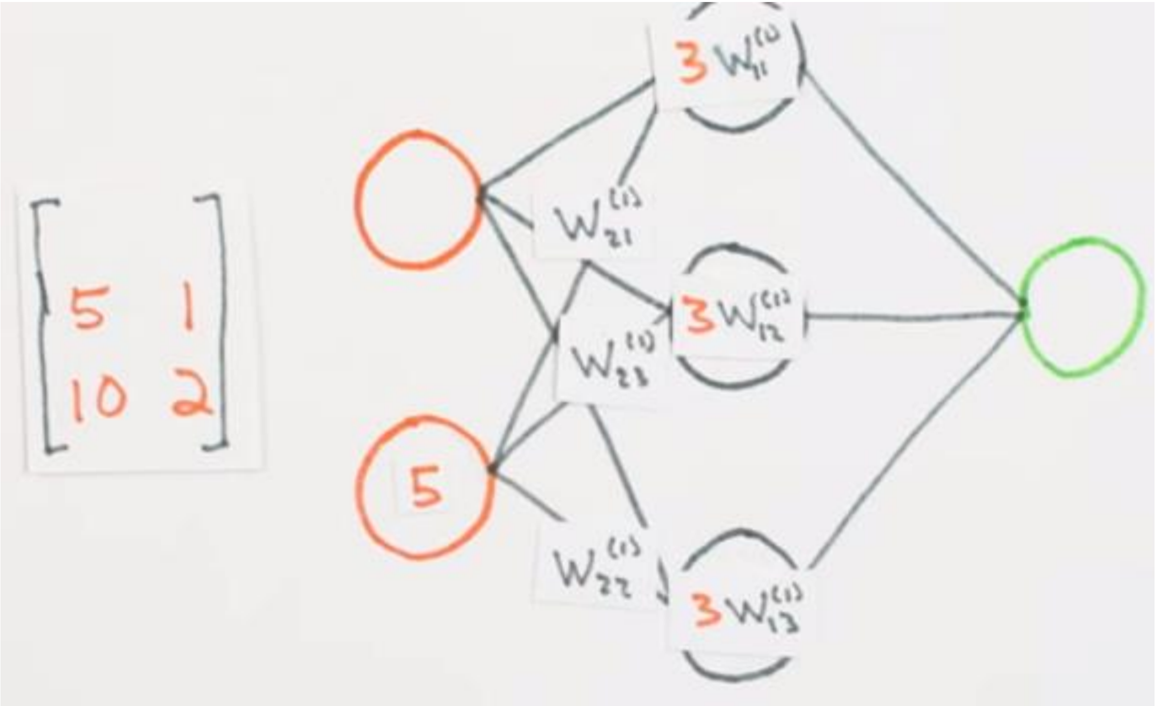
# Multi-Neuron Networks :: FORWARD PROPAGATION



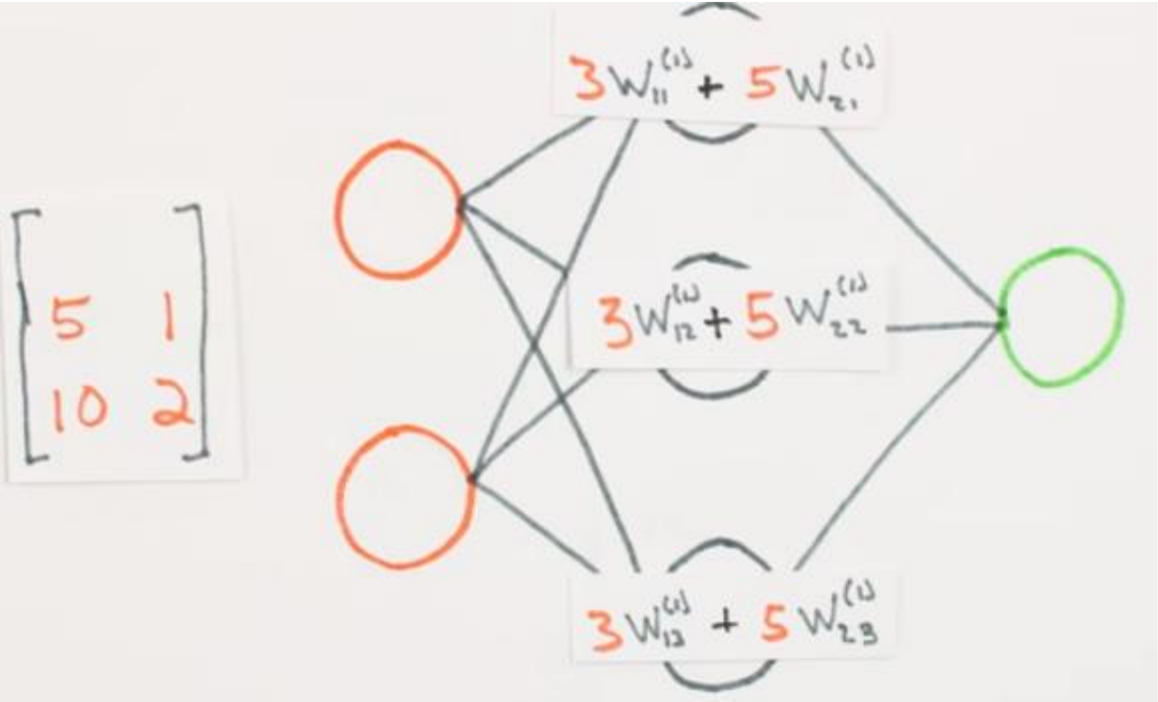
# Multi-Neuron Networks :: FORWARD PROPAGATION



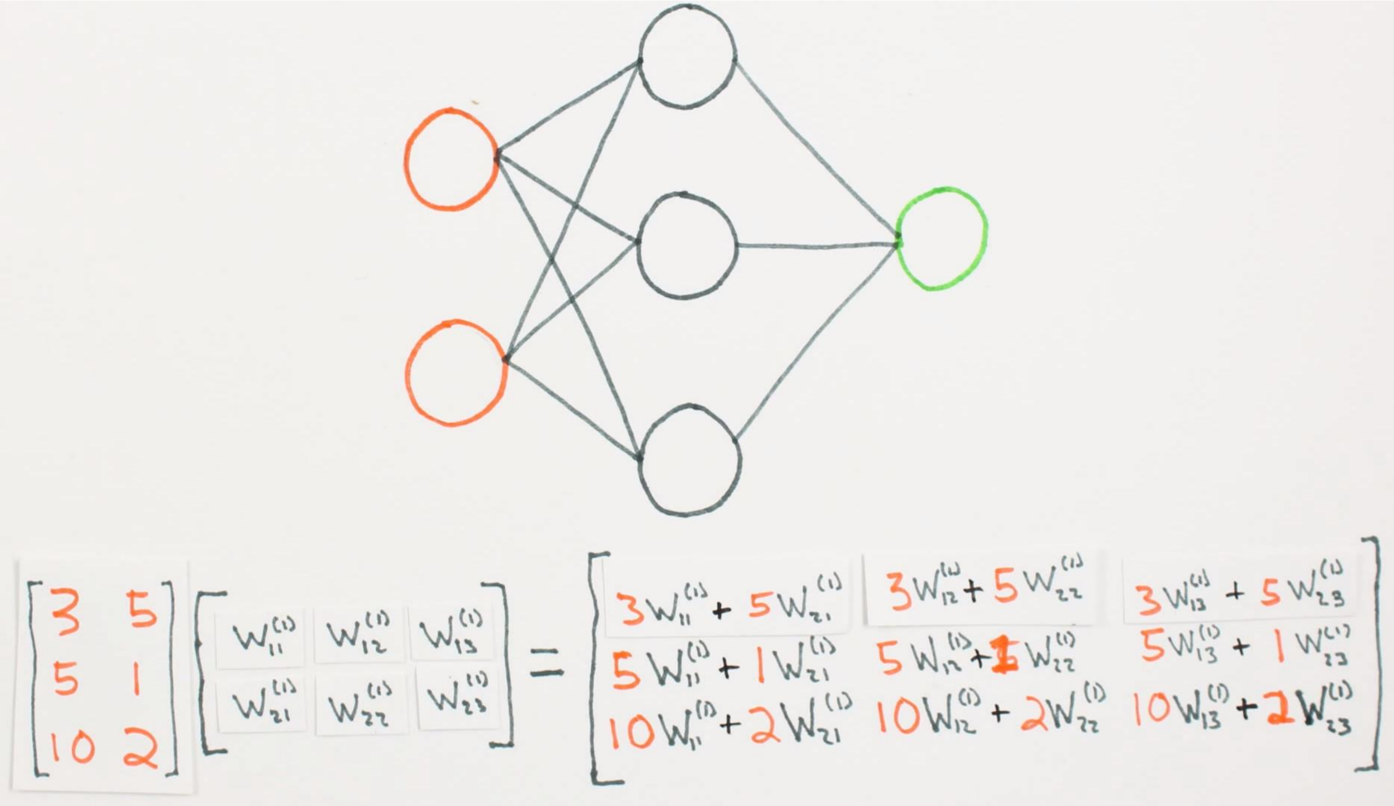
# Multi-Neuron Networks :: Forward Propagation



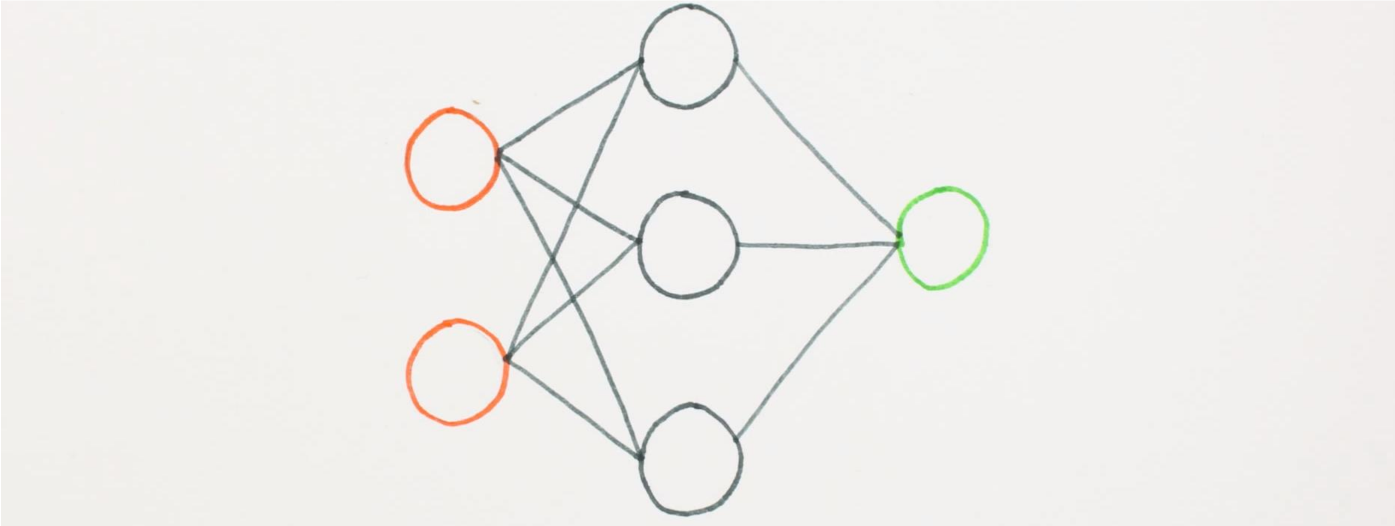
# Multi-Neuron Networks :: FORWARD PROPAGATION



# Multi-Neuron Networks :: forward propagation

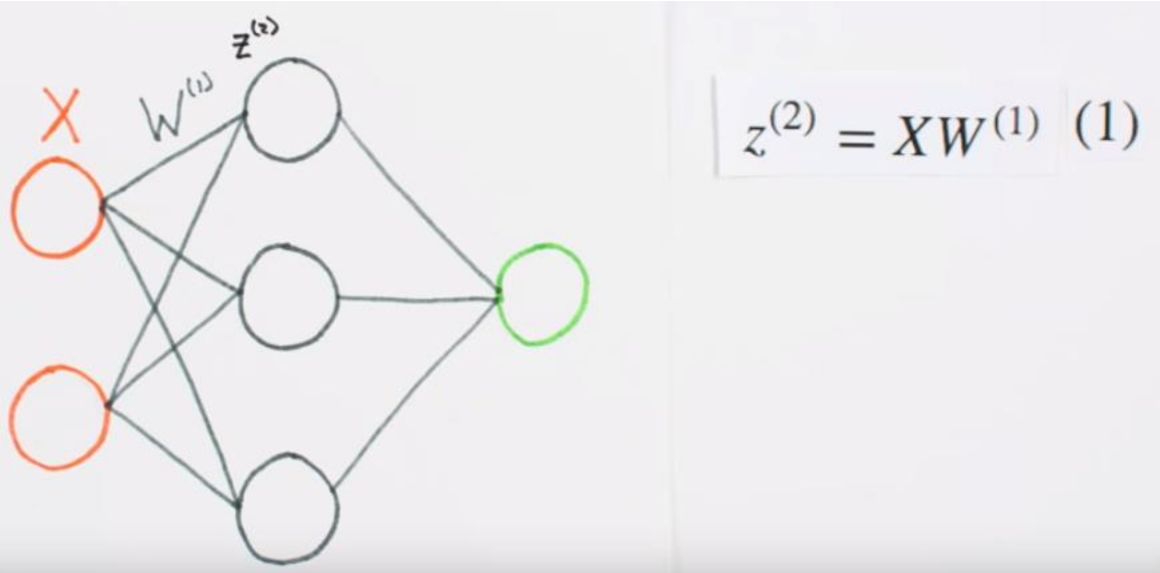


# Multi-Neuron Networks :: FORWARD PROPAGATION

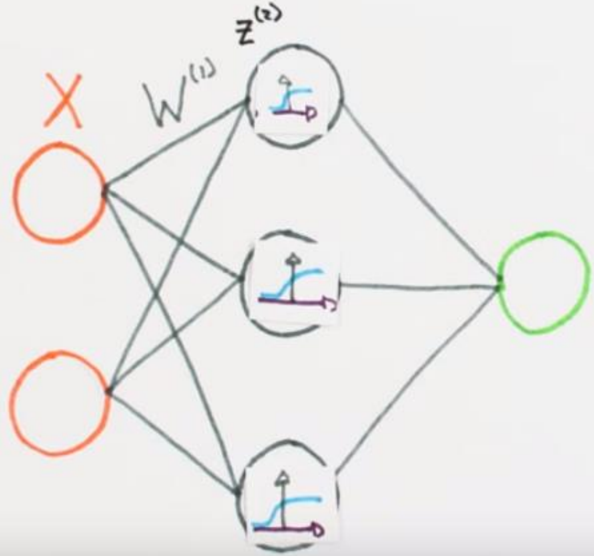


$$X \quad W^{(1)} = Z^{(2)}$$

# Multi-Neuron Networks :: FORWARD PROPAGATION

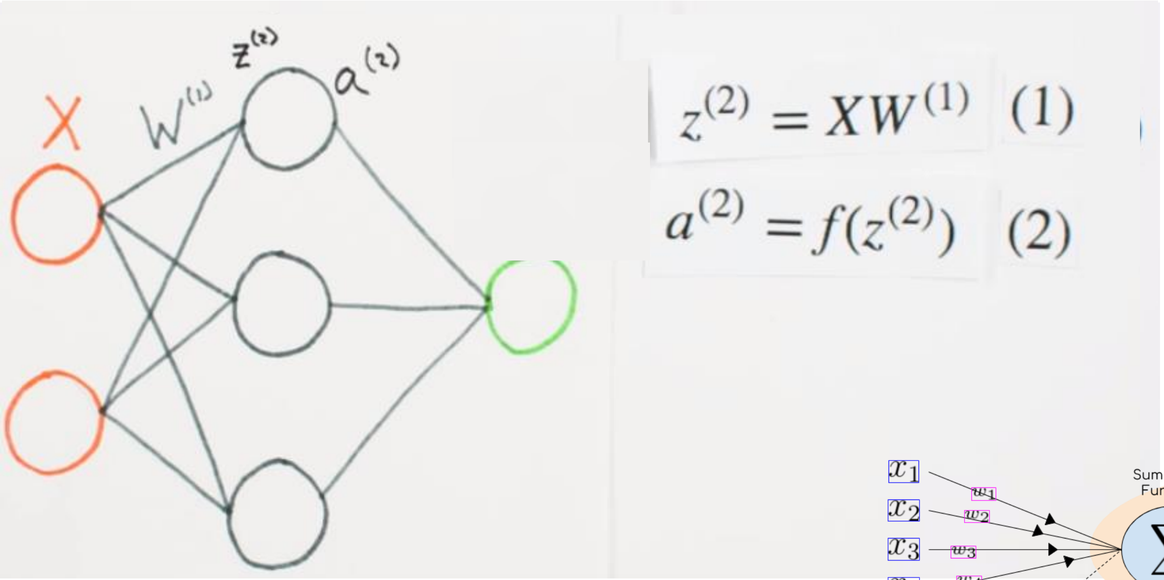


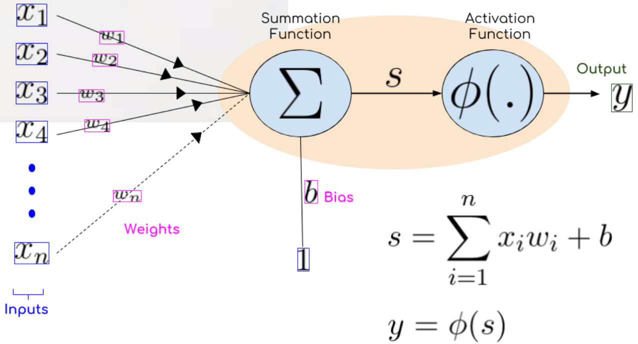
# Multi-Neuron Networks :: FORWARD PROPAGATION



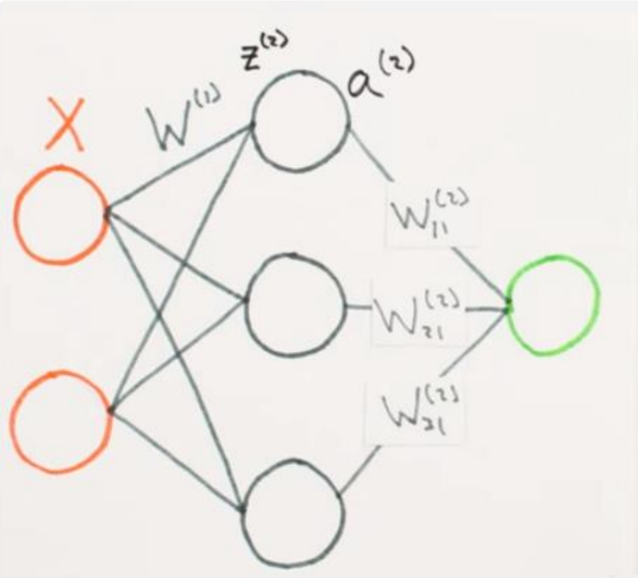

$$z^{(2)} = XW^{(1)} \quad (1)$$

# Multi-Neuron Networks :: Forward Propagation



# Multi-Neuron Networks :: FORWARD PROPAGATION

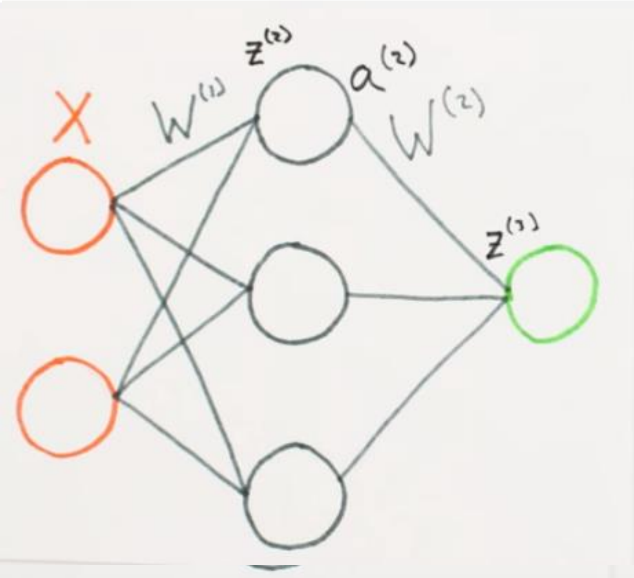



$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

# Multi-Neuron Networks :: Forward Propagation

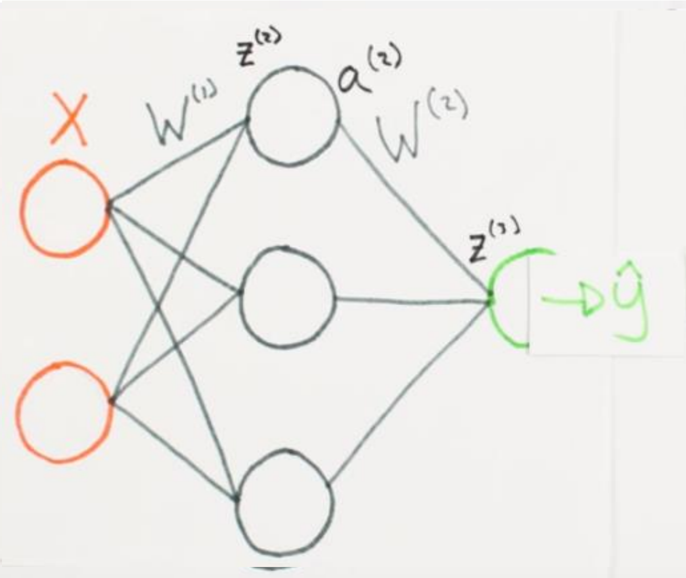



$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

# Multi-Neuron Networks :: FORWARD PROPAGATION




$$z^{(2)} = XW^{(1)} \quad (1)$$

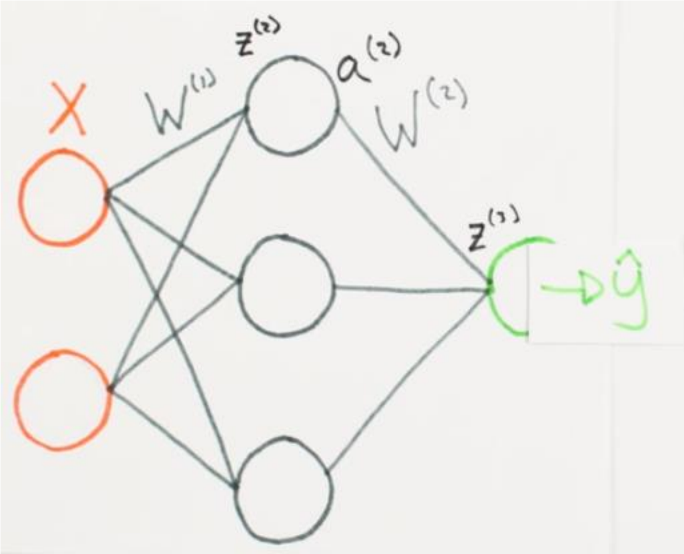
$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

# Multi-Neuron Networks :: Forward Propagation

```
In [1]: class Neural_Network(object):
            def __init__(self):
                #Define Hyperparameters
                self.inputLayerSize = 2
                self.outputLayerSize = 1
                self.hiddenLayerSize = 3

                #Weights (Parameters)
                self.W1 = np.random.randn(self.inputLayerSize, \
                                          self.hiddenLayerSize)
                self.W2 = np.random.randn(self.hiddenLayerSize, \
                                          self.outputLayerSize)

            def forward(self, X):
                #Propagate inputs though network
                self.z2 = np.dot(X, self.W1)
                self.a2 = self.sigmoid(self.z2)
                self.z3 = np.dot(self.a2, self.W2)
                yHat = self.sigmoid(self.z3)
                return yHat

            def sigmoid(self, z):
                #Apply sigmoid activation function to scalar, vector, or
                return 1/(1+np.exp(-z))
```




$$z^{(2)} = XW^{(1)} \quad (1)$$

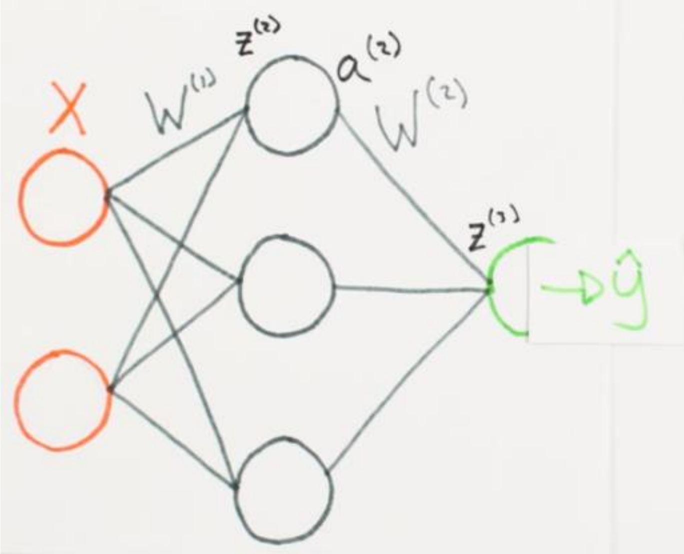
$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

# Multi-Neuron Networks :: Forward Propagation

```
In [7]: NN = Neural_Network()

In [8]: yHat = NN.forward(X)

In [9]: yHat
Out[9]: array([[ 0.59470263],
               [ 0.58177822],
               [ 0.50641742]])

n [10]: y
ut[10]: array([[ 0.75],
               [ 0.82],
               [ 0.93]])
```



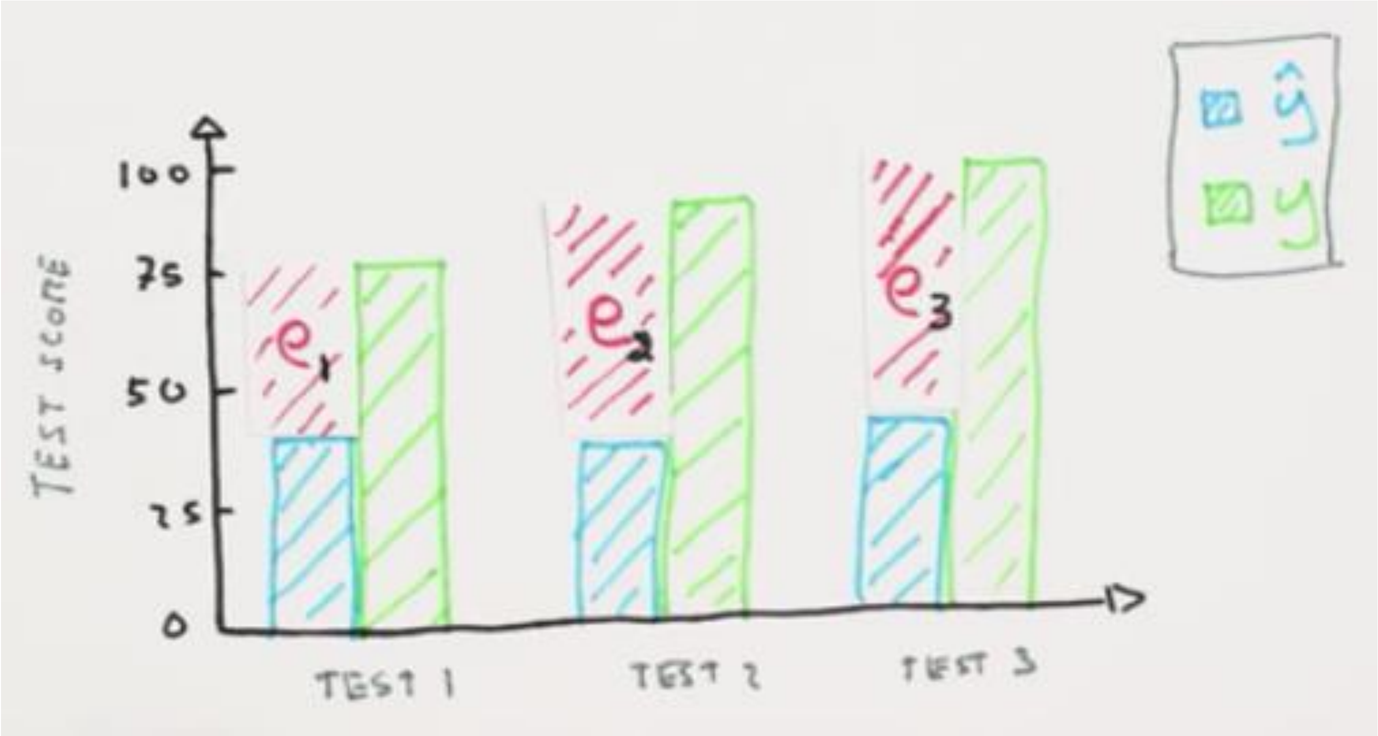
$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

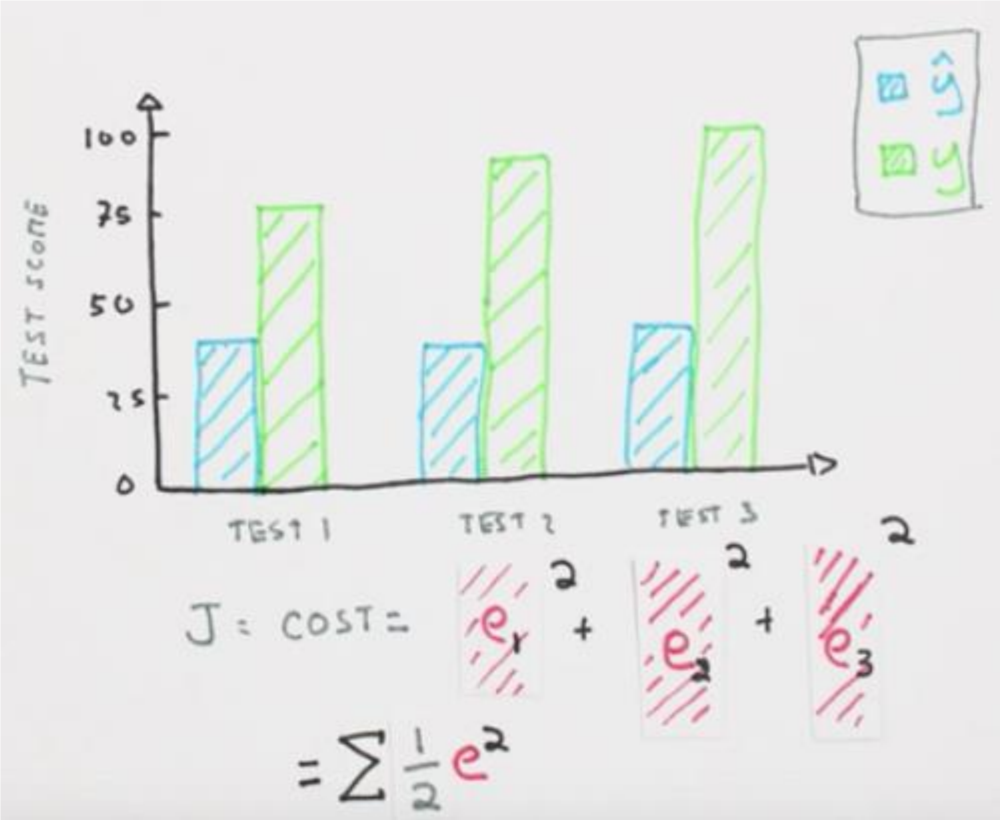
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

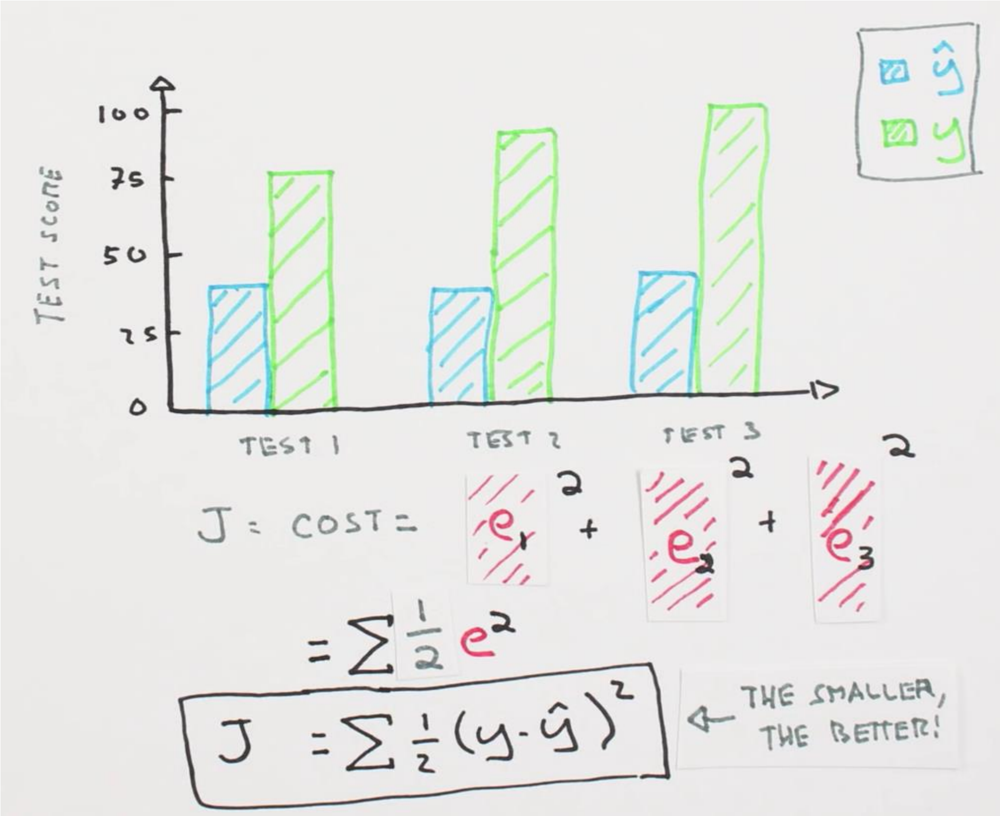
# Multi-Neuron Networks :: Gradient Descent



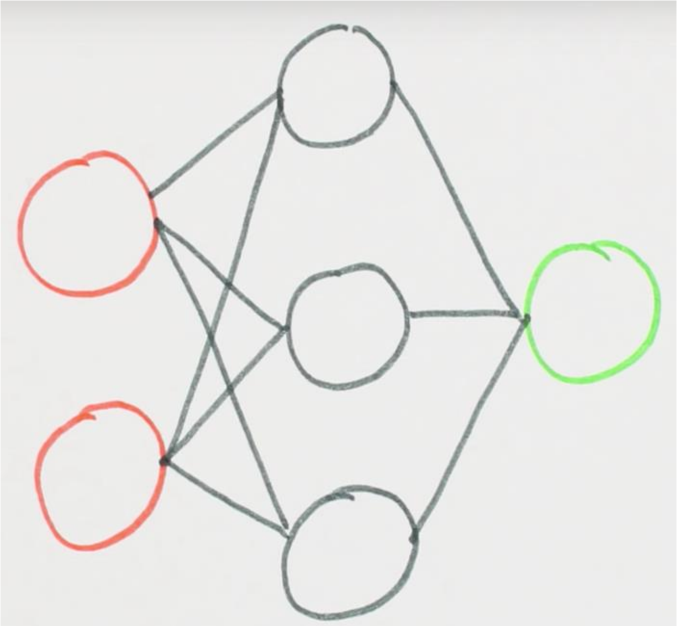
# Multi-Neuron Networks :: Gradient Descent

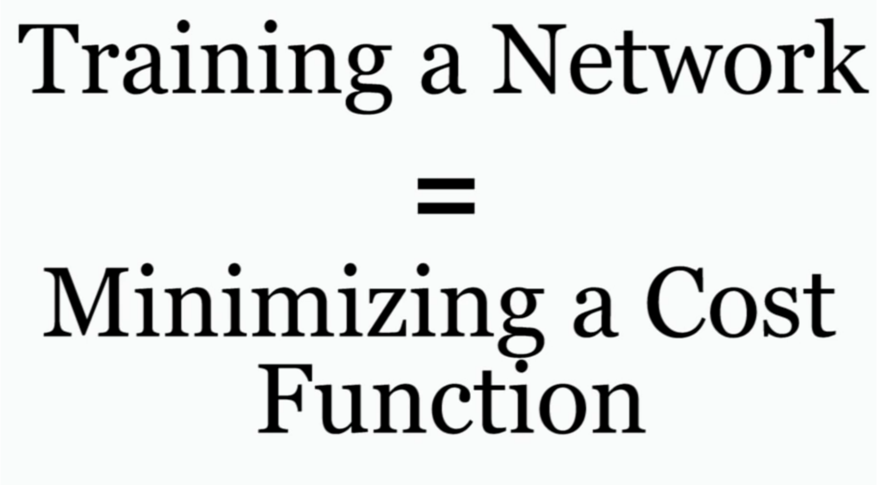


# Multi-Neuron Networks :: Gradient Descent

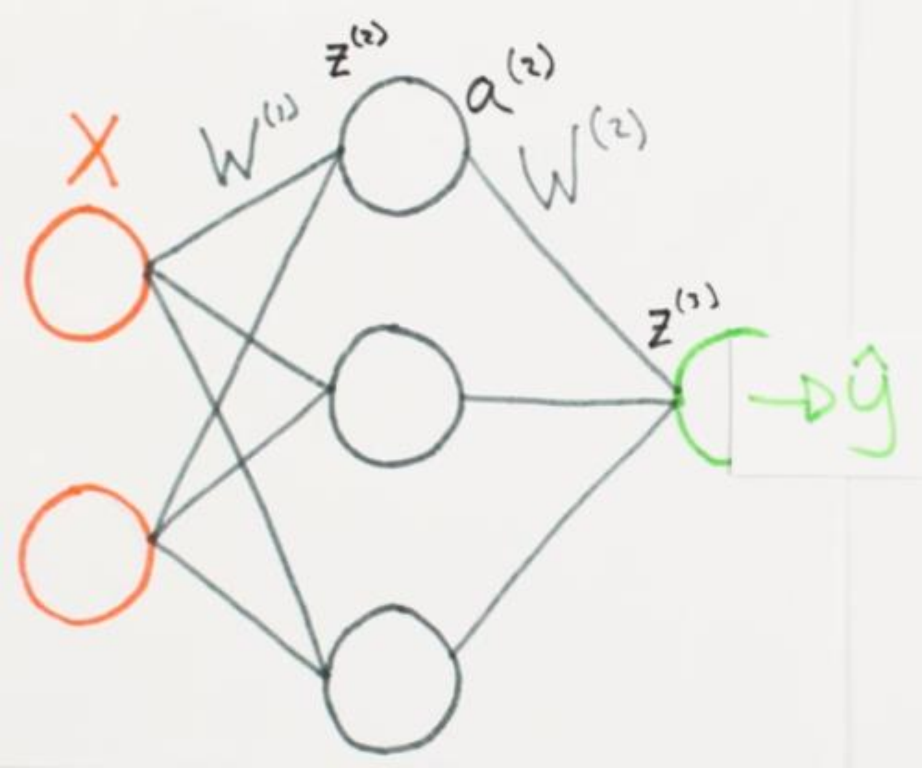


# Multi-Neuron Networks :: Gradient Descent





# Multi-Neuron Networks :: Gradient Descent




$$z^{(2)} = XW^{(1)} \tag{1}$$

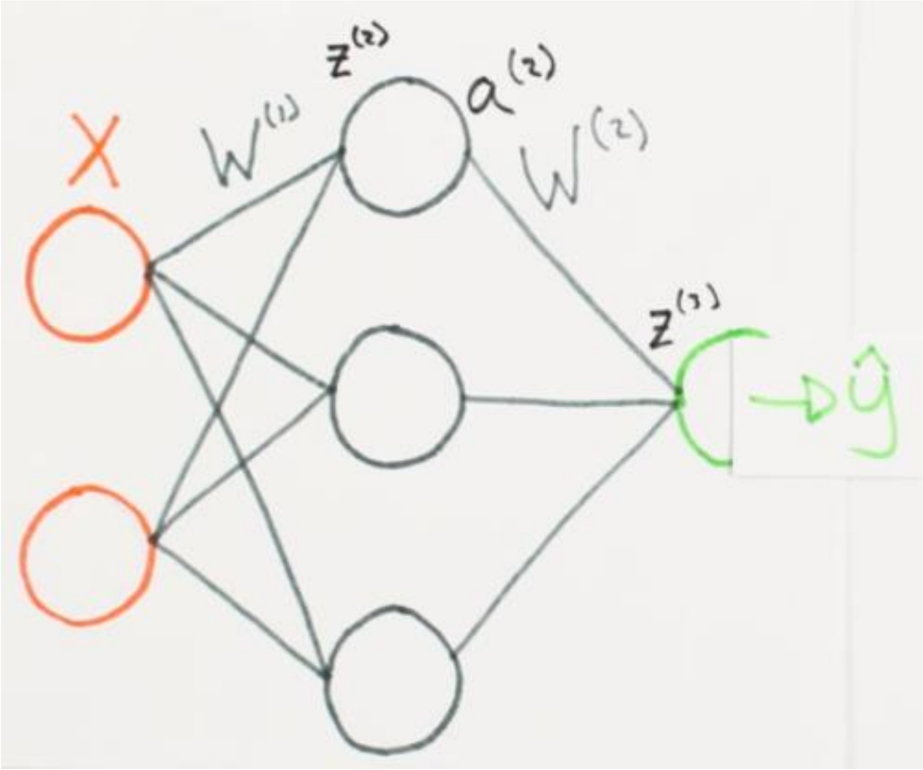
$$a^{(2)} = f(z^{(2)}) \tag{2}$$

$$z^{(3)} = a^{(2)}W^{(2)} \tag{3}$$

$$\hat{y} = f(z^{(3)}) \tag{4}$$

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

# Multi-Neuron Networks :: Gradient Descent




$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

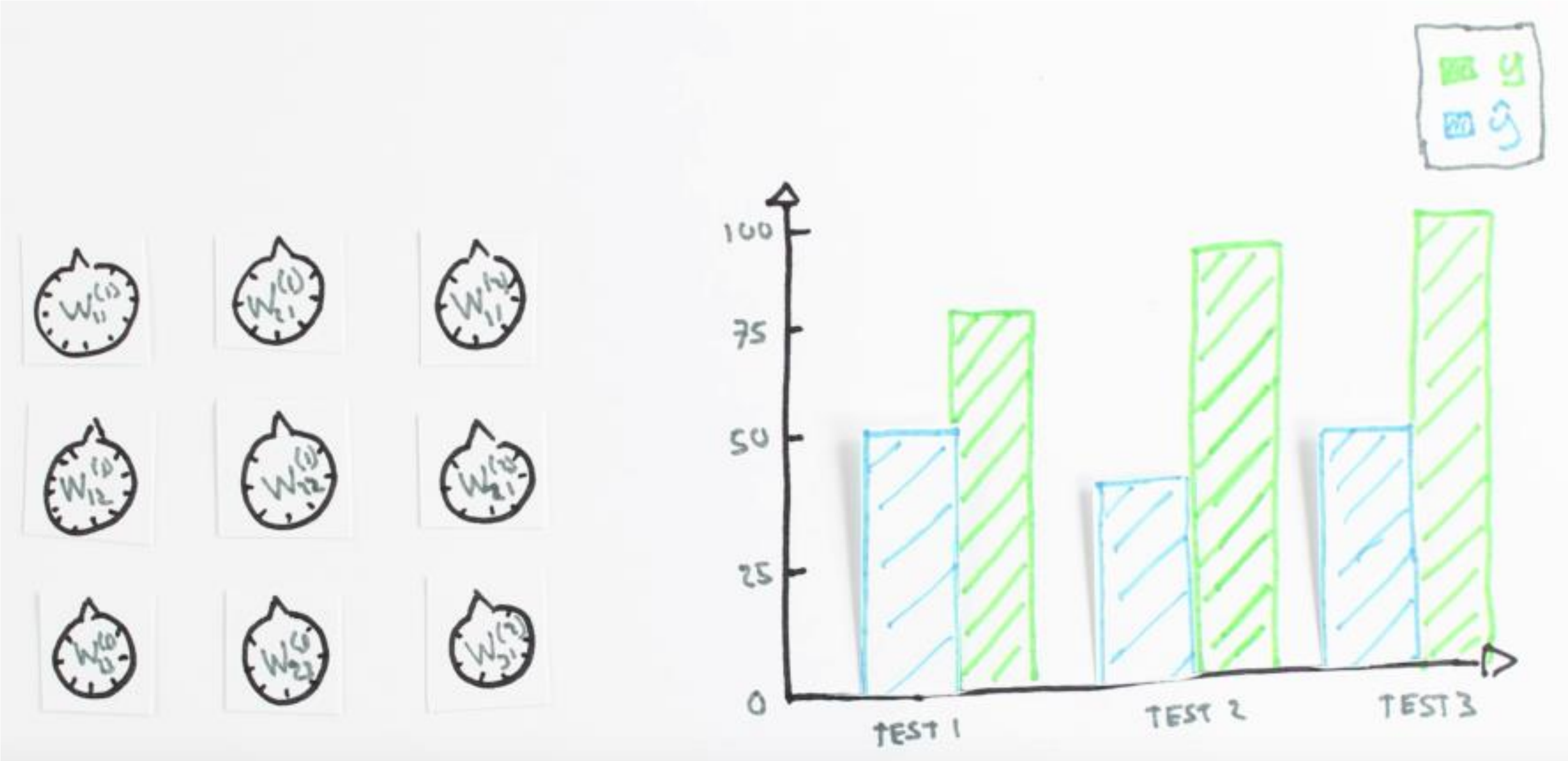
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \quad (5)$$

$$\boxed{J = \sum \frac{1}{2}\left(y - f\left(f\left(XW^{(1)}\right)W^{(2)}\right)\right)^2}$$

# Multi-Neuron Networks :: Gradient Descent



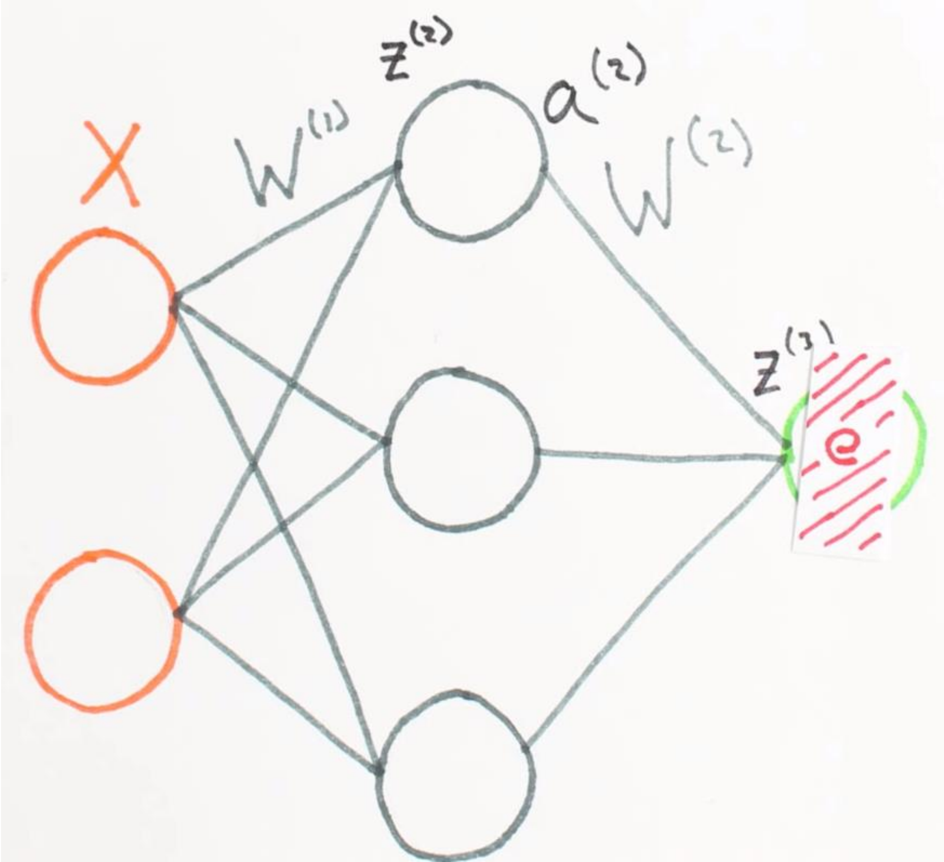
# Multi-Neuron Networks :: training

Initialize network with random weights

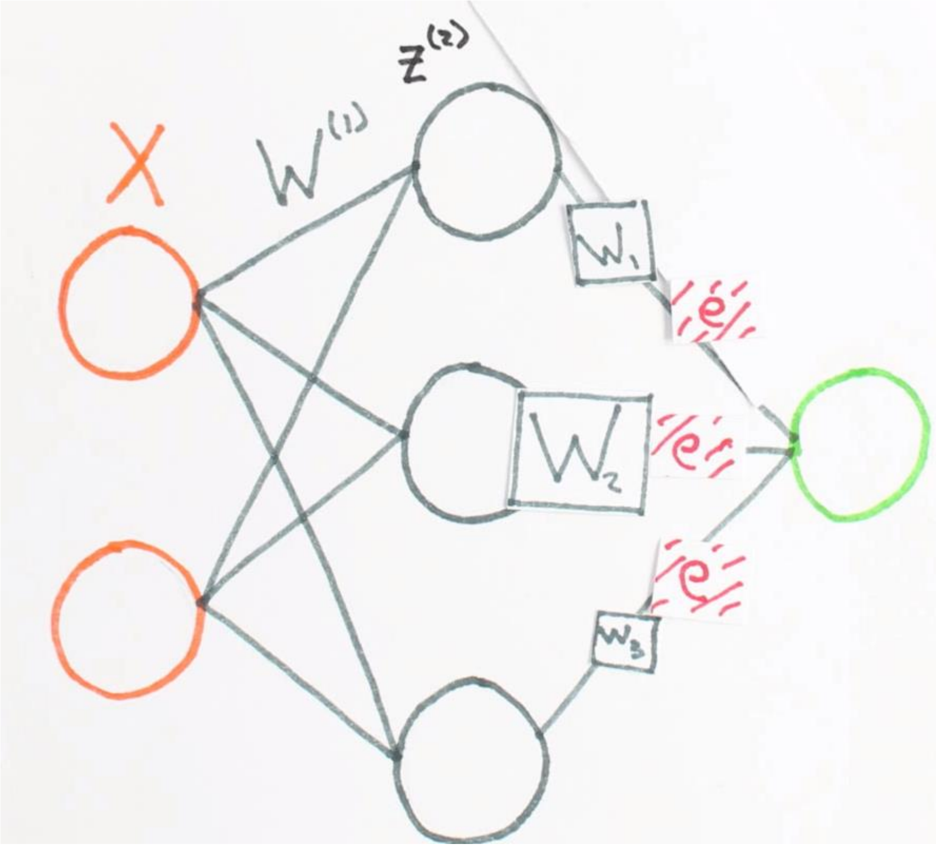
While [not converged]

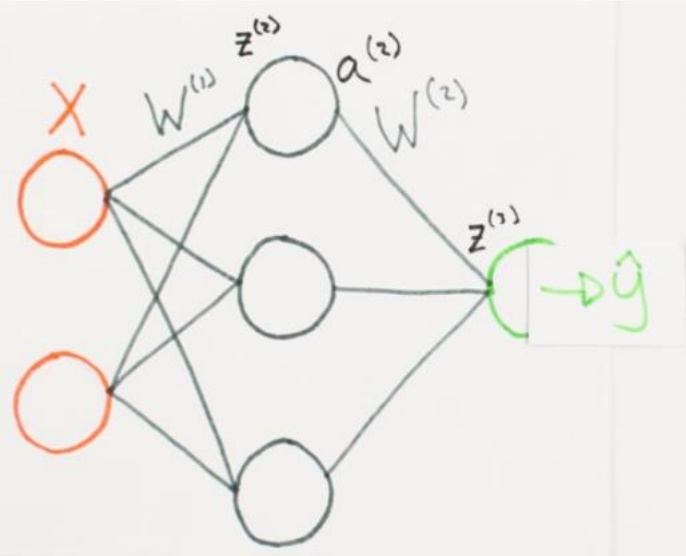
- Do forward prop
- Do backprop and determine change in weights
- Update all weights in all layers

# Multi-Neuron Networks :: BackPropagation



# Multi-Neuron Networks :: BackPropagation





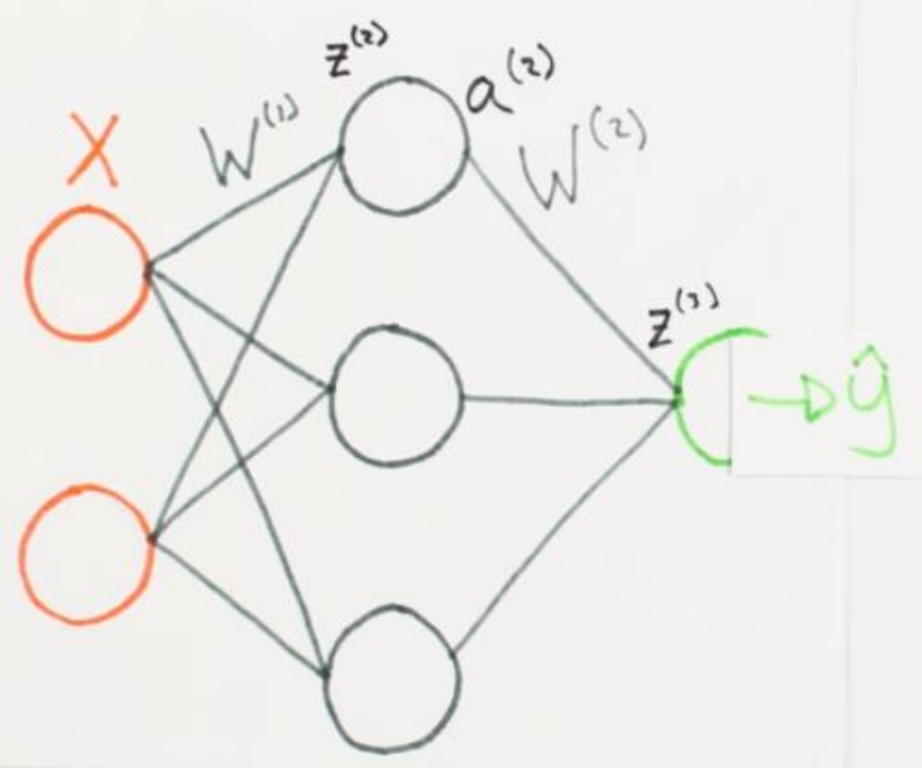

$$z^{(2)} = XW^{(1)} \quad (1)$$

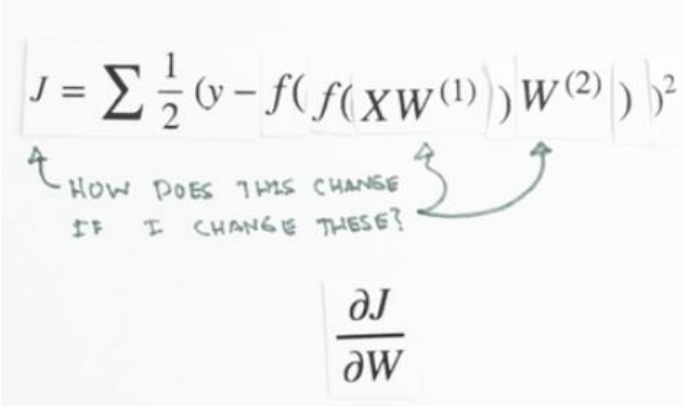
$$a^{(2)} = f(z^{(2)}) \quad (2)$$

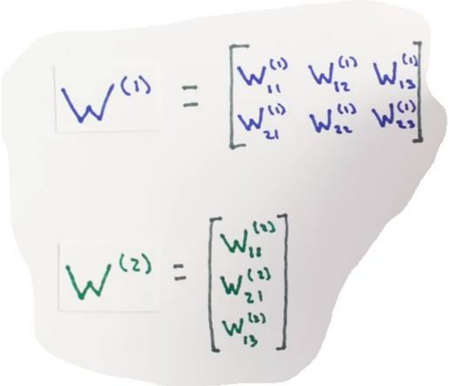
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

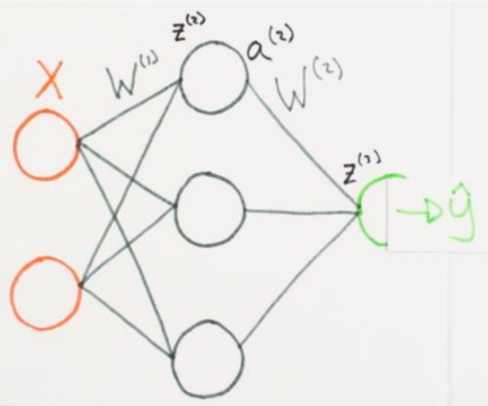
# Multi-Neuron Networks :: Backpropagation

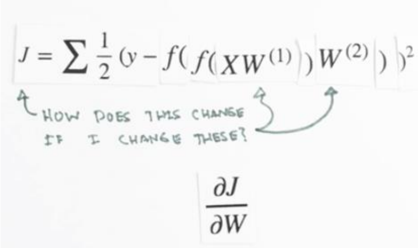


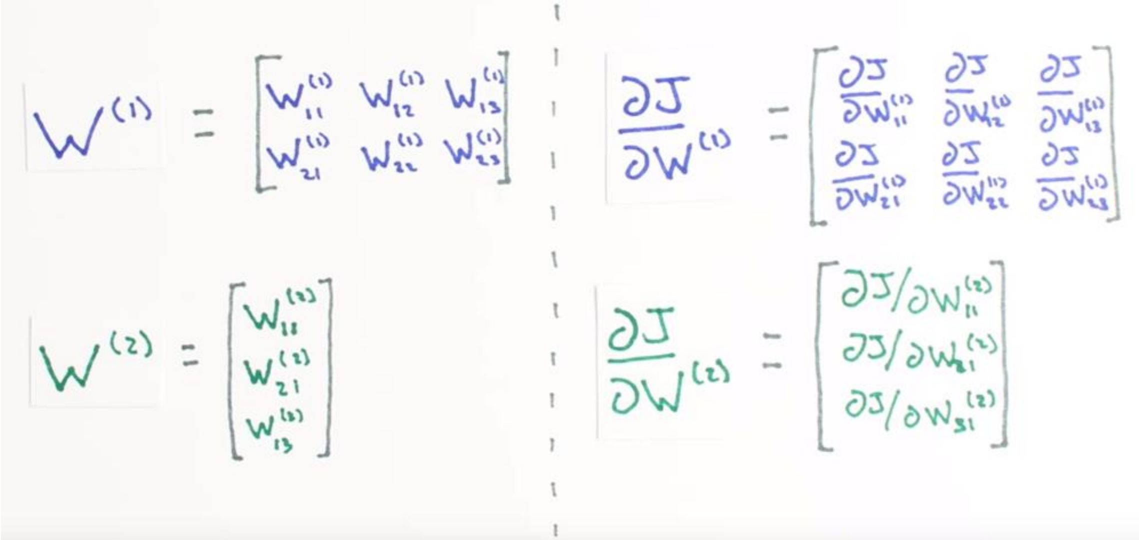



# Multi-Neuron Networks :: Backpropagation



# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

Adds cost from each example

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

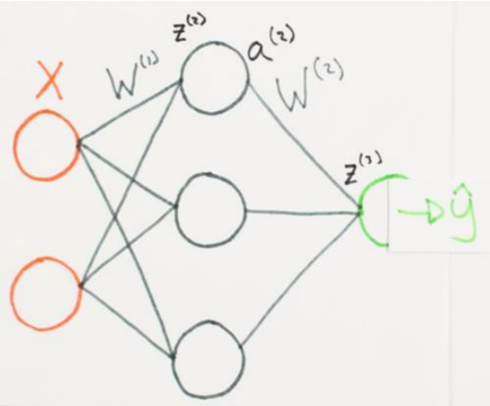
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

How does this change if I change these?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

Adds cost from each example

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$z^{(2)} = XW^{(1)} \qquad (1)$$

$$a^{(2)} = f(z^{(2)}) \qquad (2)$$

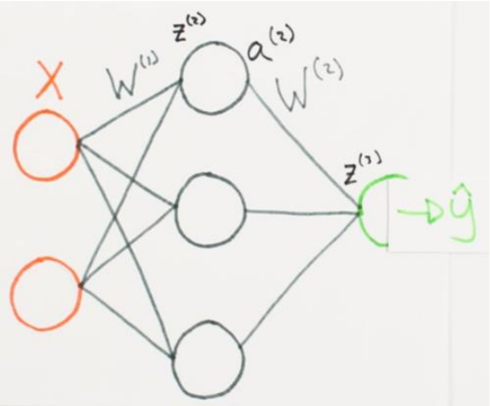
$$z^{(3)} = a^{(2)}W^{(2)} \qquad (3)$$

$$\hat{y} = f(z^{(3)}) \qquad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

How does this change if I change these?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y-\hat{y})^2$$

ADDS COST FROM EACH EXAMPLE

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y-\hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial W^{(2)}}$$

$$\boxed{\frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial W^{(2)}}}$$

$$z^{(2)} = XW^{(1)} \qquad (1)$$

$$a^{(2)} = f(z^{(2)}) \qquad (2)$$

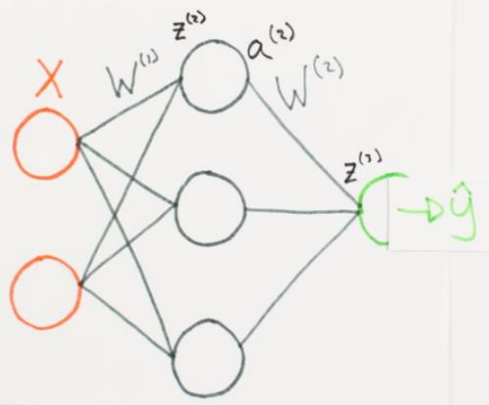
$$z^{(3)} = a^{(2)}W^{(2)} \qquad (3)$$

$$\hat{y} = f(z^{(3)}) \qquad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

Adds cost from each example

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

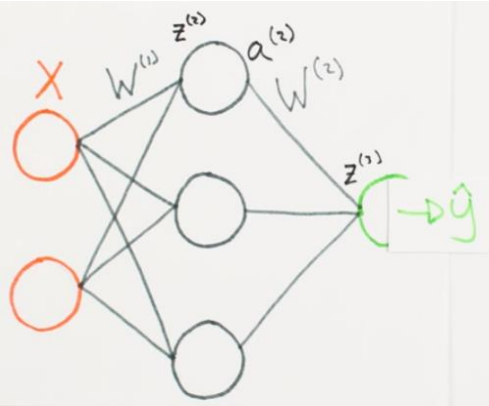
$$\boxed{\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial W^{(2)}}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$
$$a^{(2)} = f(z^{(2)}) \quad (2)$$
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$
$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

How does this change if I change these?

$$\frac{\partial J}{\partial W}$$




CHAIN RULE

ex $\frac{d}{dx}(3x + 2x^2)^2 = 2(3x + 2x^2)(3 + 6x)$

$$\boxed{\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}}$$

$$W^{(1)} = \begin{bmatrix} W^{(1)}_{11} & W^{(1)}_{12} & W^{(1)}_{13} \\ W^{(1)}_{21} & W^{(1)}_{22} & W^{(1)}_{23} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W^{(1)}_{11}} & \frac{\partial J}{\partial W^{(1)}_{12}} & \frac{\partial J}{\partial W^{(1)}_{13}} \\ \frac{\partial J}{\partial W^{(1)}_{21}} & \frac{\partial J}{\partial W^{(1)}_{22}} & \frac{\partial J}{\partial W^{(1)}_{23}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W^{(2)}_{11} \\ W^{(2)}_{21} \\ W^{(2)}_{13} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W^{(2)}_{11} \\ \partial J/\partial W^{(2)}_{21} \\ \partial J/\partial W^{(2)}_{31} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

ADDS COST FROM EACH EXAMPLE

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial W^{(2)}}$$

$$\hat{y} = f(z^{(3)})$$

$$\boxed{\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(2)}}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

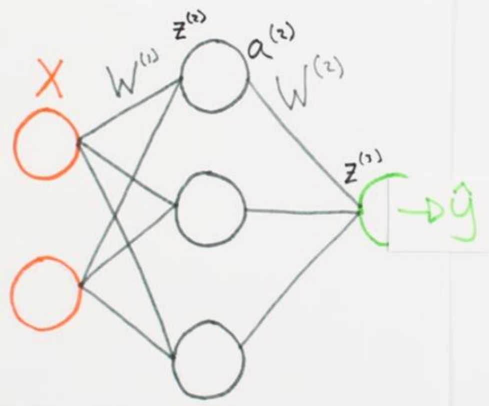
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




CHAIN RULE

ex $\frac{d}{dx}(3x + 2x^2)^2 = 2(3x + 2x^2)(3 + 6x)$

$$\boxed{\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

ADDS COST FROM EACH EXAMPLE

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial W^{(2)}}$$

$$\hat{y} = f(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(2)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

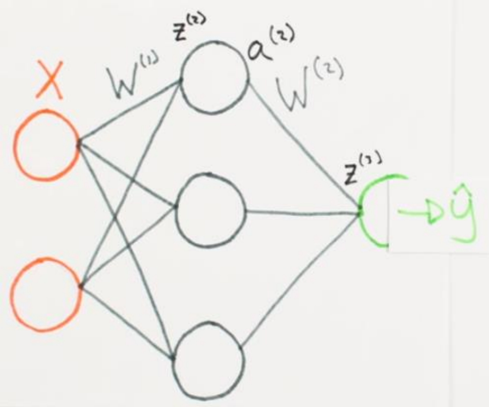
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

Adds cost from each example

$$\frac{\partial J}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = \sum \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial W^{(2)}}$$

$$\hat{y} = f(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = -(y - \hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(2)}}$$

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)} \qquad (6)$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

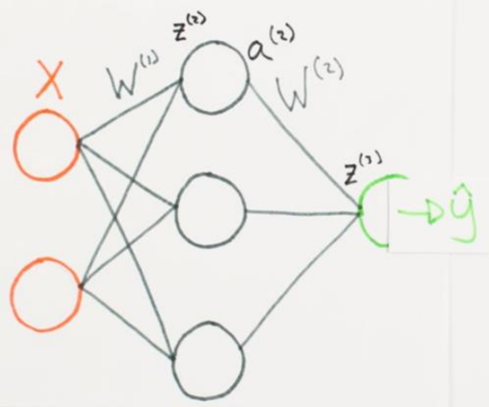
Backprop error

$$z^{(2)} = XW^{(1)} \qquad (1)$$
$$a^{(2)} = f(z^{(2)}) \qquad (2)$$
$$z^{(3)} = a^{(2)}W^{(2)} \qquad (3)$$
$$\hat{y} = f(z^{(3)}) \qquad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

How does this change if I change these?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

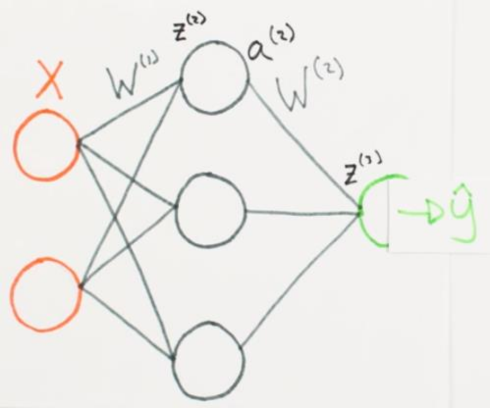
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J / \partial W_{11}^{(2)} \\ \partial J / \partial W_{21}^{(2)} \\ \partial J / \partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y}) f'(z^{(3)}) \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

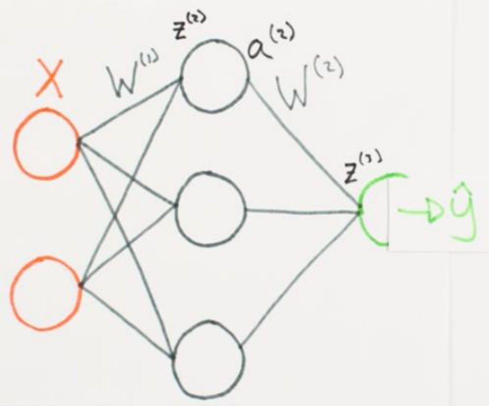
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\delta^{(3)} = -(y - \hat{y})f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

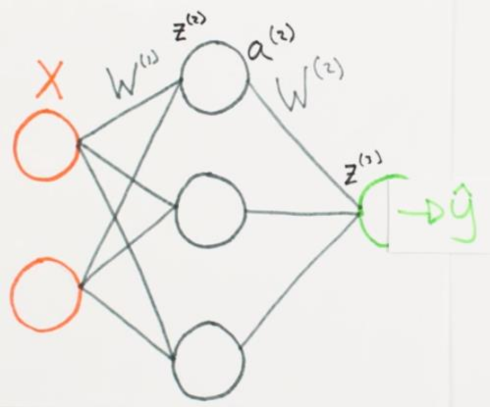
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y-\hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y-\hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y-\hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\delta^{(3)} = -(y-\hat{y})f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

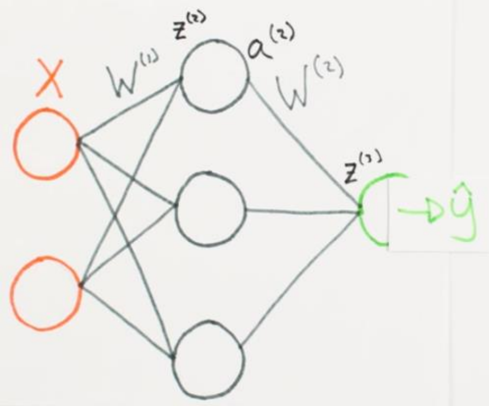
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y}) \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y}) f'(z^{(3)}) \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)} \frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)} \frac{\partial z^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)} (W^{(2)})^T \frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

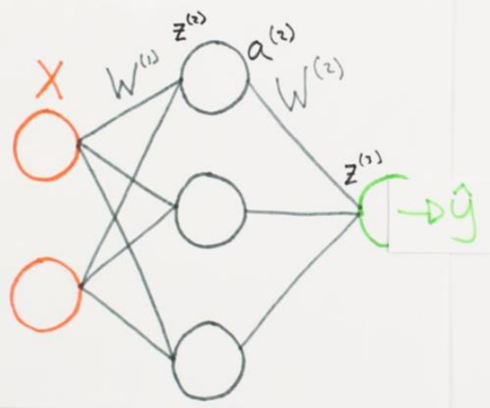
$$z^{(3)} = a^{(2)} W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix}$$

$$\frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix}$$

$$\frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J / \partial W_{11}^{(2)} \\ \partial J / \partial W_{21}^{(2)} \\ \partial J / \partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T\frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T f'(z^{(2)})\frac{\partial z^{(2)}}{\partial W^{(1)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

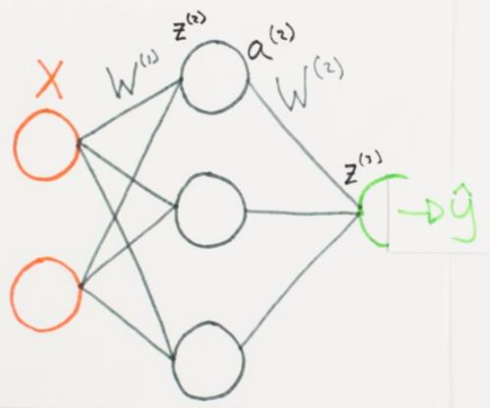
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W^{(1)}_{11} & W^{(1)}_{12} & W^{(1)}_{13} \\ W^{(1)}_{21} & W^{(1)}_{22} & W^{(1)}_{23} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W^{(1)}_{11}} & \frac{\partial J}{\partial W^{(1)}_{12}} & \frac{\partial J}{\partial W^{(1)}_{13}} \\ \frac{\partial J}{\partial W^{(1)}_{21}} & \frac{\partial J}{\partial W^{(1)}_{22}} & \frac{\partial J}{\partial W^{(1)}_{23}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W^{(2)}_{11} \\ W^{(2)}_{21} \\ W^{(2)}_{13} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W^{(2)}_{11} \\ \partial J/\partial W^{(2)}_{21} \\ \partial J/\partial W^{(2)}_{31} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(1)}} = \frac{\partial \sum \frac{1}{2}(y - \hat{y})^2}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = -(y - \hat{y})f'(z^{(3)})\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T\frac{\partial a^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = \delta^{(3)}(W^{(2)})^T f'(z^{(2)})\frac{\partial z^{(2)}}{\partial W^{(1)}}$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T\overbrace{\delta^{(3)}(W^{(2)})^T f'(z^{(2)})}^{\delta^{(2)}}$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

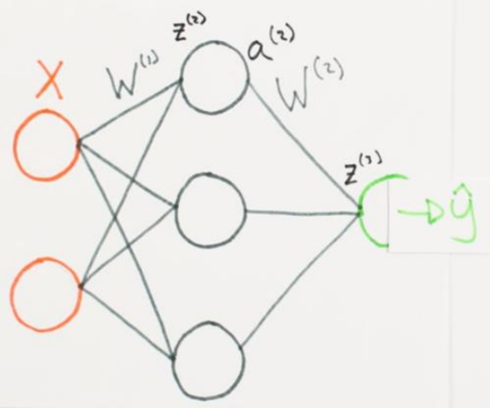
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{31}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y}) f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

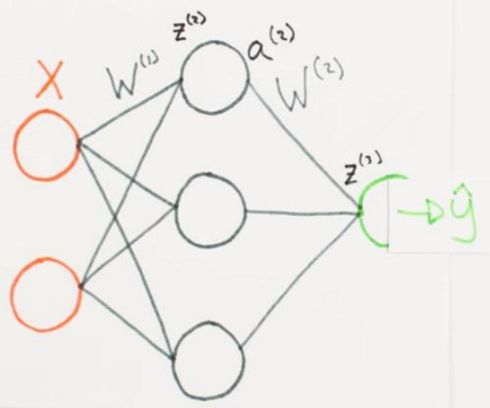
$$z^{(3)} = a^{(2)} W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J / \partial W_{11}^{(2)} \\ \partial J / \partial W_{21}^{(2)} \\ \partial J / \partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y}) f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

```
In [2]: NN = Neural_Network()

In [3]: cost1 = NN.costFunction(X,y)

In [4]: dJdW1, dJdW2 = NN.costFunctionPrime(X,y)

In [5]: dJdW1

Out[5]: array([[-0.0071096 , -0.01059837, -0.00094283],
               [-0.00172302, -0.00234379, -0.00019984]])

In [6]: dJdW2

Out[6]: array([[-0.0229961 ],
               [-0.01631712],
               [-0.02079302]])

In [7]: scalar = 3
        NN.W1 = NN.W1 + scalar*dJdW1
        NN.W2 = NN.W2 + scalar*dJdW2
        cost2 = NN.costFunction(X,y)

In [8]: print cost1, cost2

        [ 0.01906658] [ 0.02396064]

In [9]: dJdW1, dJdW2 = NN.costFunctionPrime(X,y)
        NN.W1 = NN.W1 - scalar*dJdW1
        NN.W2 = NN.W2 - scalar*dJdW2
        cost3 = NN.costFunction(X,y)

In [10]: print cost2, cost3

        [ 0.02396064] [ 0.01773225]
```

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

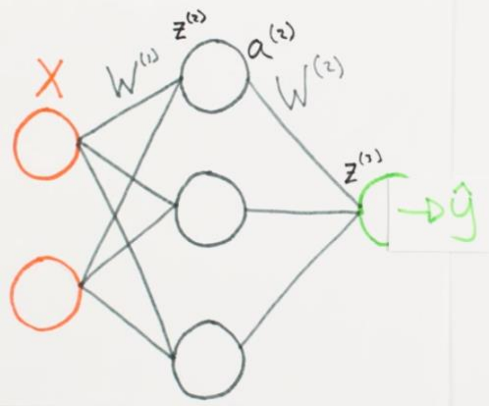
$$z^{(3)} = a^{(2)} W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J/\partial W_{11}^{(2)} \\ \partial J/\partial W_{21}^{(2)} \\ \partial J/\partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: training

Initialize network with random weights

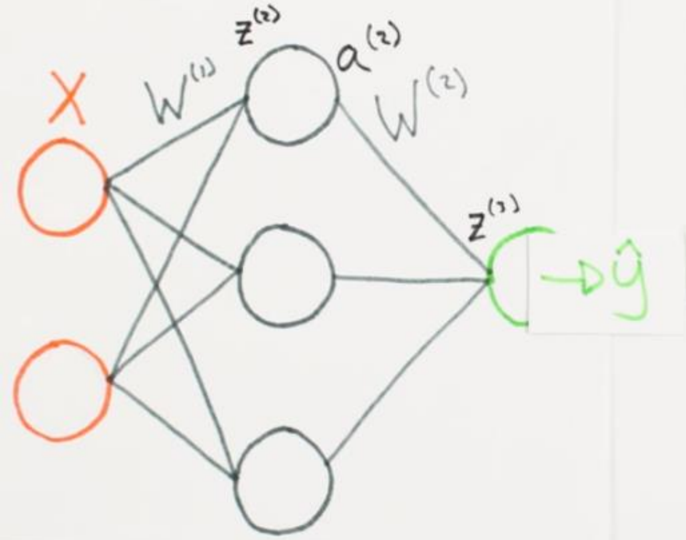
While [not converged]

Do forward prop

Do backprop and determine change in weights

Update all weights in all layers

One Iteration




$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

# Multi-Neuron Networks :: training

Initialize network with random weights

While [not converged]

Do forward prop

Do backprop and determine change in weights

Update all weights in all layers

One Iteration

# Multi-Neuron Networks :: Backpropagation

$$J = \sum \frac{1}{2}(y - \hat{y})^2$$

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

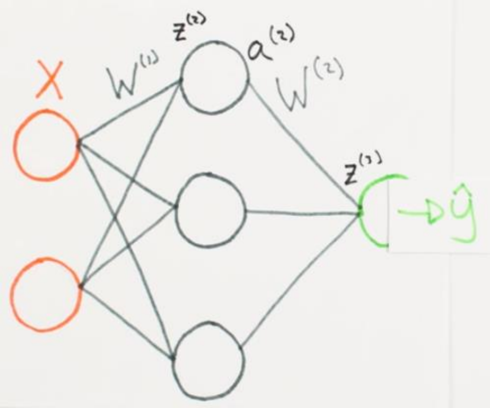
$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2}(y - f(f(XW^{(1)})W^{(2)}))^2$$

HOW DOES THIS CHANGE IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$




$$\frac{\partial J}{\partial W^{(2)}} = (a^{(2)})^T \delta^{(3)}$$

$$\delta^{(3)} = -(y - \hat{y}) f'(z^{(3)})$$

$$\frac{\partial J}{\partial W^{(1)}} = X^T \delta^{(2)}$$

$$\delta^{(2)} = \delta^{(3)} (W^{(2)})^T f'(z^{(2)})$$

$$W^{(1)} = \begin{bmatrix} W_{11}^{(1)} & W_{12}^{(1)} & W_{13}^{(1)} \\ W_{21}^{(1)} & W_{22}^{(1)} & W_{23}^{(1)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial J}{\partial W_{11}^{(1)}} & \frac{\partial J}{\partial W_{12}^{(1)}} & \frac{\partial J}{\partial W_{13}^{(1)}} \\ \frac{\partial J}{\partial W_{21}^{(1)}} & \frac{\partial J}{\partial W_{22}^{(1)}} & \frac{\partial J}{\partial W_{23}^{(1)}} \end{bmatrix}$$

$$W^{(2)} = \begin{bmatrix} W_{11}^{(2)} \\ W_{21}^{(2)} \\ W_{13}^{(2)} \end{bmatrix} \qquad \frac{\partial J}{\partial W^{(2)}} = \begin{bmatrix} \partial J / \partial W_{11}^{(2)} \\ \partial J / \partial W_{21}^{(2)} \\ \partial J / \partial W_{31}^{(2)} \end{bmatrix}$$

# Multi-Neuron Networks :: training

Initialize network with random weights

While [not converged]

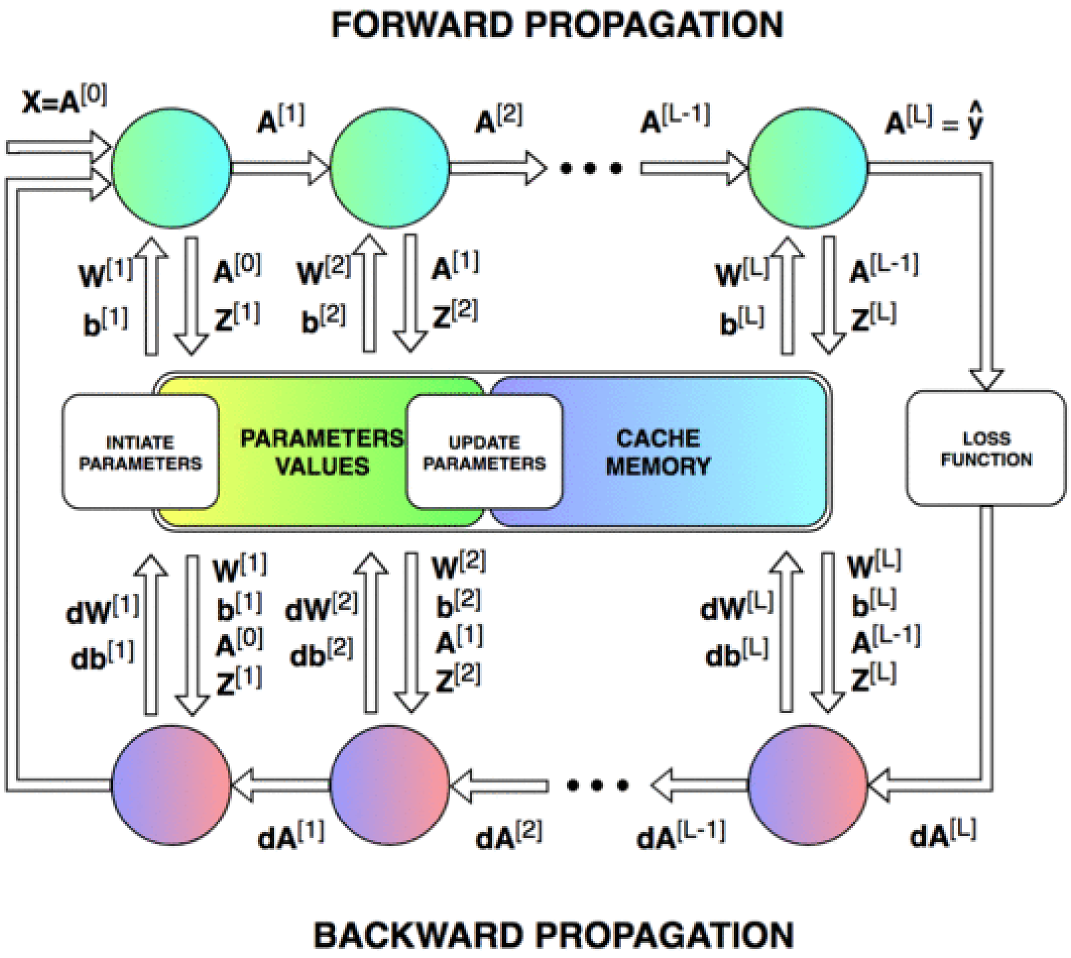
Do forward prop

Do backprop and determine change in weights

Update all weights in all layers

$$\mathbf{w}^{(\mathbf{t+1})} = \mathbf{w}^{(\mathbf{t})} - \eta^{(t)} \nabla_{\mathbf{w}} \mathbf{J}(\mathbf{w})$$

One Iteration


FORWARD PROPAGATION
X=A[0]
A[1]
A[2]
A[L-1]
A[L] = ŷ
W[1] b[1]
A[0] Z[1]
W[2] b[2]
A[1] Z[2]
W[L] b[L]
A[L-1] Z[L]
INTIATE PARAMETERS
PARAMETERS VALUES
UPDATE PARAMETERS
CACHE MEMORY
LOSS FUNCTION
dW[1] db[1]
W[1] b[1] A[0] Z[1]
dW[2] db[2]
W[2] b[2] A[1] Z[2]
dW[L] db[L]
W[L] b[L] A[L-1] Z[L]
dA[1]
dA[2]
dA[L-1]
dA[L]
BACKWARD PROPAGATION

# MULTI-NEURON NETWORK — NUMERICAL GRADIENT CHECKING

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - F(x)}{\Delta x}$$

$$f(x) = x^2$$

# MULTI-NEURON NETWORK — NUMERICAL GRADIENT CHECKING

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - F(x)}{\Delta x}$$

$$f(x) = x^2$$

$$f'(x) = \lim_{\Delta x \to 0} \frac{(x+\Delta x)^2 - x^2}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{x^2 + 2x\Delta x + \Delta x^2 - x^2}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{\Delta x^2 + 2x\Delta x}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \Delta x + 2x = \boxed{2x}$$

# MULTI-NEURON NETWORK – NUMERICAL GRADIENT CHECKING

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - F(x)}{\Delta x}$$

$$f(x) = x^2$$

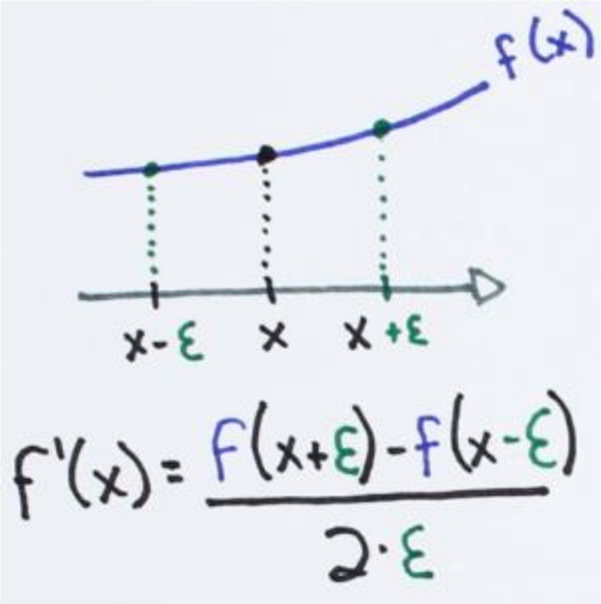
$$f'(x) = \lim_{\Delta x \to 0} \frac{(x+\Delta x)^2 - x^2}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{x^2 + 2x\Delta x + \Delta x^2 - x^2}{\Delta x}$$

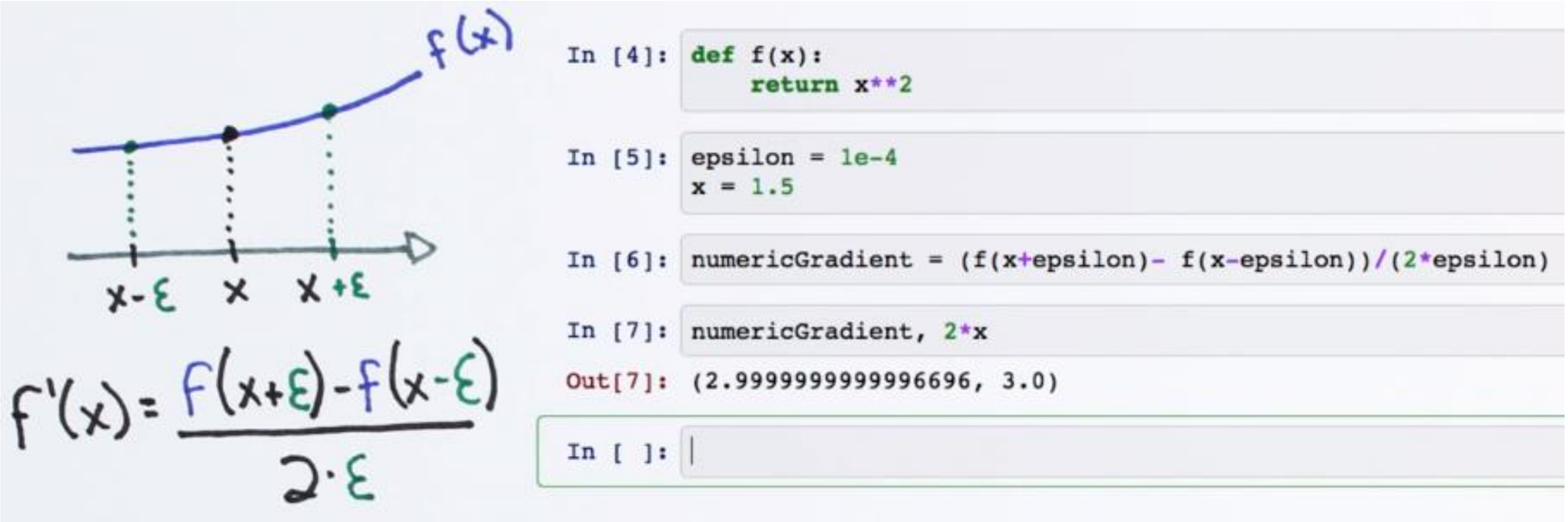
$$= \lim_{\Delta x \to 0} \frac{\Delta x^2 + 2x\Delta x}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \Delta x + 2x = \boxed{2x}$$

# MULTI-NEURON NETWORK – NUMERICAL GRADIENT CHECKING



# MULTI-NEURON NETWORK – NUMERICAL GRADIENT CHECKING



# MULTI-NEURON NETWORK – NUMERICAL GRADIENT CHECKING

**Parameter vector** $\theta$

$\theta \in \mathbb{R}^n$ (E.g. $\theta$ is "unrolled" version of $\Theta^{(1)}, \Theta^{(2)}, \Theta^{(3)}$)

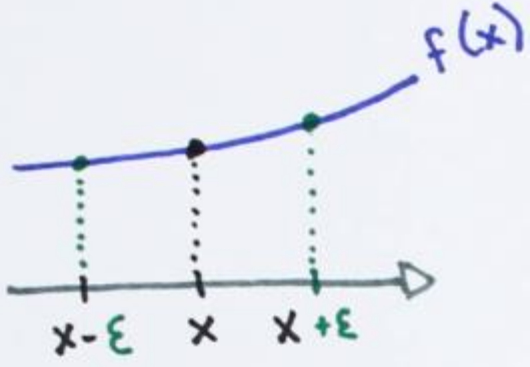
$\theta = [\theta_1, \theta_2, \theta_3, \ldots, \theta_n]$

$$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\boxed{\theta_1+\epsilon}, \theta_2, \theta_3, \ldots, \theta_n) - J(\boxed{\theta_1-\epsilon}, \theta_2, \theta_3, \ldots, \theta_n)}{2\epsilon}$$

$$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2+\epsilon, \theta_3, \ldots, \theta_n) - J(\theta_1, \theta_2-\epsilon, \theta_3, \ldots, \theta_n)}{2\epsilon}$$

$$\vdots$$

# MULTI-NEURON NETWORK – NUMERICAL GRADIENT CHECKING



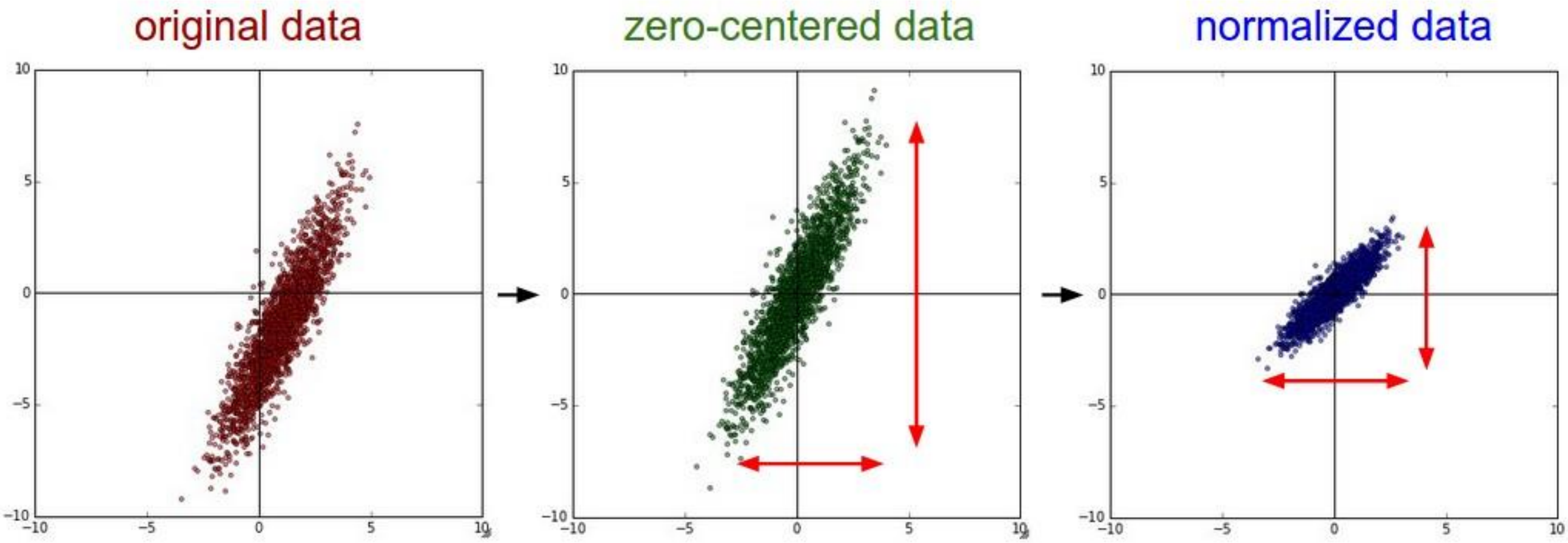

$$f'(x) = \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2 \cdot \varepsilon}$$

```
In [3]: numgrad = computeNumericalGradient(NN, X, y)

In [4]: grad = NN.computeGradients(X, y)

In [5]: numgrad

Out[5]: array([ -7.10752568e-03,  -6.30194392e-03,  -4.96392693e-03,
                -6.55946987e-04,   7.57595597e-05,  -1.11297012e-03,
                -8.81243102e-03,  -4.45550176e-03,  -1.93471143e-02])

In [6]: grad

Out[6]: array([ -7.10752569e-03,  -6.30194393e-03,  -4.96392693e-03,
                -6.55946985e-04,   7.57595604e-05,  -1.11297012e-03,
                -8.81243100e-03,  -4.45550176e-03,  -1.93471142e-02])
```

```
In [7]: norm(grad-numgrad)/norm(grad+numgrad)

Out[7]: 1.9824969610227768e-09
```

# Data Setup

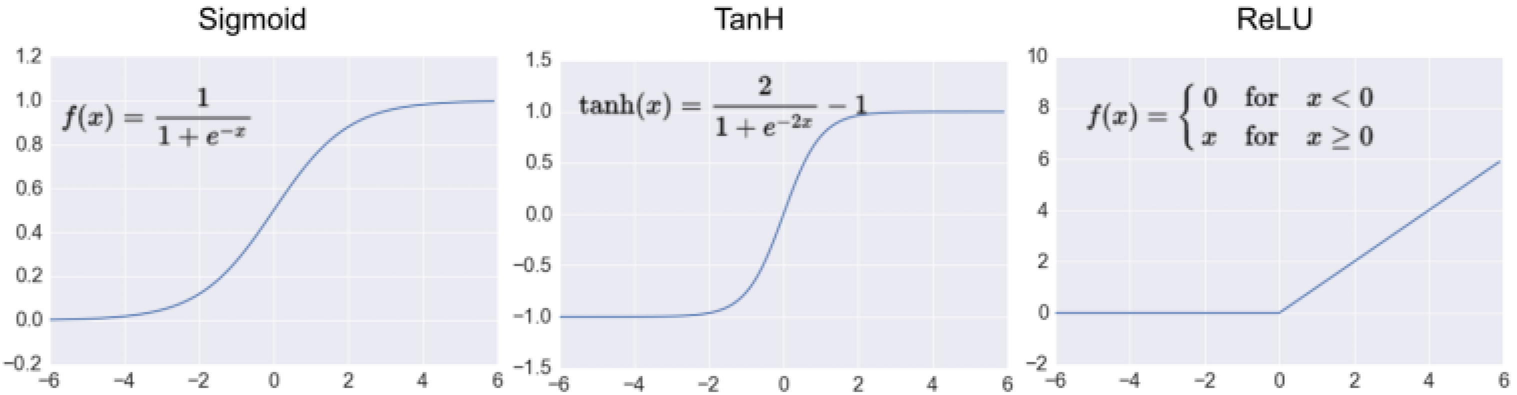
- Preprocessing:



# Weight initialization

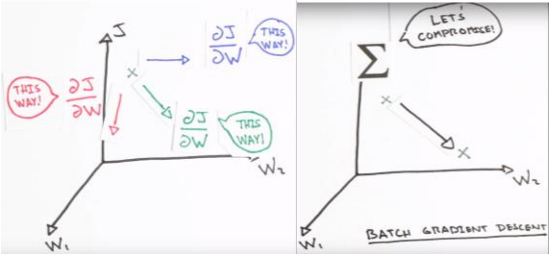
- All Zeros
- Random [0,1]
- Random [-1,1]
- `w = np.random.randn(n) * sqrt(2.0/n)`, n = # of inputs to neuron

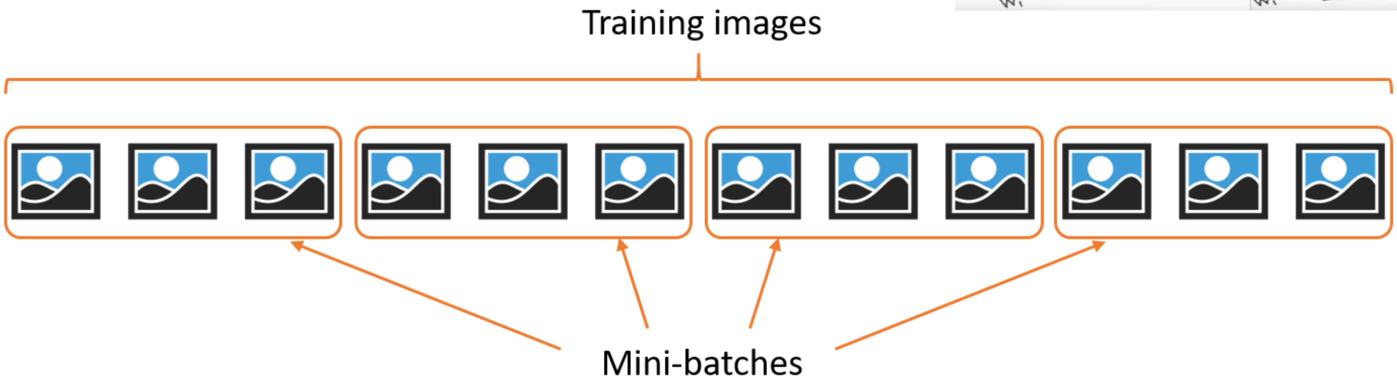
# Activation functions



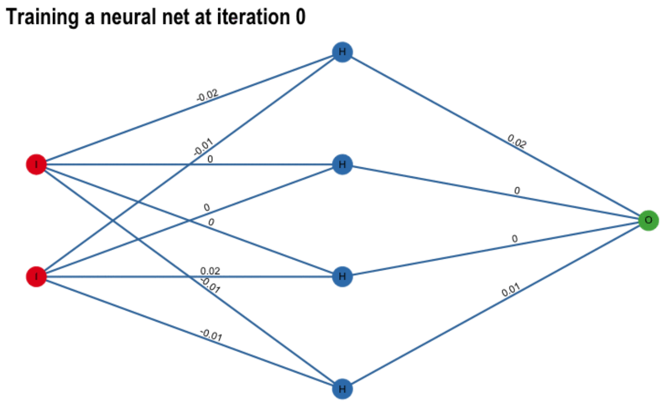
# MINIBATCH VS SINGLE

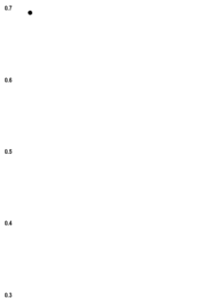
- Average error, gradients

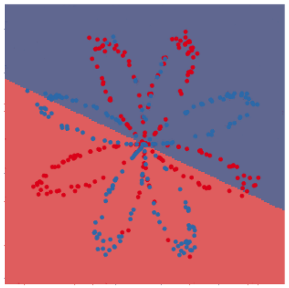


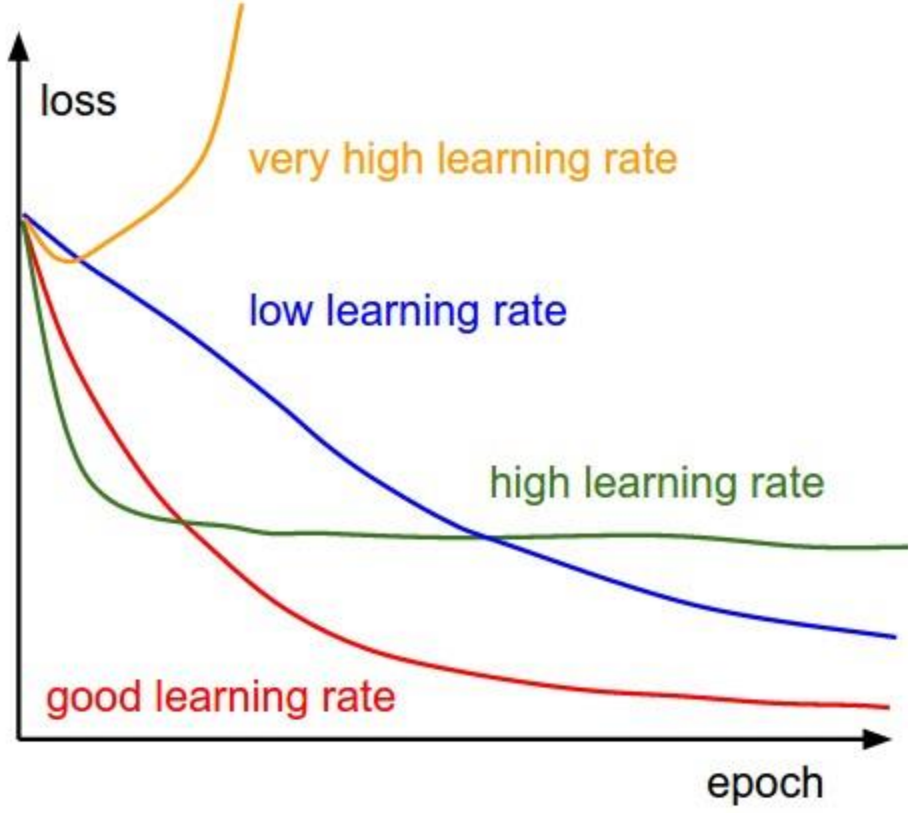

# Multi-Neuron Networks :: training

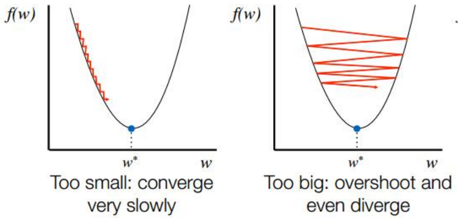


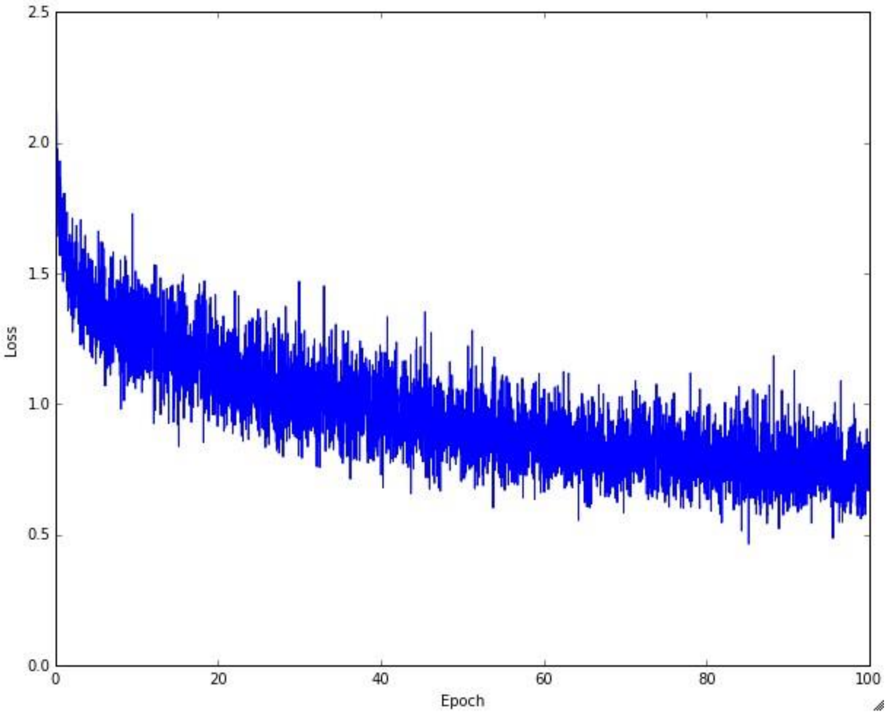



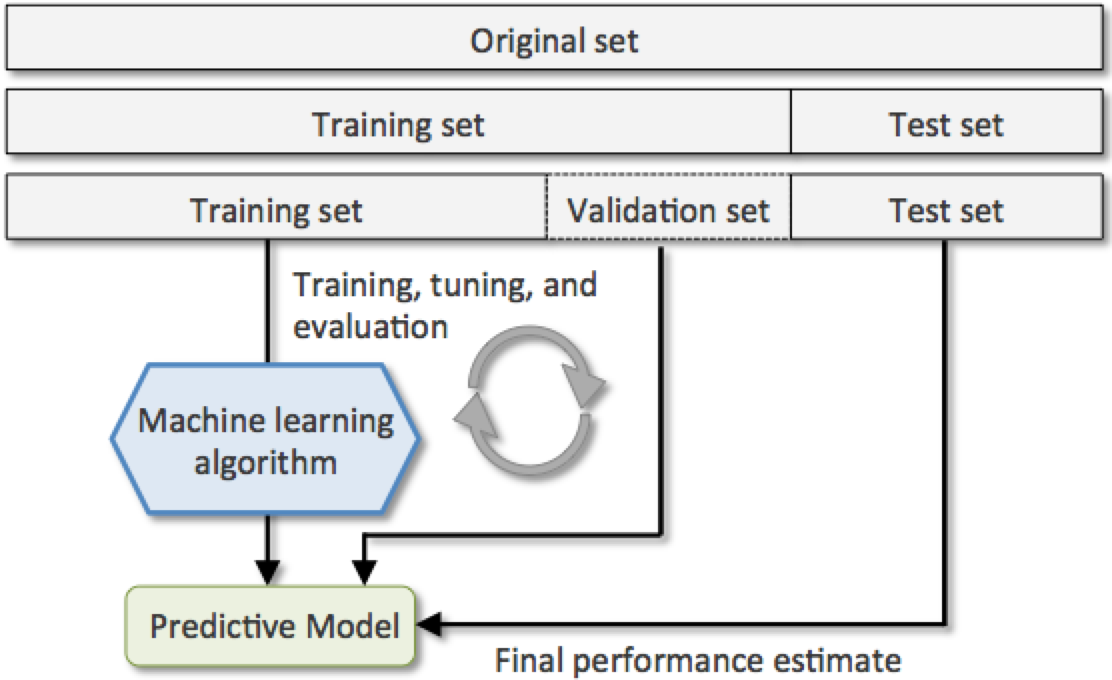



# Training – setting learning rate

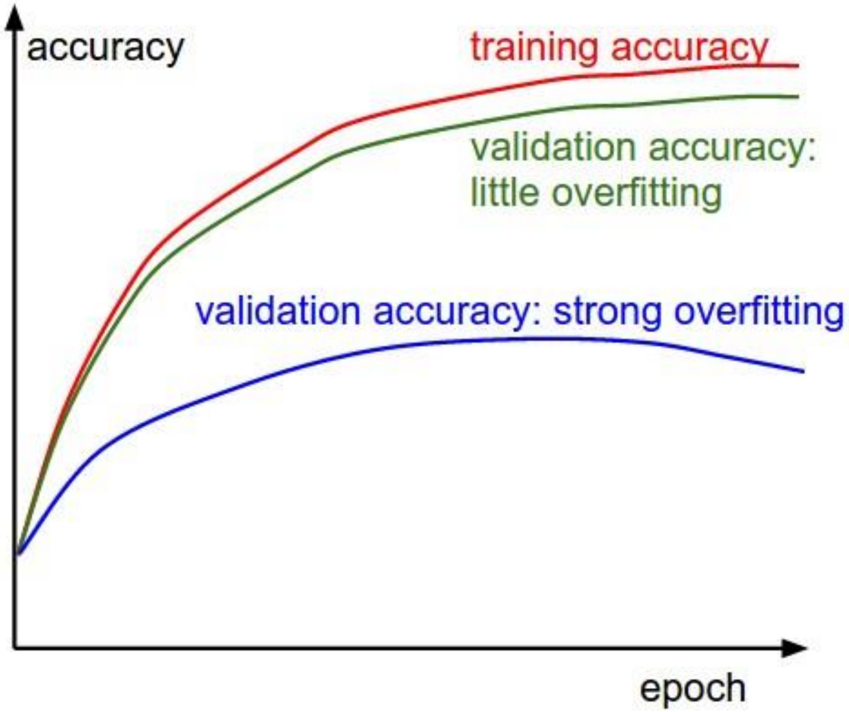


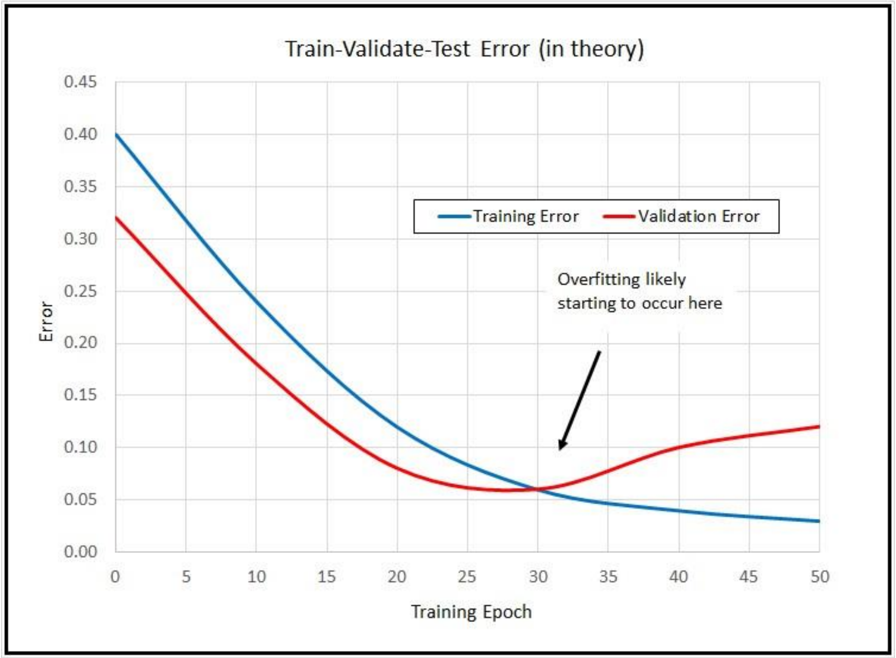




Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

# When to stop training

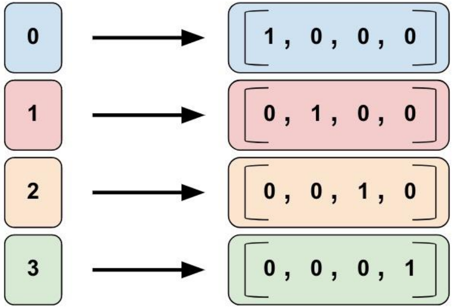


# Classification

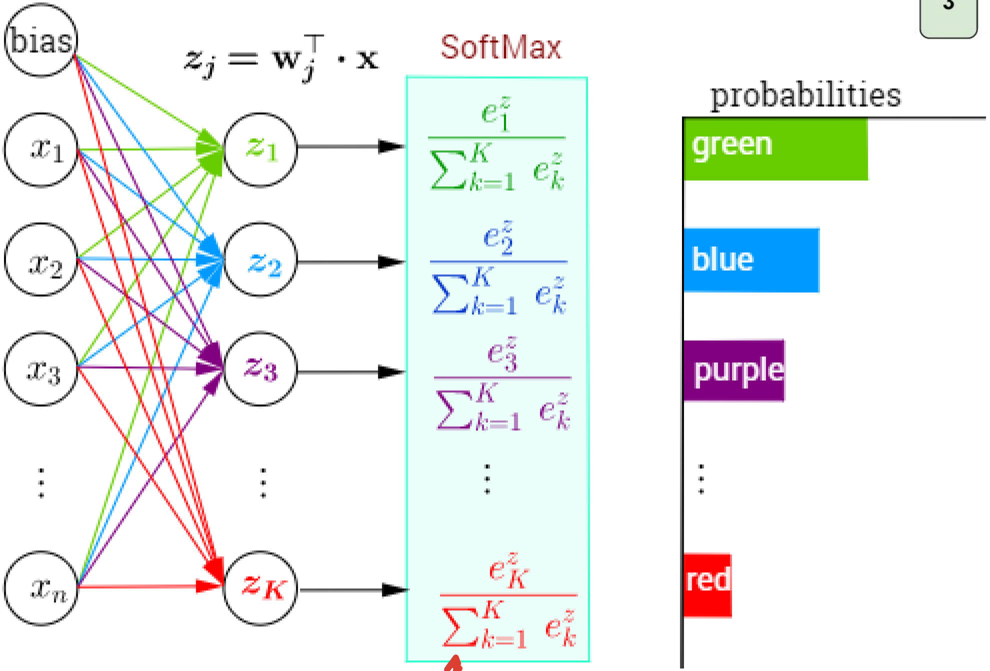
- How to represent class labels ?
- Diagram
- Loss Function

# Classification loss



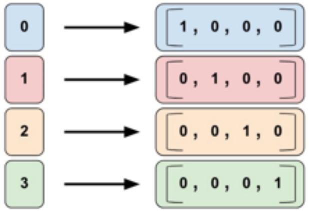

## Multi-Class Classification with NN and SoftMax Function

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

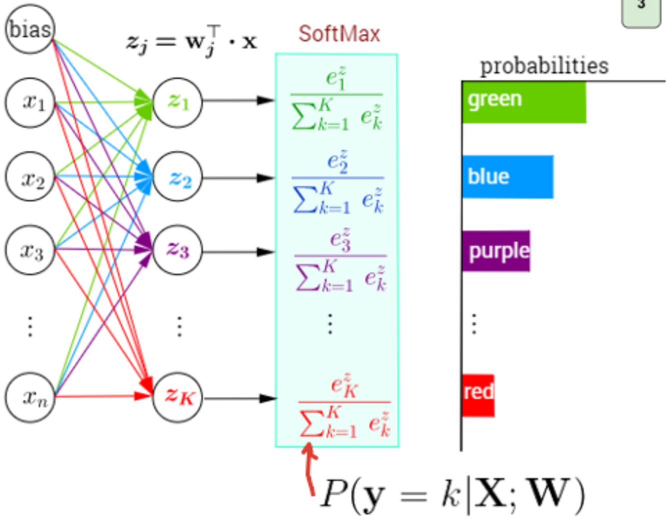


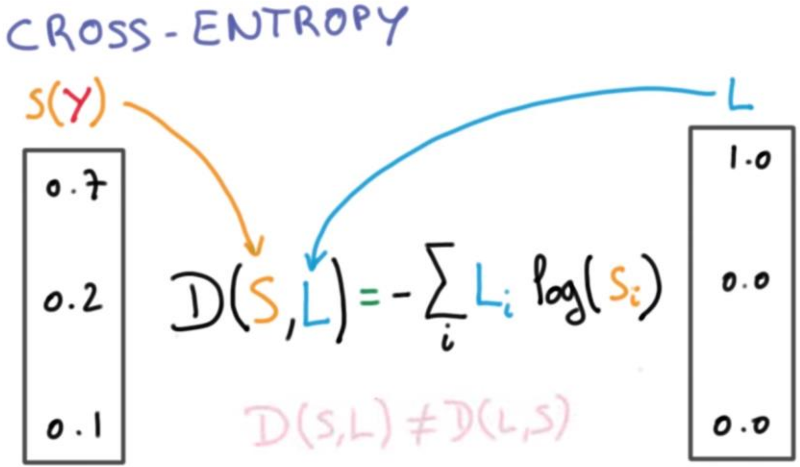

$$P(\mathbf{y} = k | \mathbf{X}; \mathbf{W})$$

# Classification Loss




$$
\begin{aligned}
D_{\mathrm{KL}}(p|q) &= \textstyle\sum_i p_i \log \frac{p_i}{q_i} \\
&= \textstyle\sum_i (-p_i \log q_i + p_i \log p_i) \\
&= -\textstyle\sum_i p_i \log q_i + \sum_i p_i \log p_i \\
&= -\textstyle\sum_i p_i \log q_i - \sum_i p_i \log \frac{1}{p_i} \\
&= -\textstyle\sum_i p_i \log q_i - H(p) \\
&= \textstyle\sum_i p_i \log \frac{1}{q_i} - H(p)
\end{aligned}
$$



# Classification loss

A

```
computed       | targets               | correct?
-----------------------------------------------
0.3  0.3  0.4  | 0   0   1             | yes
0.3  0.4  0.3  | 0   1   0             | yes
0.1  0.2  0.7  | 1   0   0             | no
```

Average Classification Error ?

# CLASSIFICATION LOSS

A

```
computed          | targets              | correct?
-----------------------------------------------------
0.3   0.3   0.4   | 0   0   1            | yes
0.3   0.4   0.3   | 0   1   0            | yes
0.1   0.2   0.7   | 1   0   0            | no
```

Average Classification Error ?

B

```
computed          | targets              | correct?
-----------------------------------------------------
0.1   0.2   0.7   | 0   0   1            | yes
0.1   0.7   0.2   | 0   1   0            | yes
0.3   0.4   0.3   | 1   0   0            | no
```

Average Classification Error ?

# Classification loss

A

```
computed          | targets               | correct?
----------------------------------------------------
0.3   0.3   0.4   | 0    0    1           | yes
0.3   0.4   0.3   | 0    1    0           | yes
0.1   0.2   0.7   | 1    0    0           | no
```

**Which classifier is better ?**

B

```
computed          | targets               | correct?
----------------------------------------------------
0.1   0.2   0.7   | 0    0    1           | yes
0.1   0.7   0.2   | 0    1    0           | yes
0.3   0.4   0.3   | 1    0    0           | no
```

# Classification loss

A

```
computed         | targets              | correct?
------------------------------------------------
0.3  0.3  0.4    | 0   0   1            | yes
0.3  0.4  0.3    | 0   1   0            | yes
0.1  0.2  0.7    | 1   0   0            | no
```

**Which classifier is better ?**

B

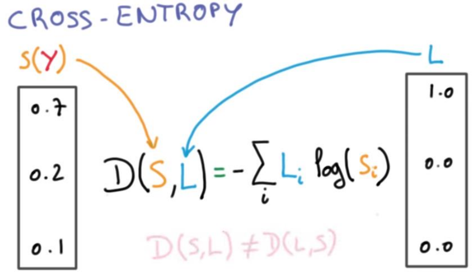
```
computed         | targets              | correct?
------------------------------------------------
0.1  0.2  0.7    | 0   0   1            | yes
0.1  0.7  0.2    | 0   1   0            | yes
0.3  0.4  0.3    | 1   0   0            | no
```

Classification accuracy is a crude way to measure how well NN has been trained!

# CLASSIFICATION LOSS

A

```
computed         | targets               | correct?
-----------------------------------------------
0.3  0.3  0.4    | 0   0   1             | yes
0.3  0.4  0.3    | 0   1   0             | yes
0.1  0.2  0.7    | 1   0   0             | no
```




**Cross-entropy error ?**

B

```
computed         | targets               | correct?
-----------------------------------------------
0.1  0.2  0.7    | 0   0   1             | yes
0.1  0.7  0.2    | 0   1   0             | yes
0.3  0.4  0.3    | 1   0   0             | no
```

Classification accuracy is a crude way to measure how well NN has been trained!

# Classification loss

A

MSE ?

| computed | targets | correct? |
|---|---|---|
| 0.3 0.3 0.4 | 0 0 1 | yes |
| 0.3 0.4 0.3 | 0 1 0 | yes |
| 0.1 0.2 0.7 | 1 0 0 | no |

B

| computed | targets | correct? |
|---|---|---|
| 0.1 0.2 0.7 | 0 0 1 | yes |
| 0.1 0.7 0.2 | 0 1 0 | yes |
| 0.3 0.4 0.3 | 1 0 0 | no |

ln() function in cross-entropy takes into account the closeness of a prediction and is a more granular way to compute error.

# Resources

- https://playground.tensorflow.org/
- https://betterexplained.com/articles/derivatives-product-power-chain/
- http://www.3blue1brown.com/videos/2017/10/9/neural-network
- http://cs231n.github.io/neural-networks-2/
- http://cs231n.github.io/neural-networks-3