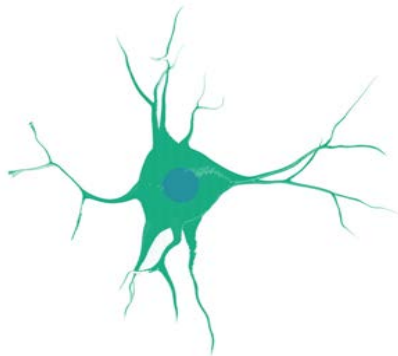


# Neural Networks

Cynthia Rudin  
Duke Machine Learning

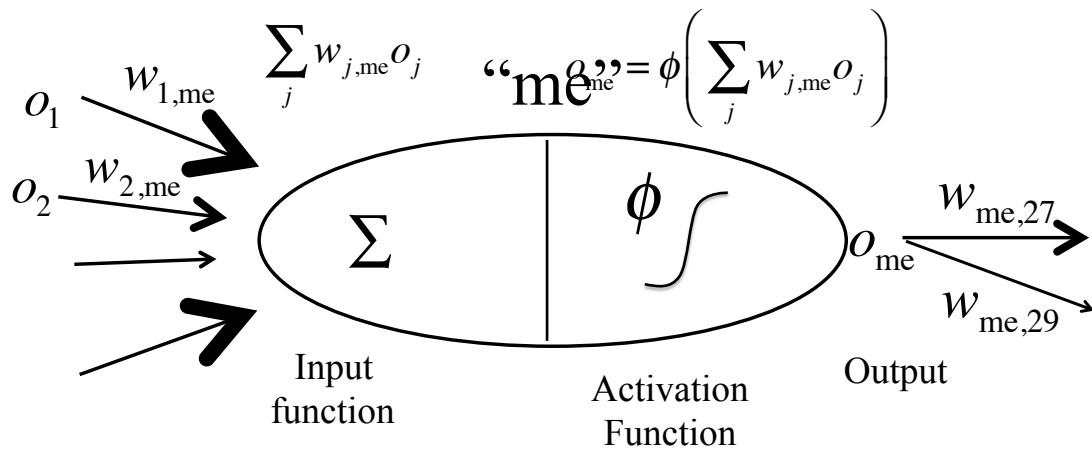
## Neurons

- $10^{11}$  neurons in a brain,  $10^{14}$  synapses (connections).
- Signals are electrical potential spikes that travel through the network.

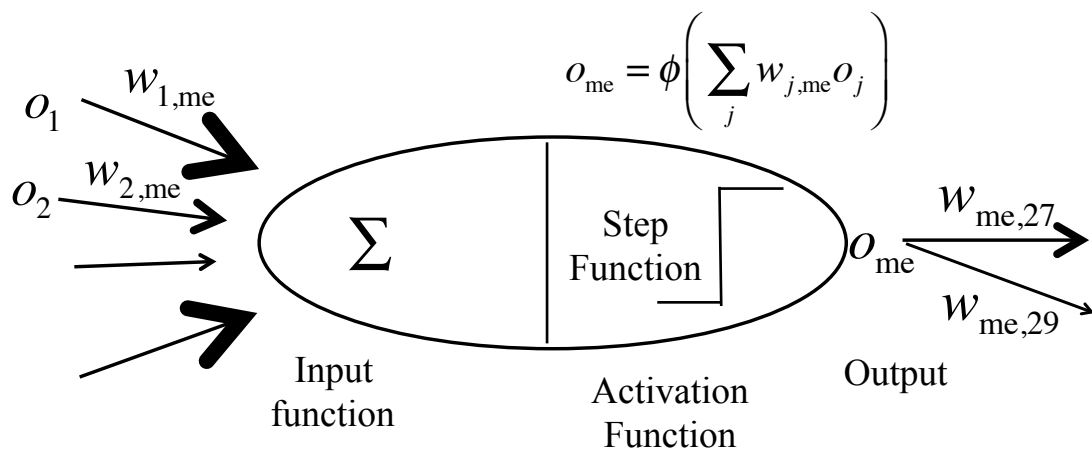


(Credit: Adapted from Russell and Norvig)

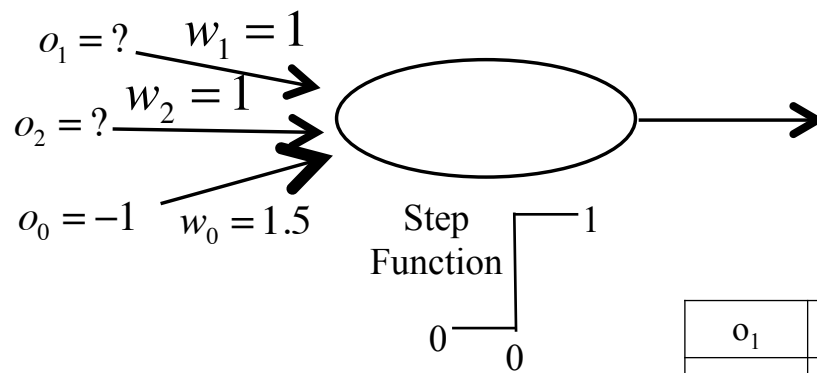
## McCulloch-Pitts “Neuron”



## McCulloch-Pitts “Neuron”

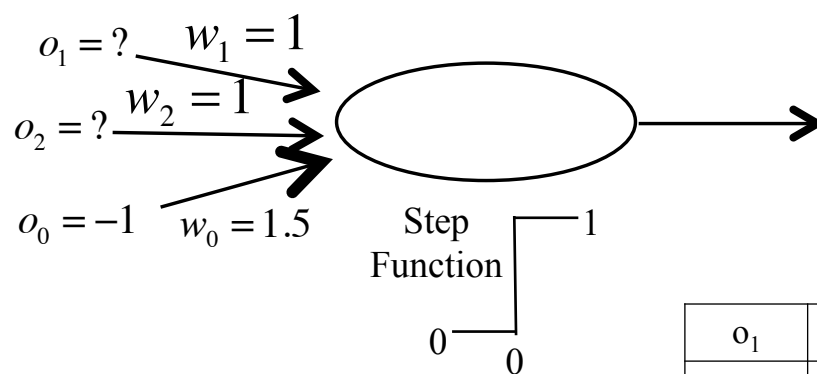


## McCulloch-Pitts “Neuron”



$o_1$	$o_2$	output

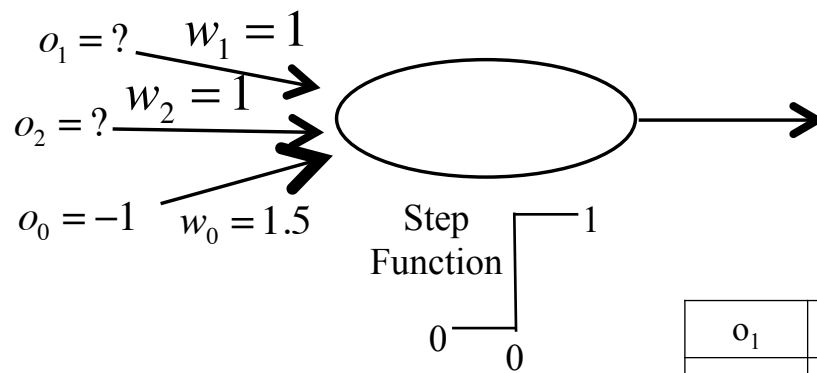
## McCulloch-Pitts “Neuron”



$$0 + 0 - 1.5 = -1.5$$

$o_1$	$o_2$	output
0	0	

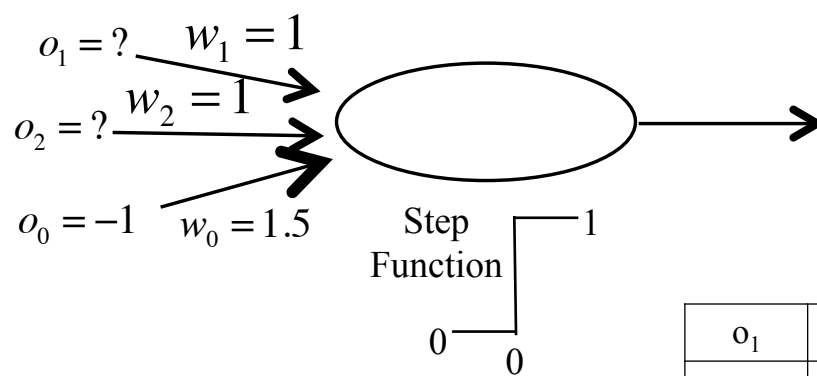
## McCulloch-Pitts “Neuron”



$$0+0-1.5 = -1.5$$

$o_1$	$o_2$	output
0	0	0

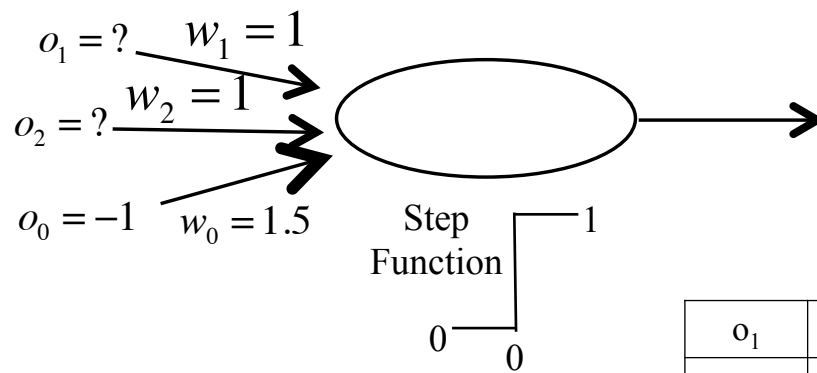
## McCulloch-Pitts “Neuron”



$$1+0-1.5 = -0.5$$

$o_1$	$o_2$	output
0	0	0
1	0	0

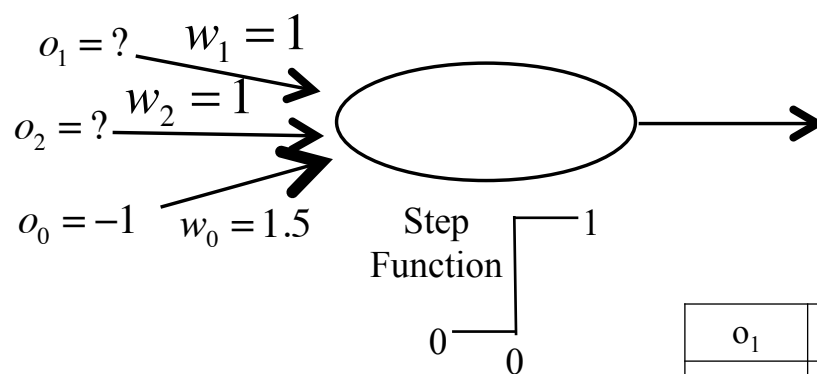
## McCulloch-Pitts “Neuron”



$$0 + 1 - 1.5 = -0.5$$

$o_1$	$o_2$	output
0	0	0
1	0	0
0	1	0

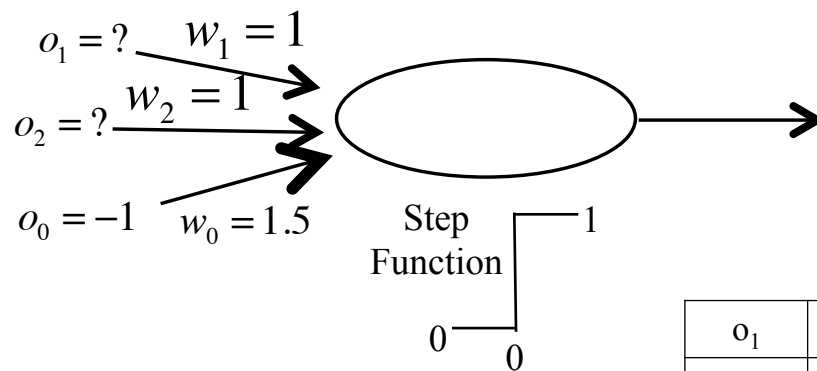
## McCulloch-Pitts “Neuron”



$$1 + 1 - 1.5 = 0.5$$

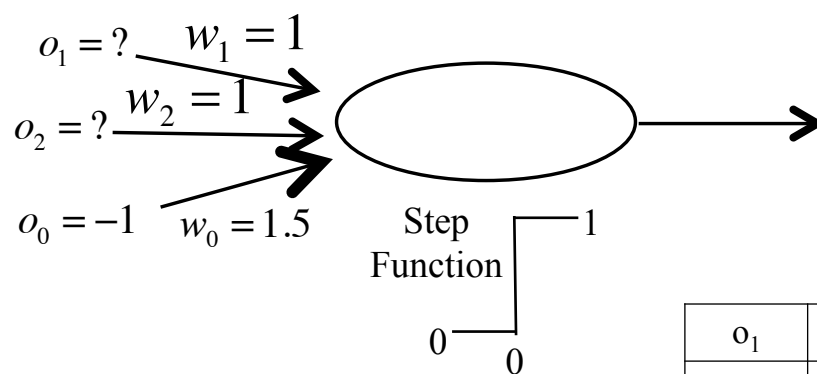
$o_1$	$o_2$	output
0	0	0
1	0	0
0	1	0
1	1	

## McCulloch-Pitts “Neuron”



$o_1$	$o_2$	output
0	0	0
1	0	0
0	1	0
1	1	1

## McCulloch-Pitts “Neuron”



This neuron  
computes the  
function “and.”

There are “or” and “not” neurons too.

$o_1$	$o_2$	output
0	0	0
1	0	0
0	1	0
1	1	1

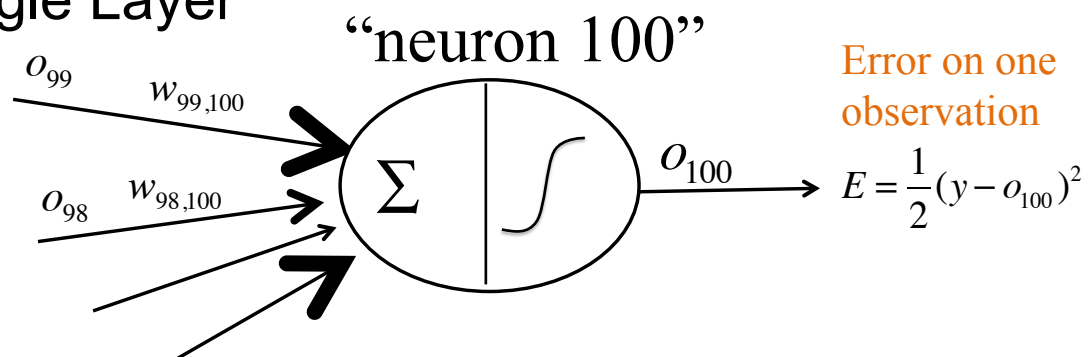
## McCulloch-Pitts “Neuron”

$$\phi\left(\sum_j w_{j,me} a_j\right) = 1 / (1 + e^{-x}) \quad \text{“Sigmoid”}$$

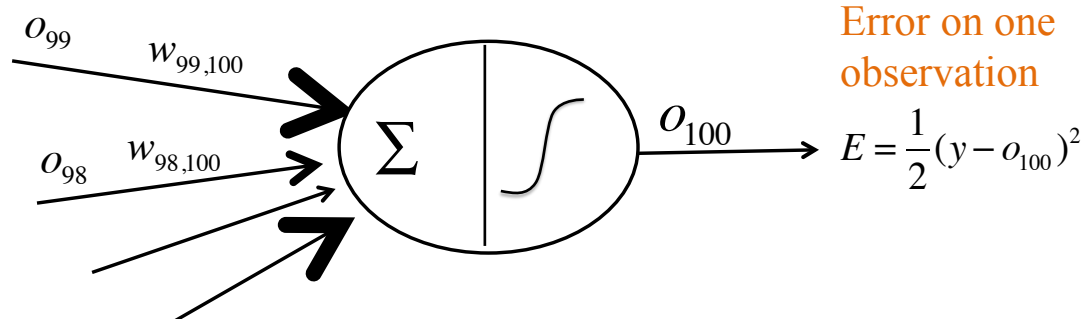


Activation  
Function

## Single Layer



## Single Layer



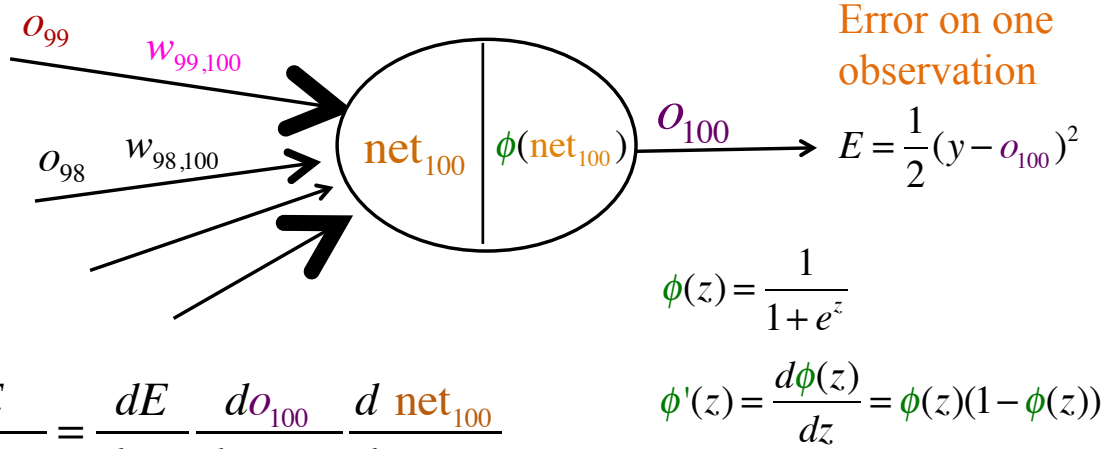
- In a brain, the synapses strengthen and weaken in order to learn.
- Say the same thing happens here.
- How should we set the weights in order to learn (reduce the error)?
- Minimize  $E$  with respect to the weights.

## Backpropagation

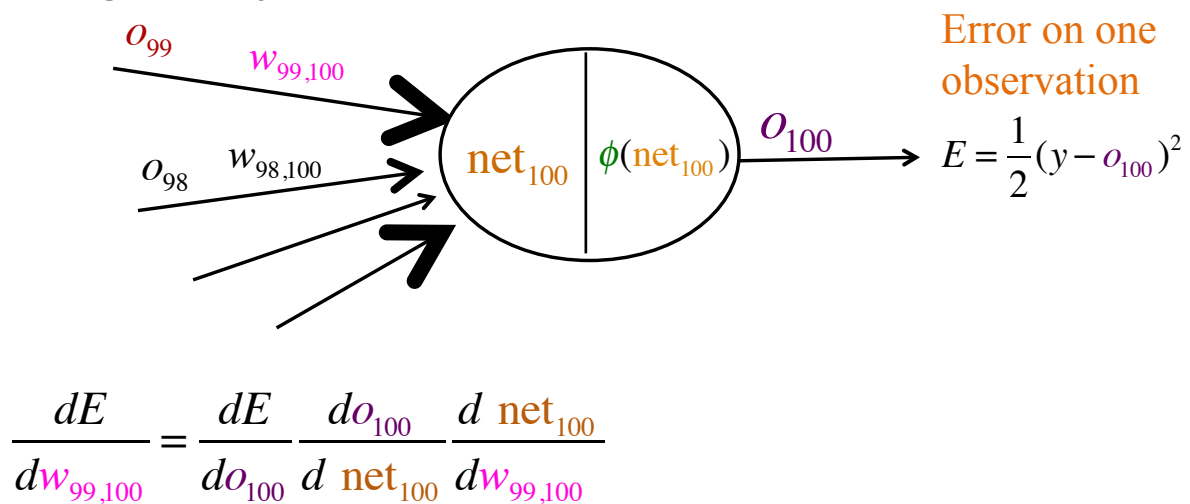
- An algorithm that trains the weights of a neural network
- Requires us to propagate information backwards through the network, then forwards, then backwards, then forwards, etc.
- Propagate backwards = chain rule from calculus.



## Single Layer



## Single Layer



## Single Layer

Error on one observation

$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{do_{100}} = \frac{1}{2} 2(y - o_{100})(-1) = -(y - o_{100})$$

$$\frac{dE}{dw_{99,100}} = \left( \frac{dE}{do_{100}} \right) \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$

## Single Layer

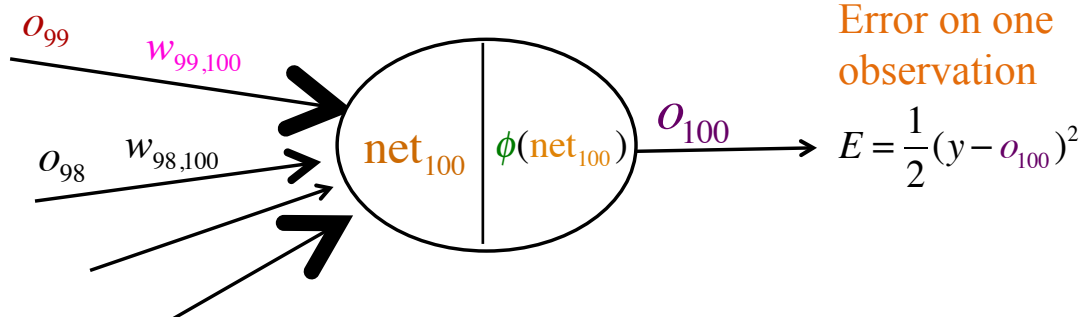
Error on one observation

$$E = \frac{1}{2}(y - o_{100})^2$$

$$\frac{dE}{dw_{99,100}} = \left( \frac{dE}{do_{100}} \right) \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$

## Single Layer



$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \left( \frac{do_{100}}{d \text{net}_{100}} \right) \frac{d \text{net}_{100}}{dw_{99,100}}$$

$\nearrow$   
 $-(y - o_{100})$

## Single Layer

$$\phi(\text{net}_{100}) \quad o_{100}$$

$$\frac{do_{100}}{d \text{net}_{100}} = \frac{d\phi(\text{net}_{100})}{d \text{net}_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1 - \phi(\text{net}_{100})) = o_{100}(1 - o_{100})$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \left( \frac{do_{100}}{d \text{net}_{100}} \right) \frac{d \text{net}_{100}}{dw_{99,100}}$$

$\nearrow$   
 $-(y - o_{100})$

## Single Layer

$$\phi(\text{net}_{100}) \quad o_{100}$$

$$\frac{do_{100}}{d \text{net}_{100}} = \frac{d\phi(\text{net}_{100})}{d \text{net}_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1 - \phi(\text{net}_{100})) = o_{100}(1 - o_{100})$$

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

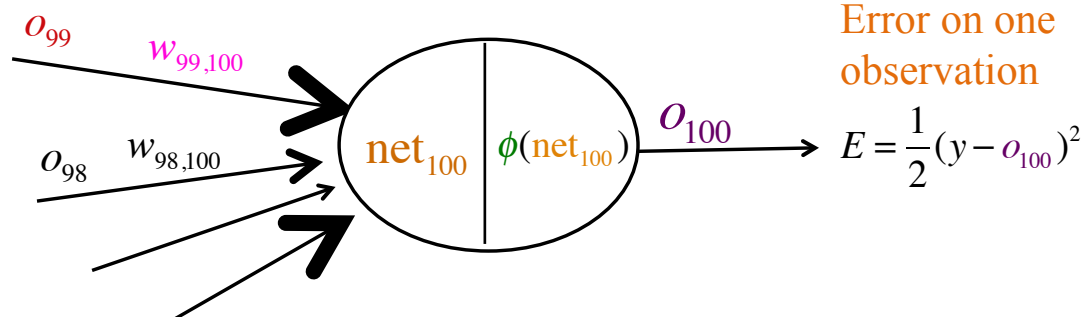
$-(y - o_{100})$        $o_{100}(1 - o_{100})$

## Single Layer

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$        $o_{100}(1 - o_{100})$

## Single Layer

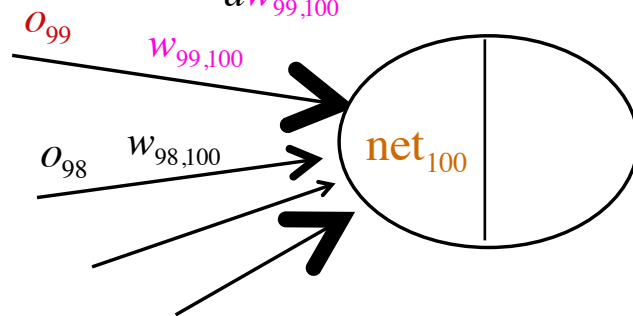


$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

Annotations for the chain rule above:

- Under  $\frac{dE}{do_{100}}$ :  $-(y - o_{100})$  (with a blue arrow pointing to the term)
- Under  $\frac{do_{100}}{d \text{net}_{100}}$ :  $o_{100}(1 - o_{100})$  (with a blue arrow pointing to the term)
- The term  $\frac{d \text{net}_{100}}{dw_{99,100}}$  is circled in the original image.

$$\frac{d \text{net}_{100}}{dw_{99,100}} = \frac{d (w_{99,100} o_{99} + w_{98,100} o_{98} + w_{97,100} o_{97} + \dots)}{dw_{99,100}} = o_{99}$$



$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

Annotations for the chain rule above:

- Under  $\frac{dE}{do_{100}}$ :  $-(y - o_{100})$  (with a blue arrow pointing to the term)
- Under  $\frac{do_{100}}{d \text{net}_{100}}$ :  $o_{100}(1 - o_{100})$  (with a blue arrow pointing to the term)
- Under  $\frac{d \text{net}_{100}}{dw_{99,100}}$ :  $o_{99}$  (with a blue arrow pointing to the term, which is also highlighted in pink in the original image)
- The term  $\frac{d \text{net}_{100}}{dw_{99,100}}$  is circled in the original image.

$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

$-(y - o_{100})$        $o_{100}(1 - o_{100})$        $o_{99}$

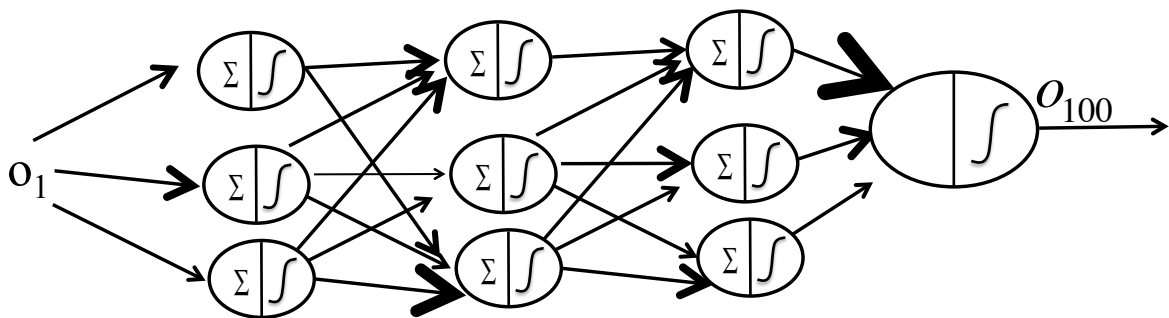
We will need this later – it depends only on node 100

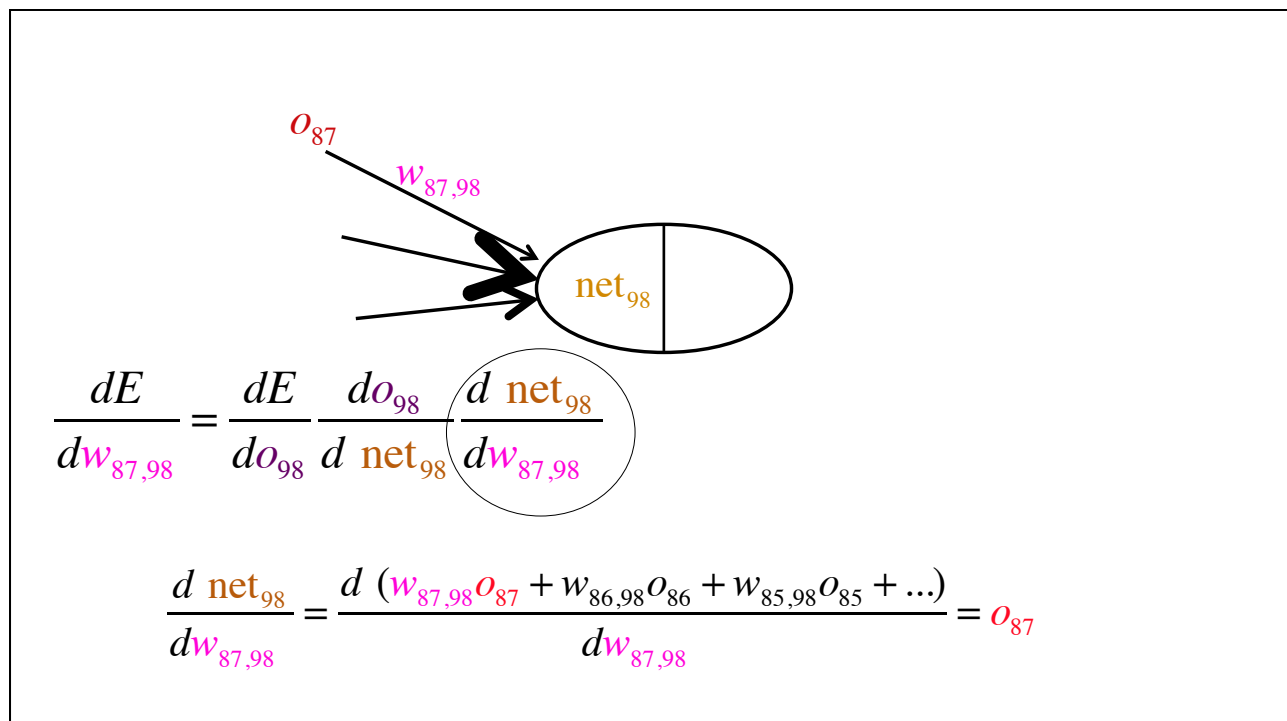
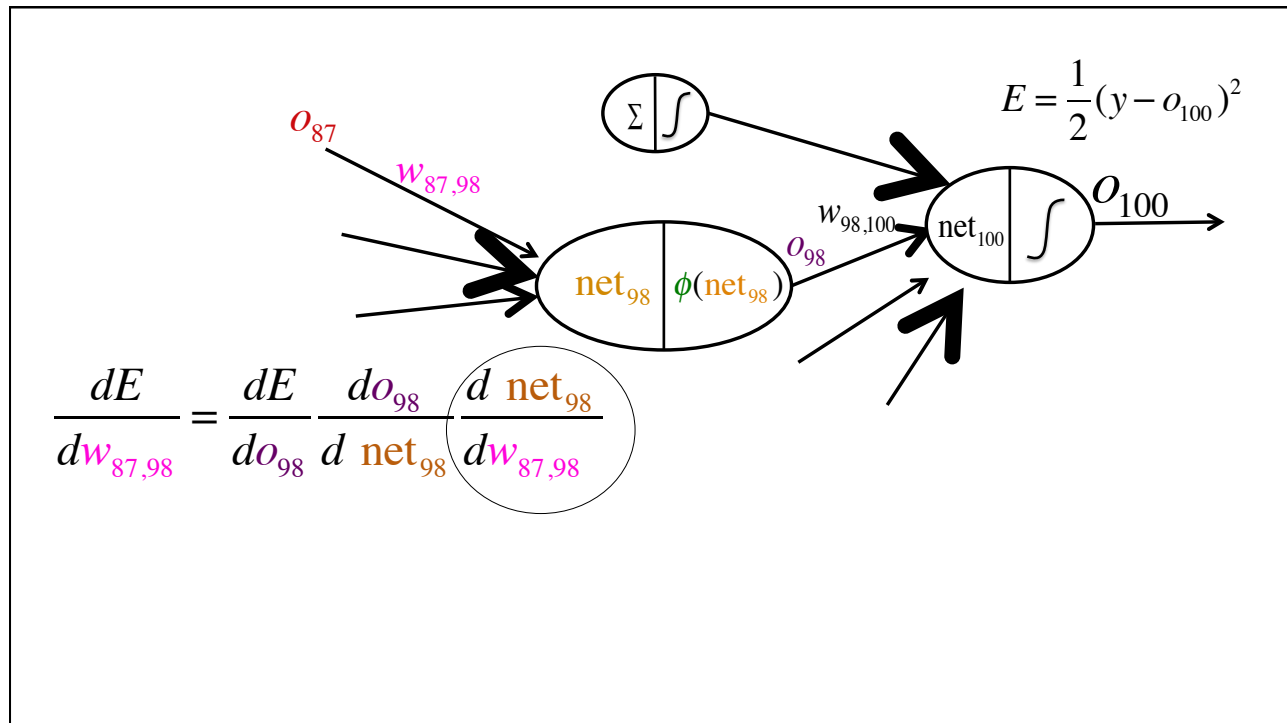
$$\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}$$

$$= \delta_{100} o_{100}(1 - o_{100}) o_{99}$$

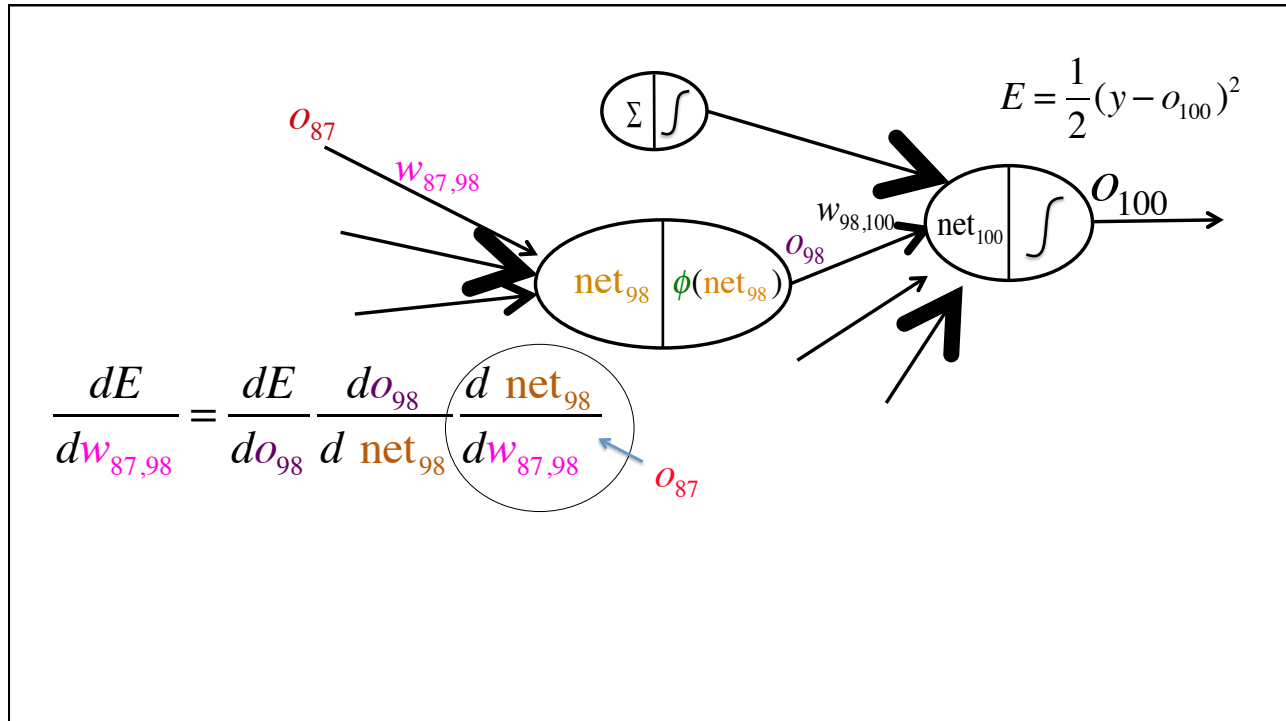
# Backpropagation

- Go one layer deeper.









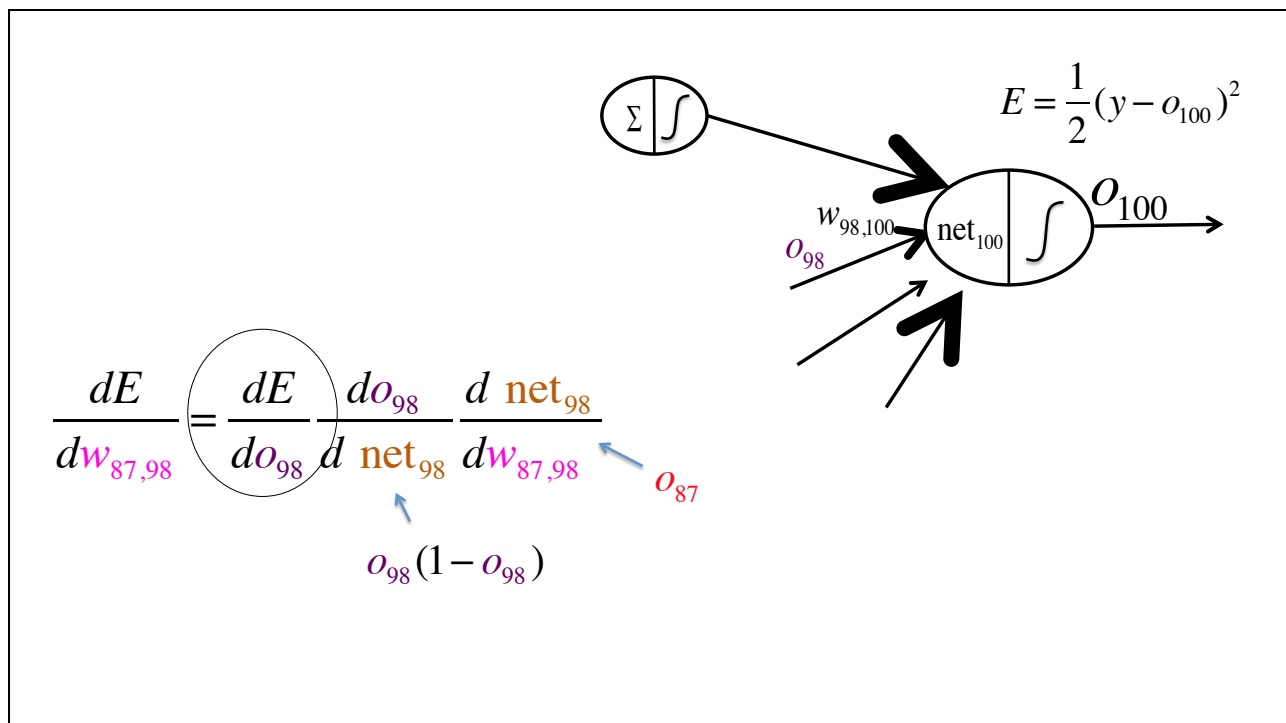
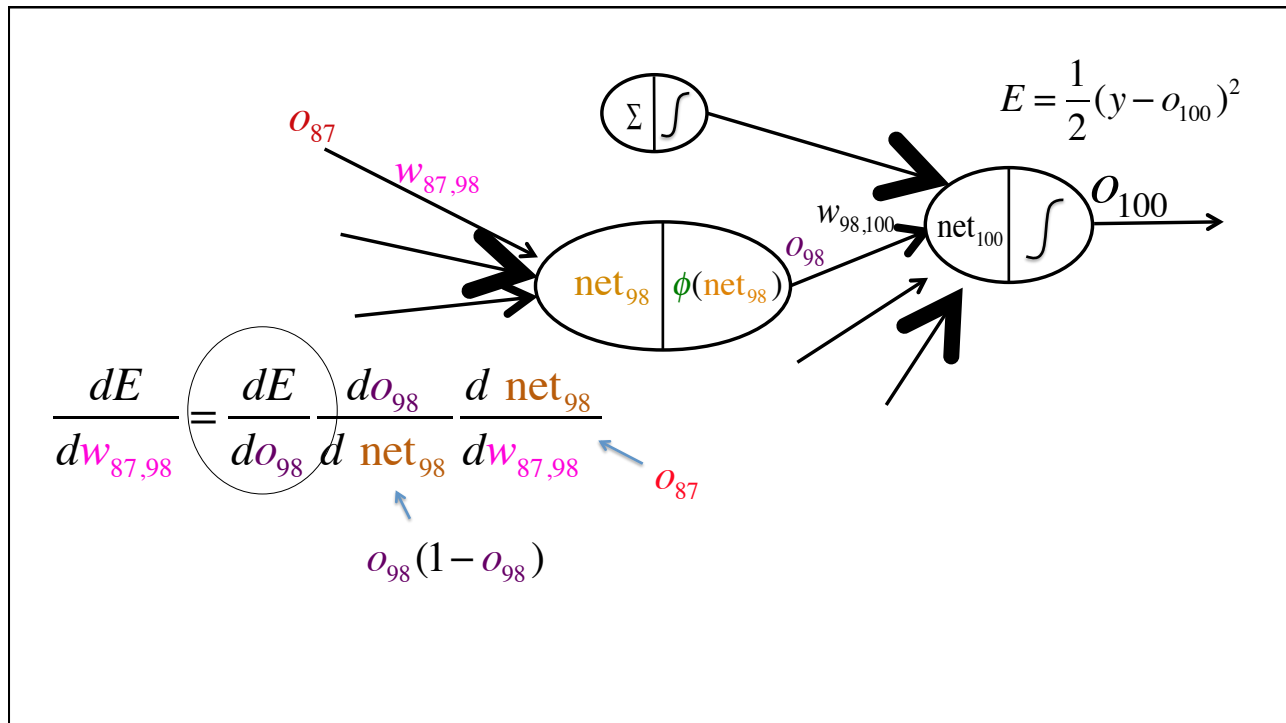
The diagram shows a single node  $net_{98}$  with input  $o_{87}$  (red) and weight  $w_{87,98}$  (pink). The node's output is  $o_{98}$  (purple). The derivative calculation for the weight  $w_{87,98}$  is shown below the diagram:

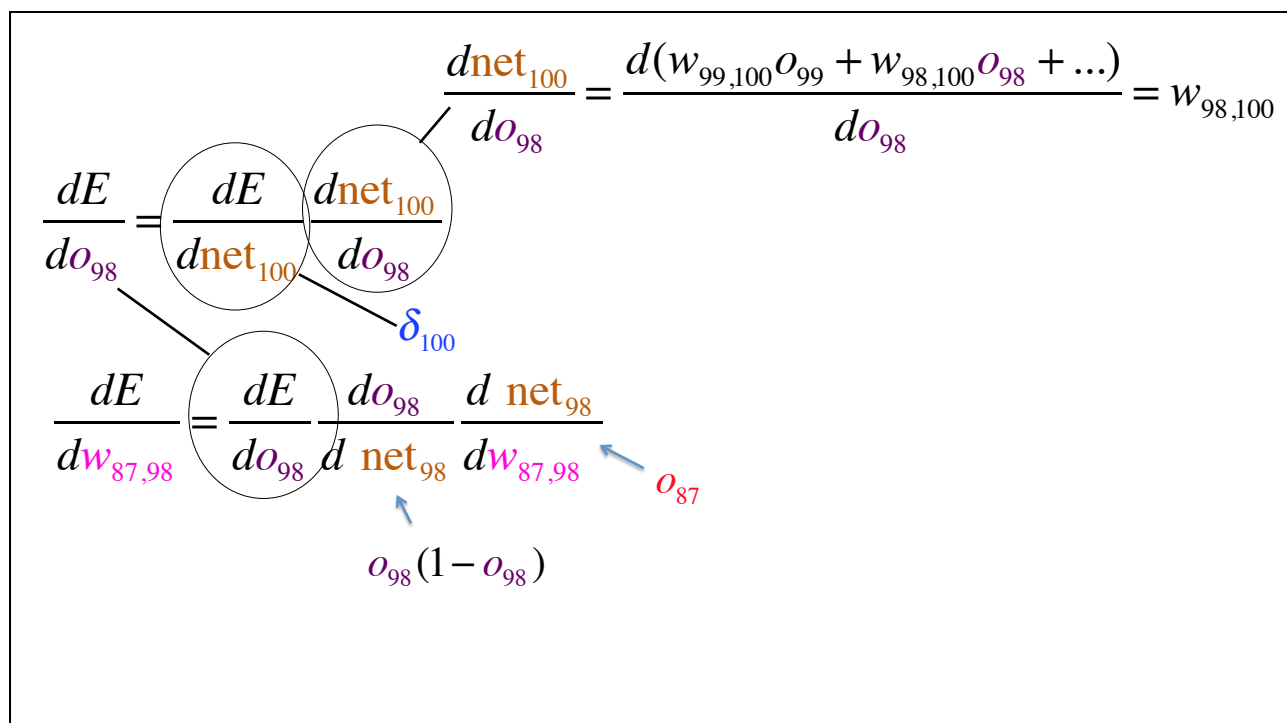
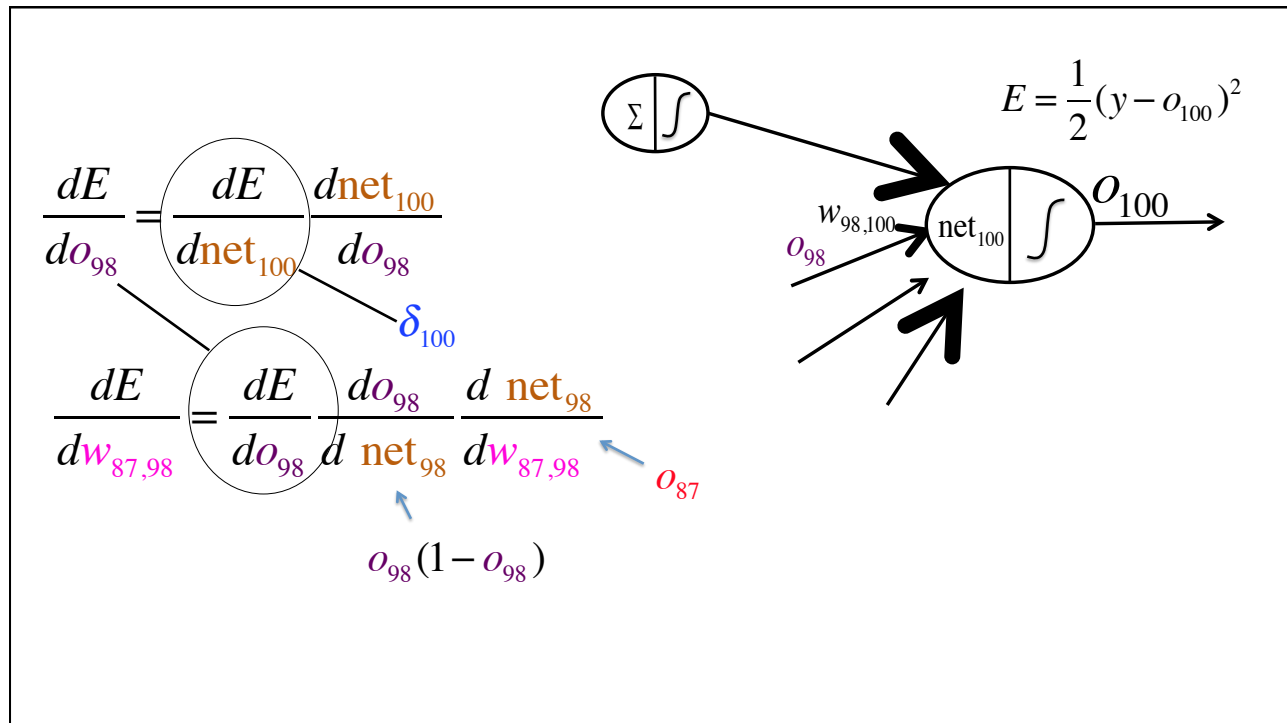
$$\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d net_{98}} \frac{d net_{98}}{dw_{87,98}}$$

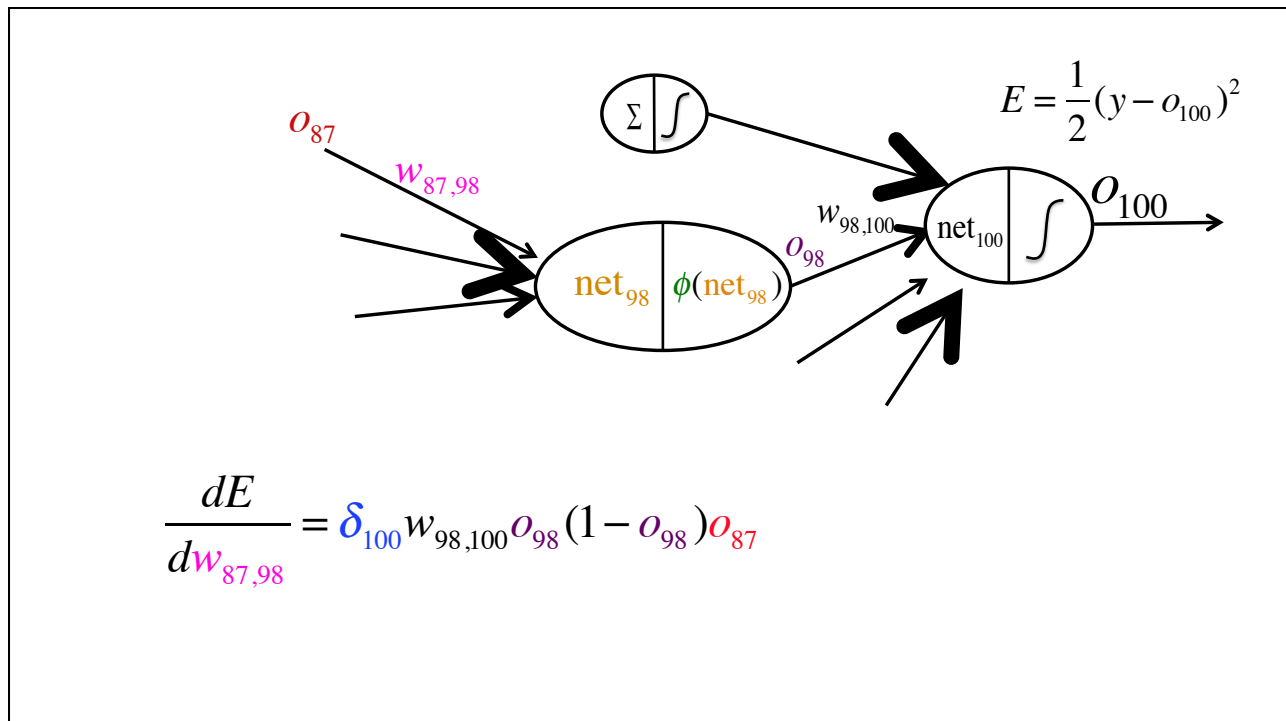
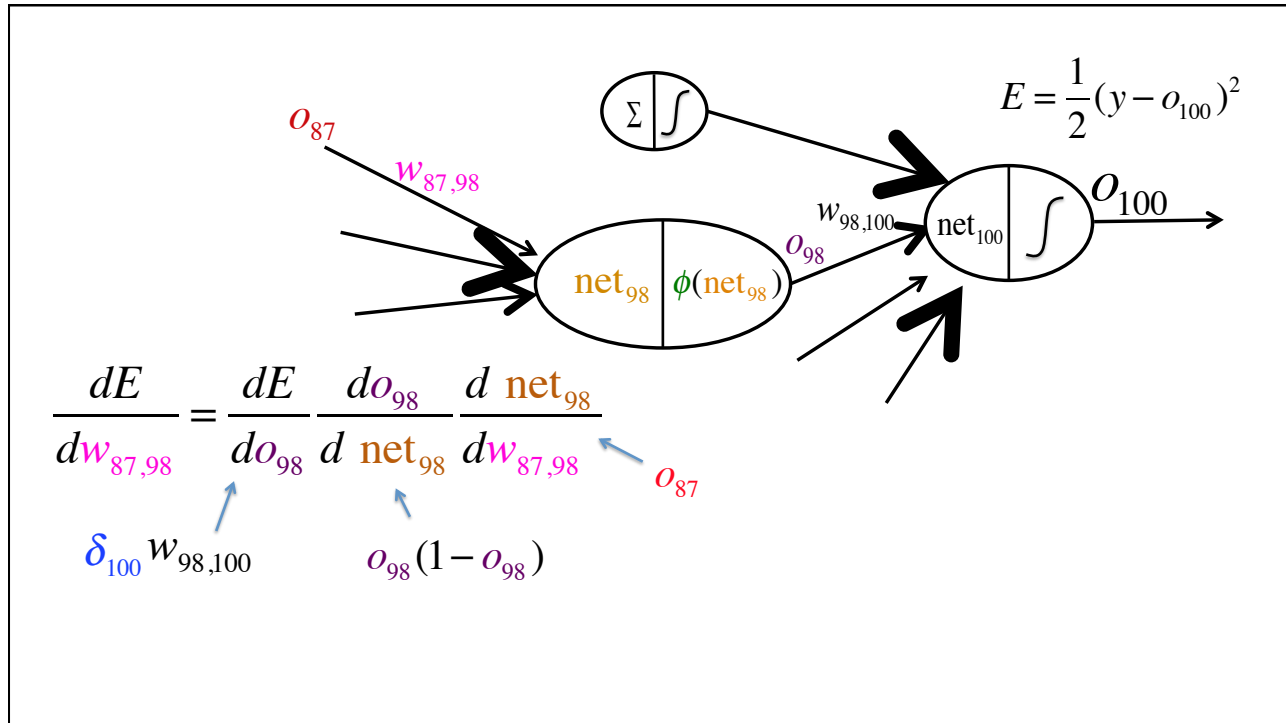
The term  $\frac{do_{98}}{d net_{98}}$  is circled, and a blue arrow points to it from the input  $o_{87}$ .

The derivative calculation for the output  $o_{98}$  is shown below the first equation:

$$\frac{do_{98}}{d net_{98}} = \frac{d\phi(net_{98})}{d net_{98}} = \phi'(net_{98}) = \phi(net_{98})(1 - \phi(net_{98})) = o_{98}(1 - o_{98})$$

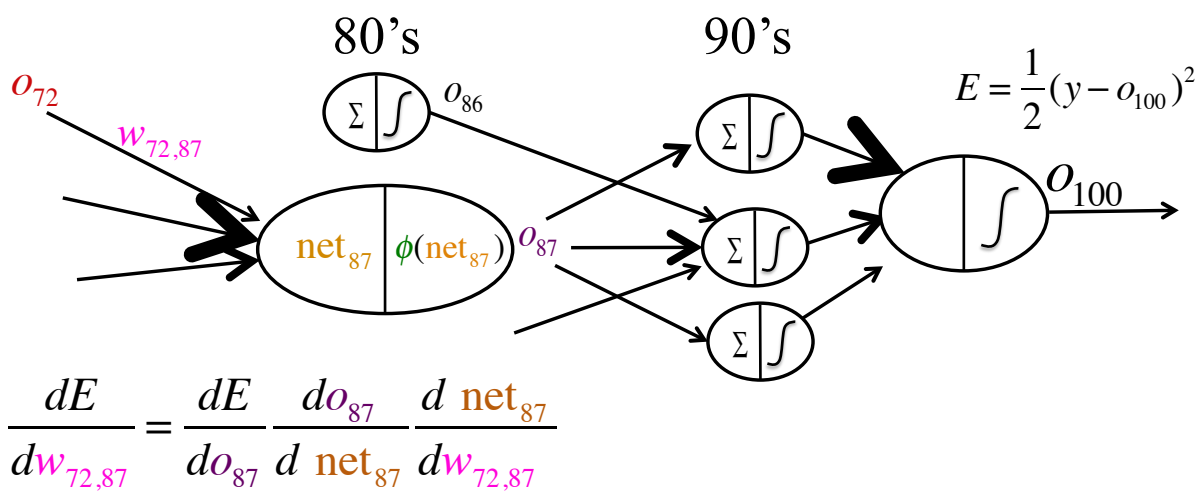


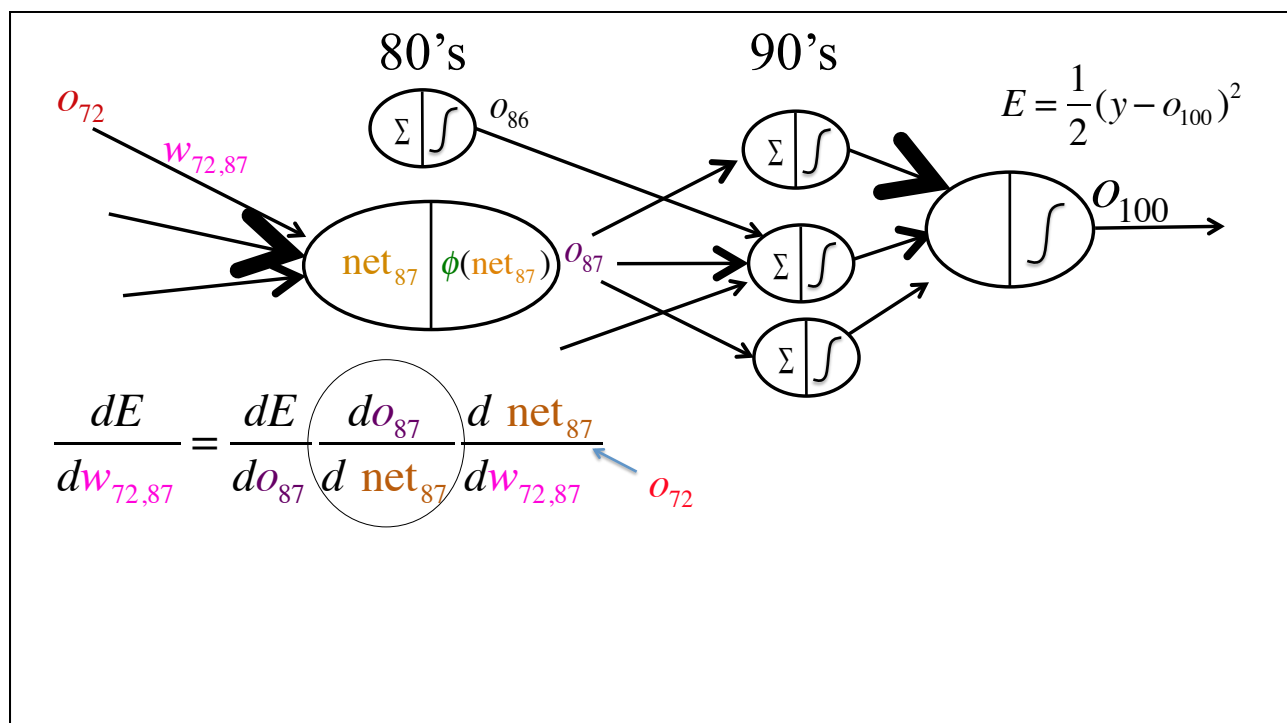
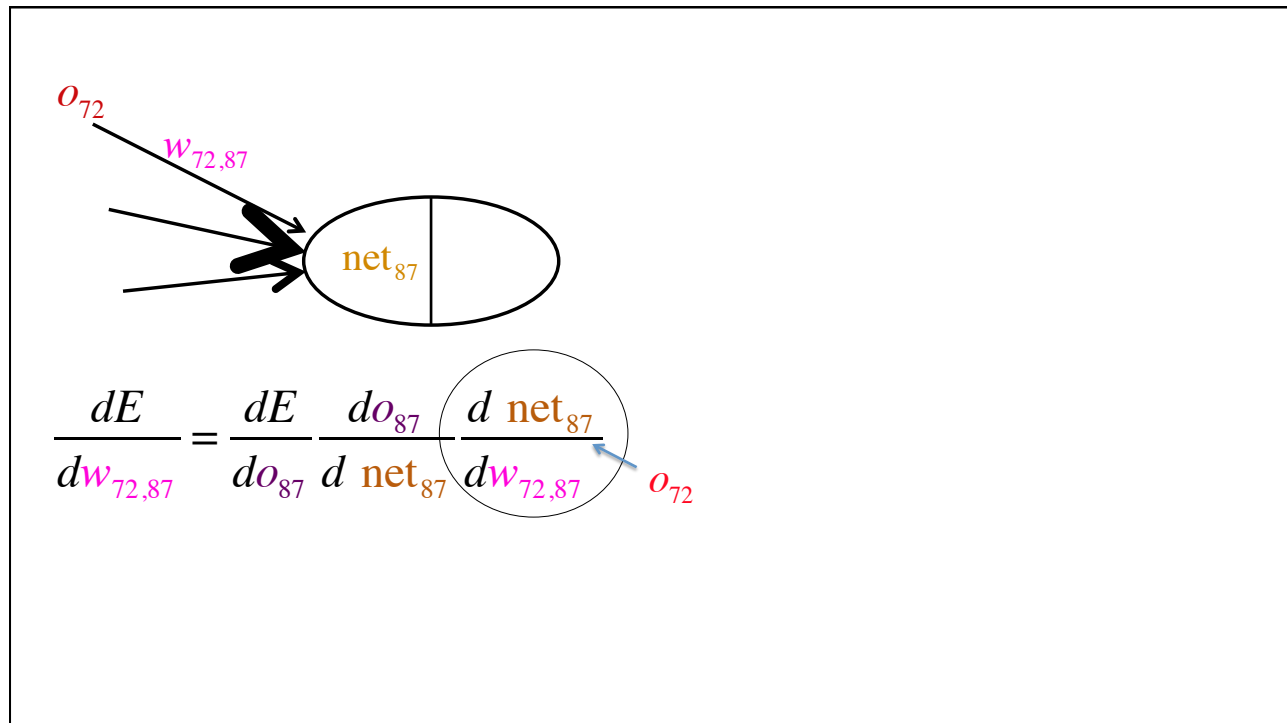




# Backpropagation

- Go even one layer deeper.
- Third time is a charm.





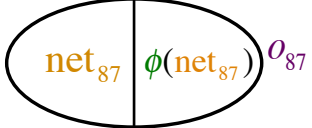
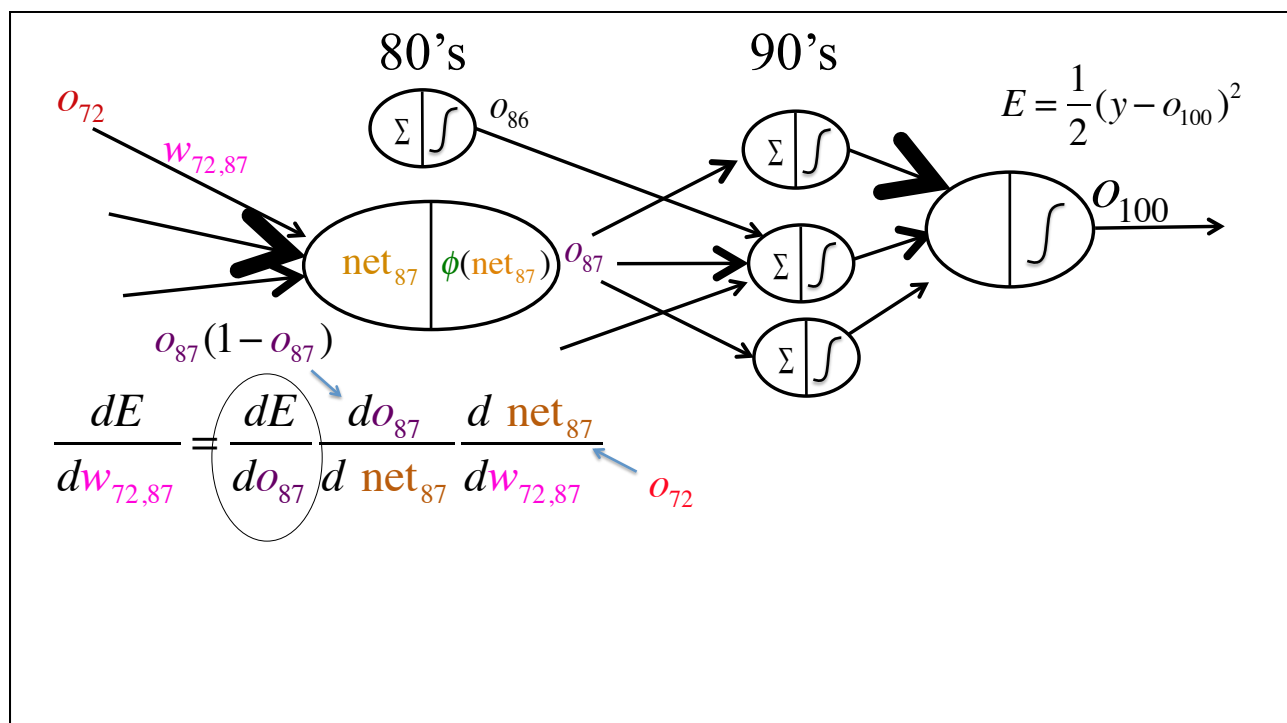


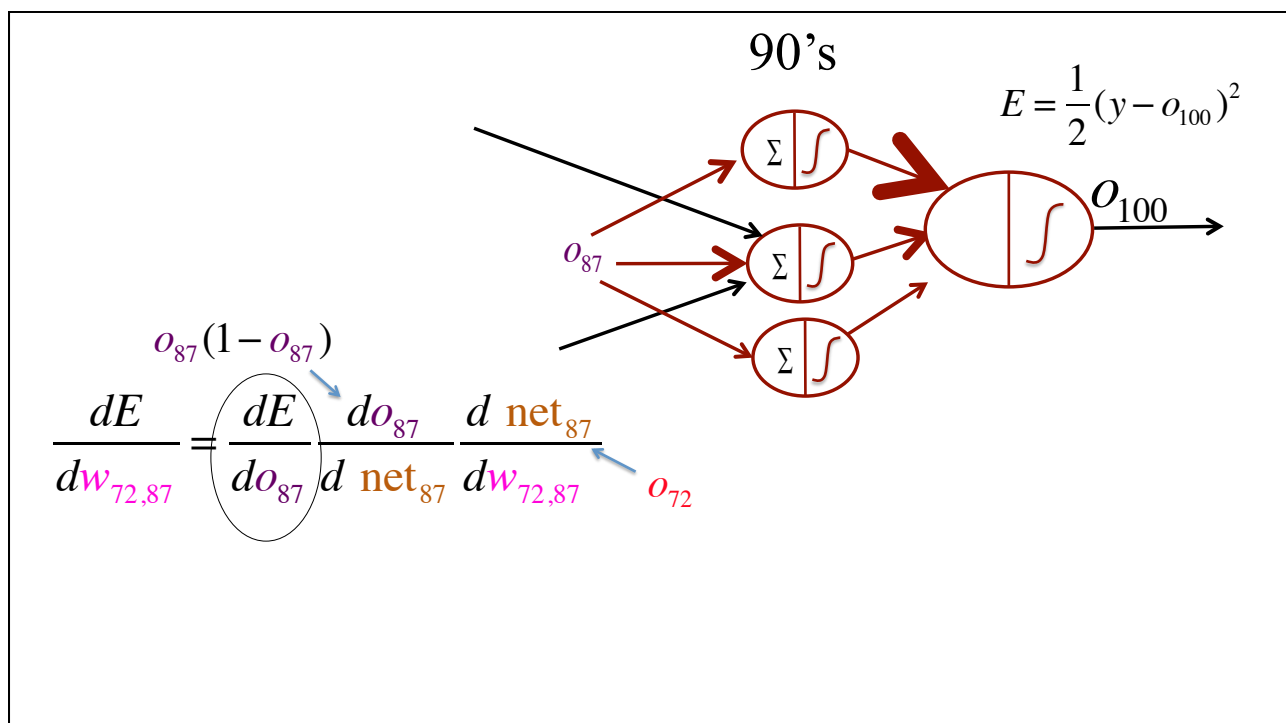
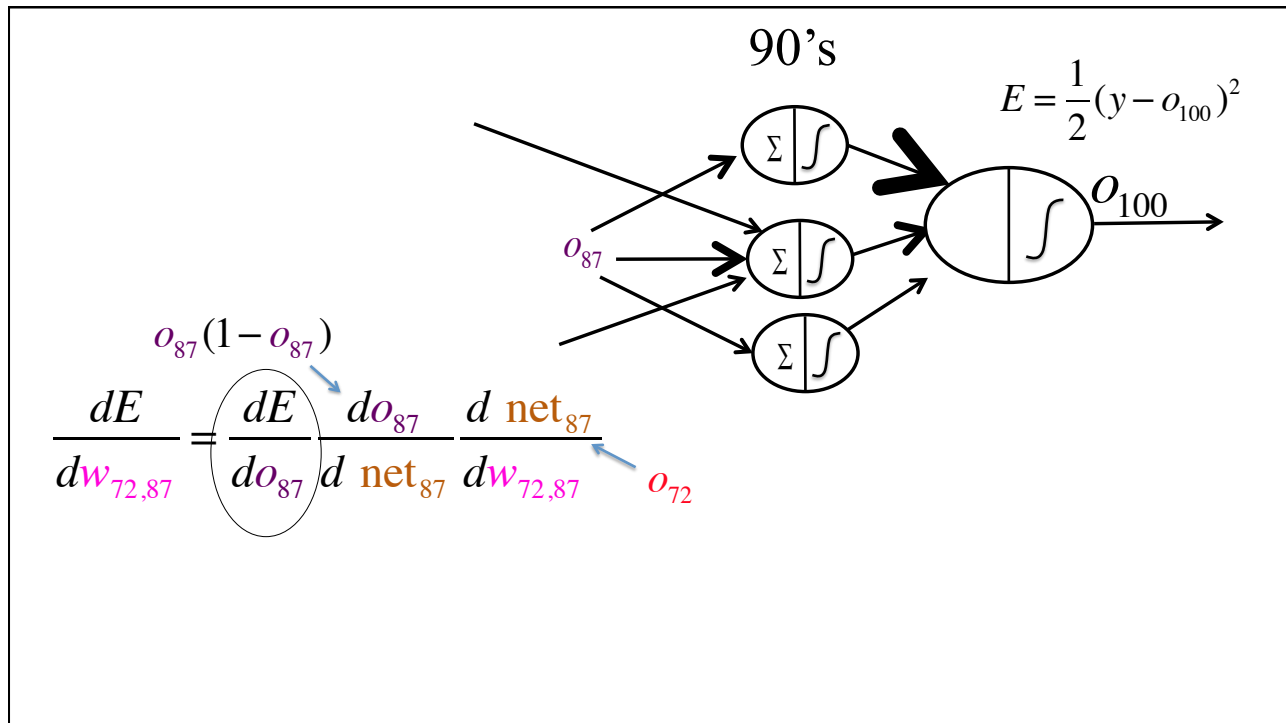
Diagram illustrating the derivative of the error with respect to the weight  $w_{72,87}$  for a single neuron node.

$$\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d \text{net}_{87}} \frac{d \text{net}_{87}}{dw_{72,87}}$$

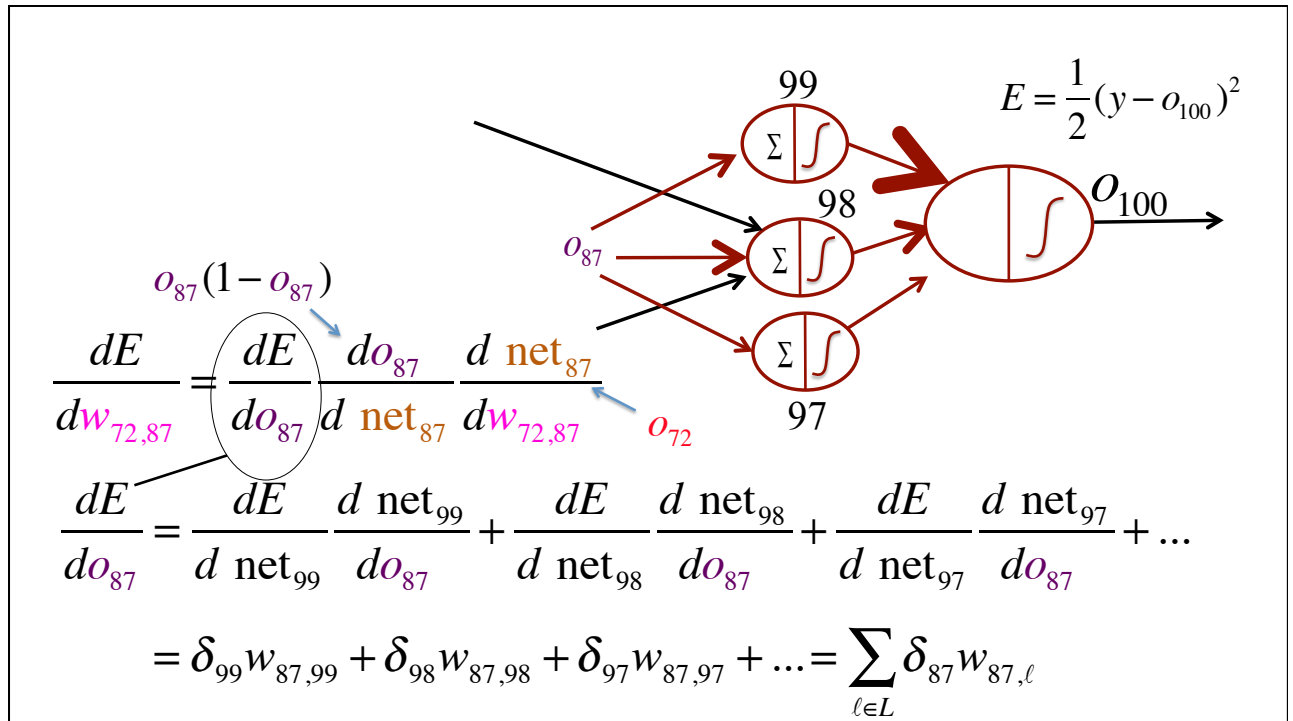
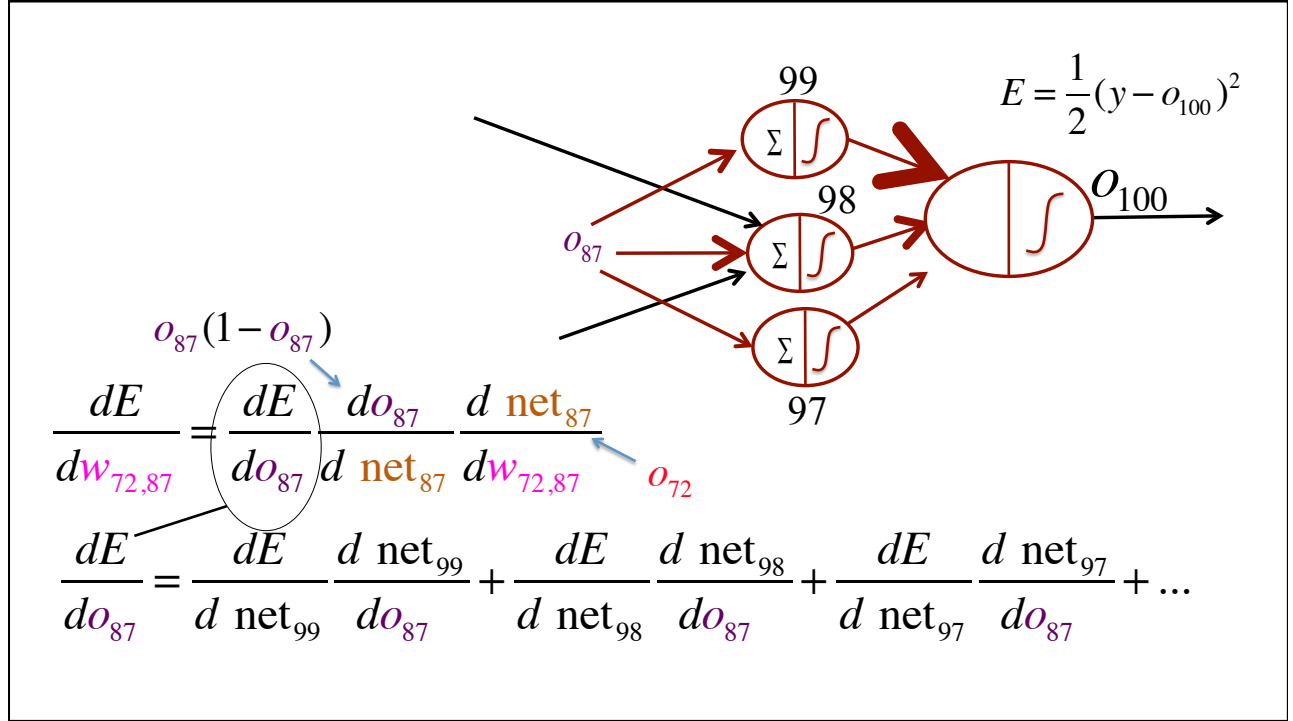
The diagram shows a node with inputs, a summation junction, an activation function  $\phi(\text{net}_{87})$ , and an output  $o_{87}$ . The derivative of the error with respect to the weight  $w_{72,87}$  is calculated as the product of the derivative of the error with respect to the output  $o_{87}$ , the derivative of the output with respect to the net input  $\text{net}_{87}$ , and the derivative of the net input with respect to the weight  $w_{72,87}$ .

$$\frac{do_{87}}{d \text{net}_{87}} = \frac{d\phi(\text{net}_{87})}{d \text{net}_{87}} = \phi'(\text{net}_{87}) = \phi(\text{net}_{87})(1 - \phi(\text{net}_{87})) = o_{87}(1 - o_{87})$$

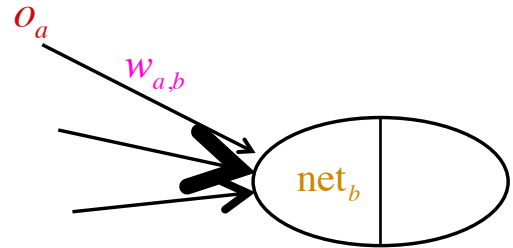




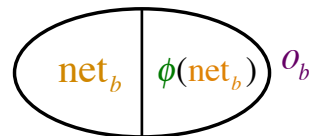




$$\begin{aligned} \frac{dE}{dw_{a,b}} &= \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} \frac{d \text{net}_b}{dw_{a,b}} \\ &= \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} o_a \end{aligned}$$



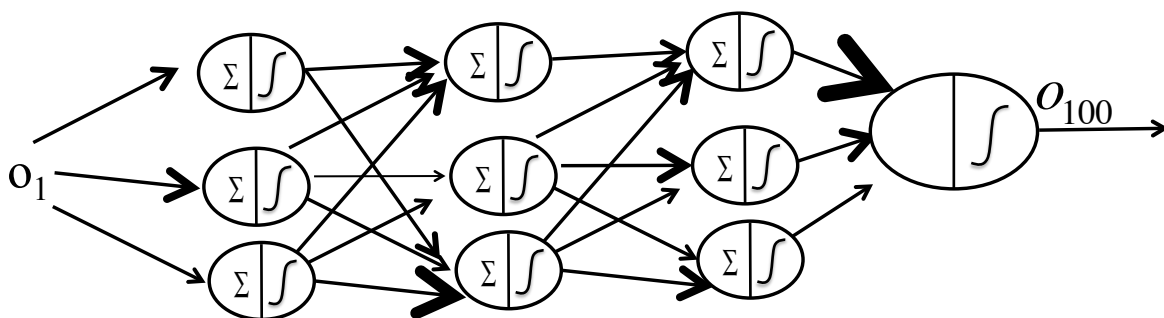
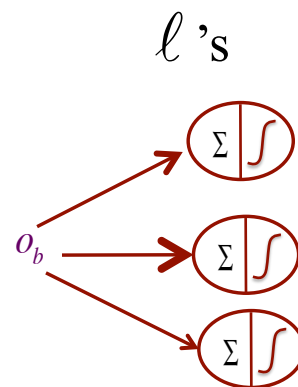
$$\begin{aligned} \frac{dE}{dw_{a,b}} &= \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} \frac{d \text{net}_b}{dw_{a,b}} \\ &= \frac{dE}{do_b} o_b (1 - o_b) o_a \end{aligned}$$

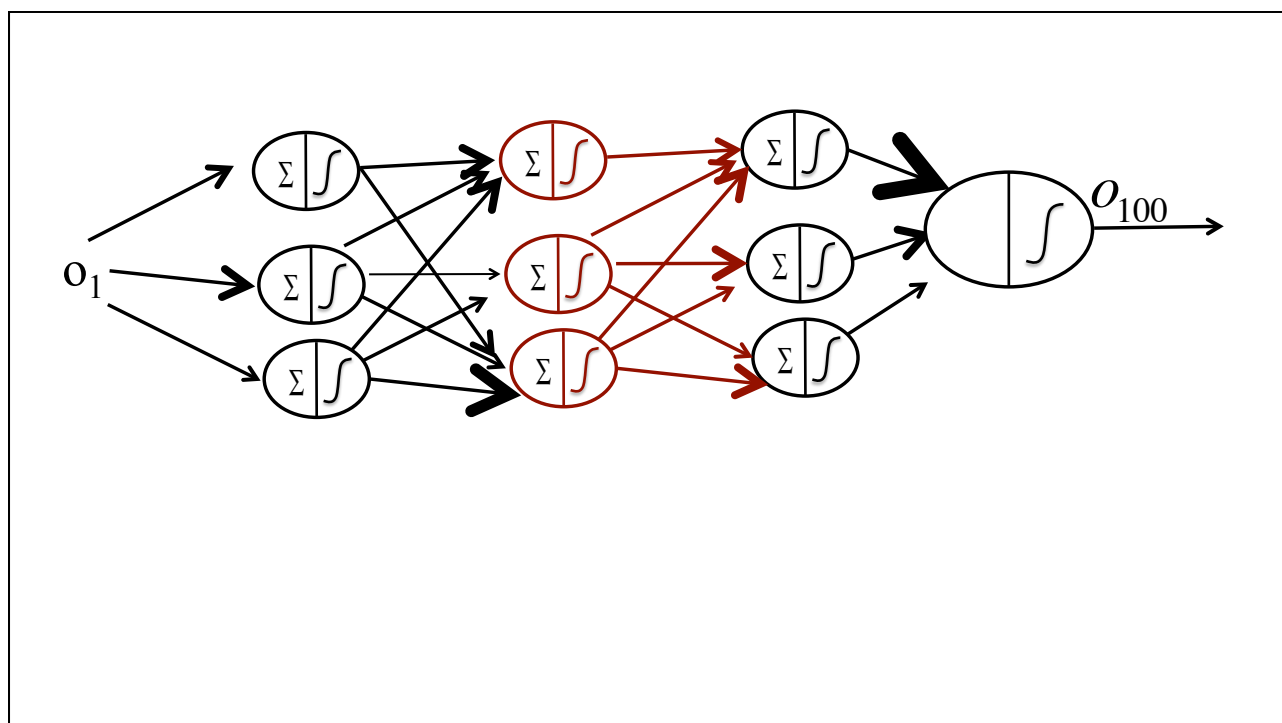
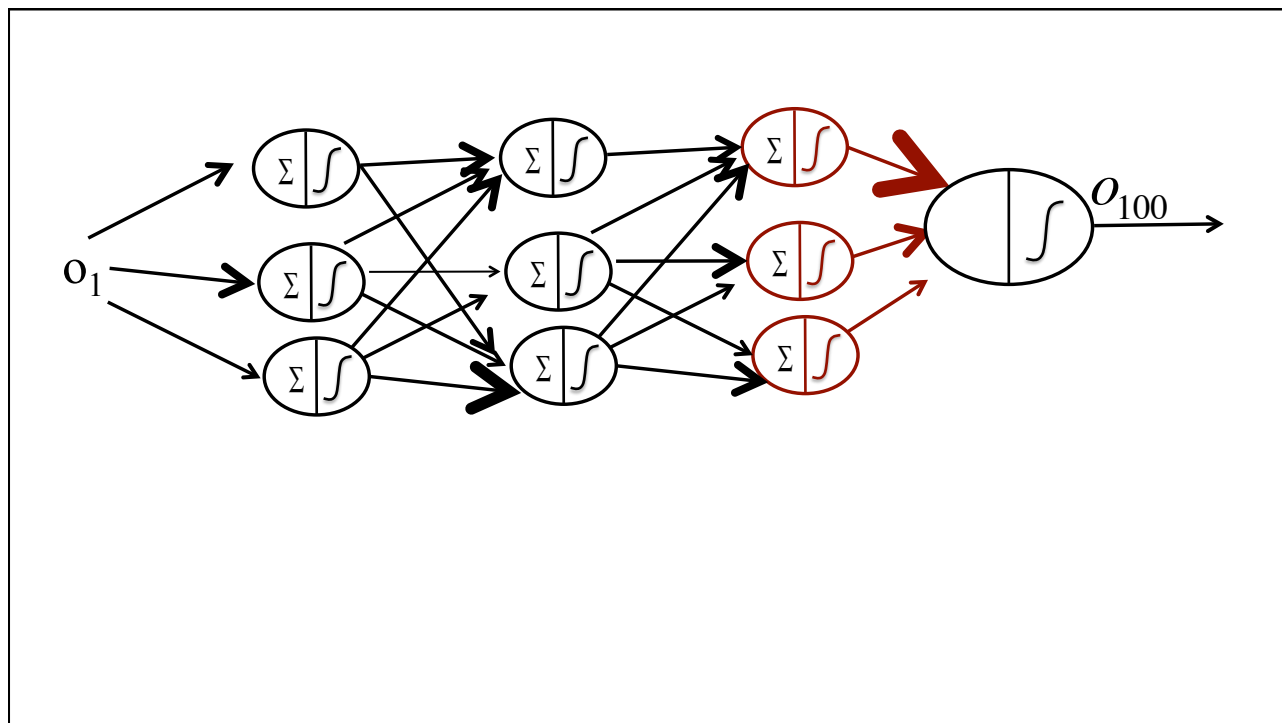


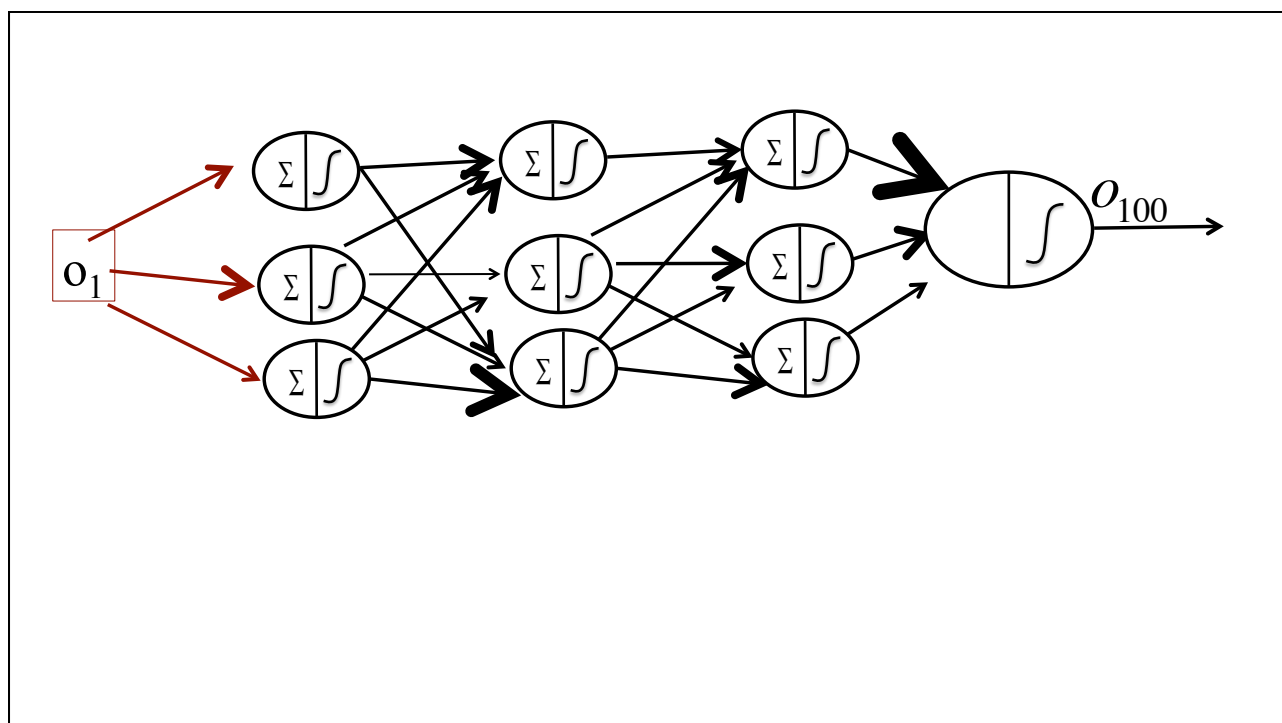
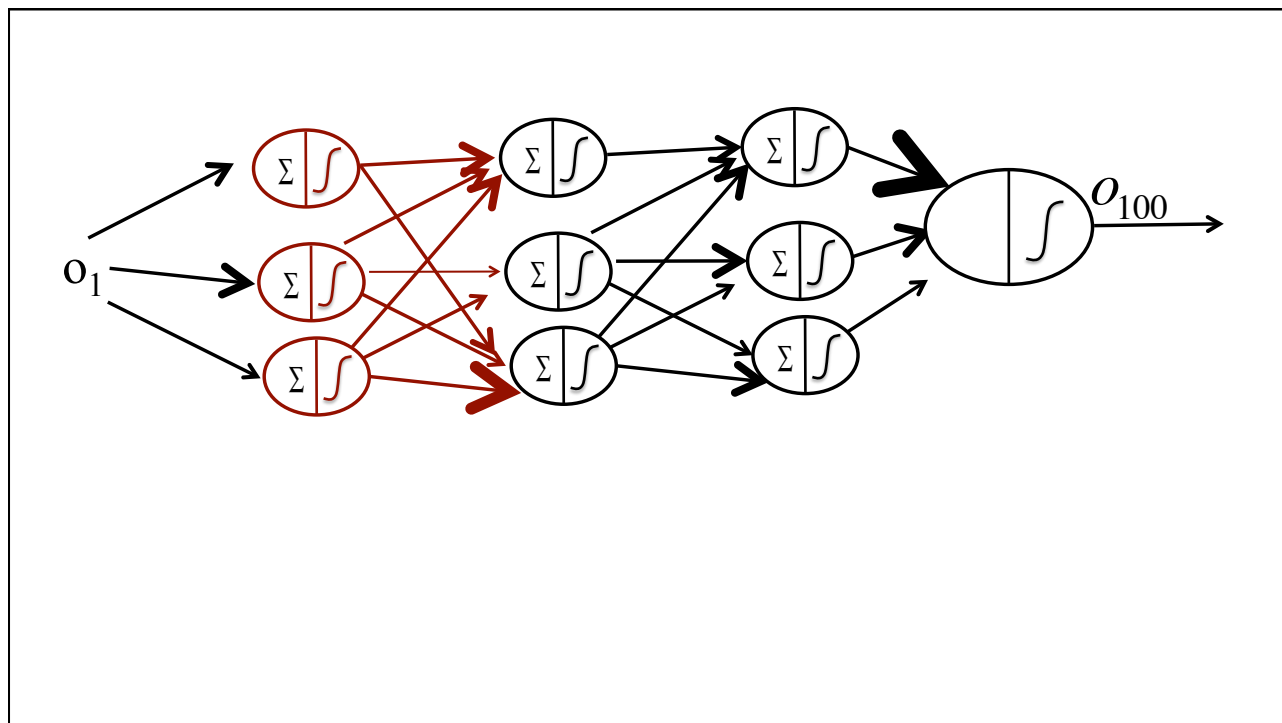
$$\frac{dE}{dw_{a,b}} = \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} \frac{d \text{net}_b}{dw_{a,b}}$$

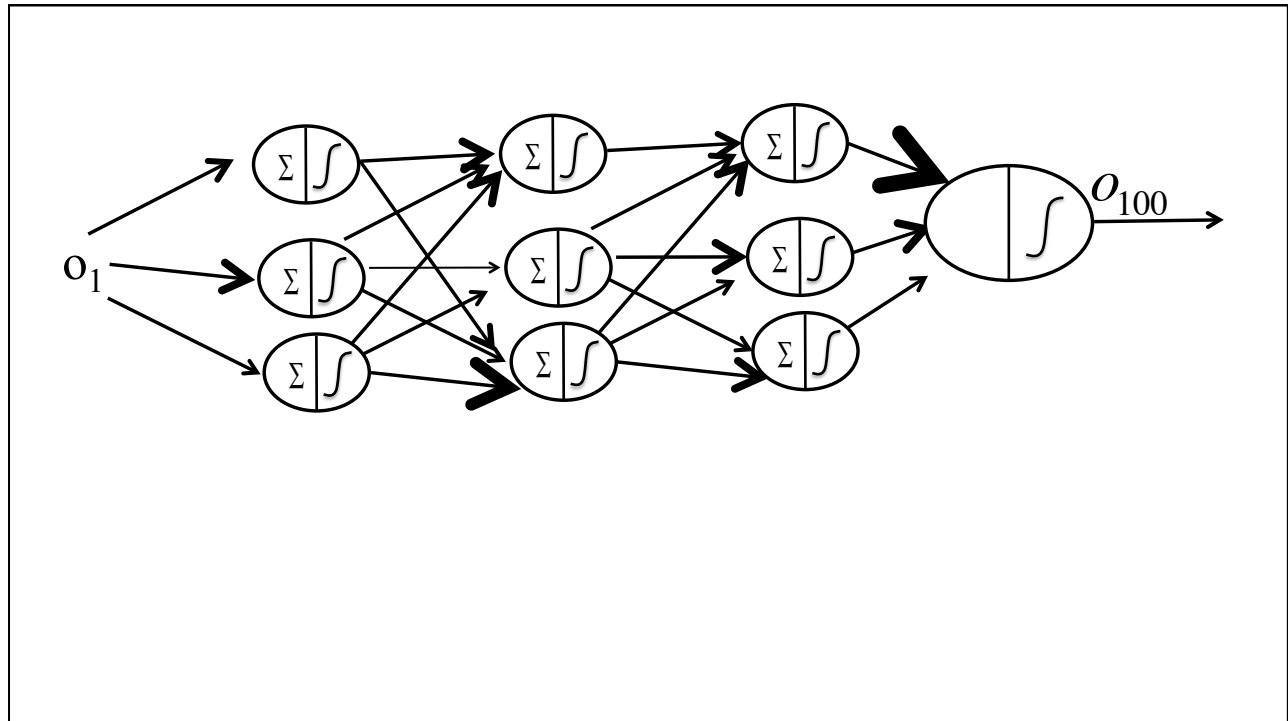
$$= \left( \sum_{\ell \in L} \delta_{\ell} w_{b,\ell} \right) o_b (1 - o_b) o_a$$

The L are downstream. We must have already computed all the  $\delta_{\ell}$ 's ahead of us to compute this.







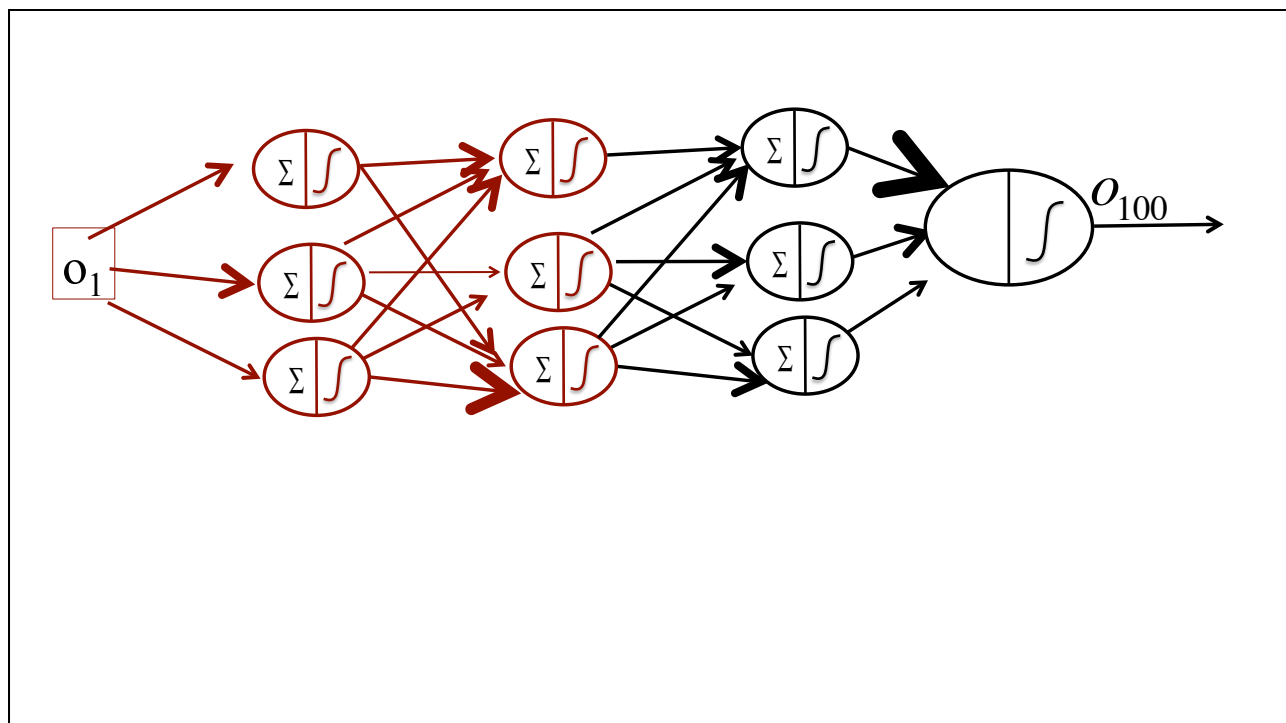
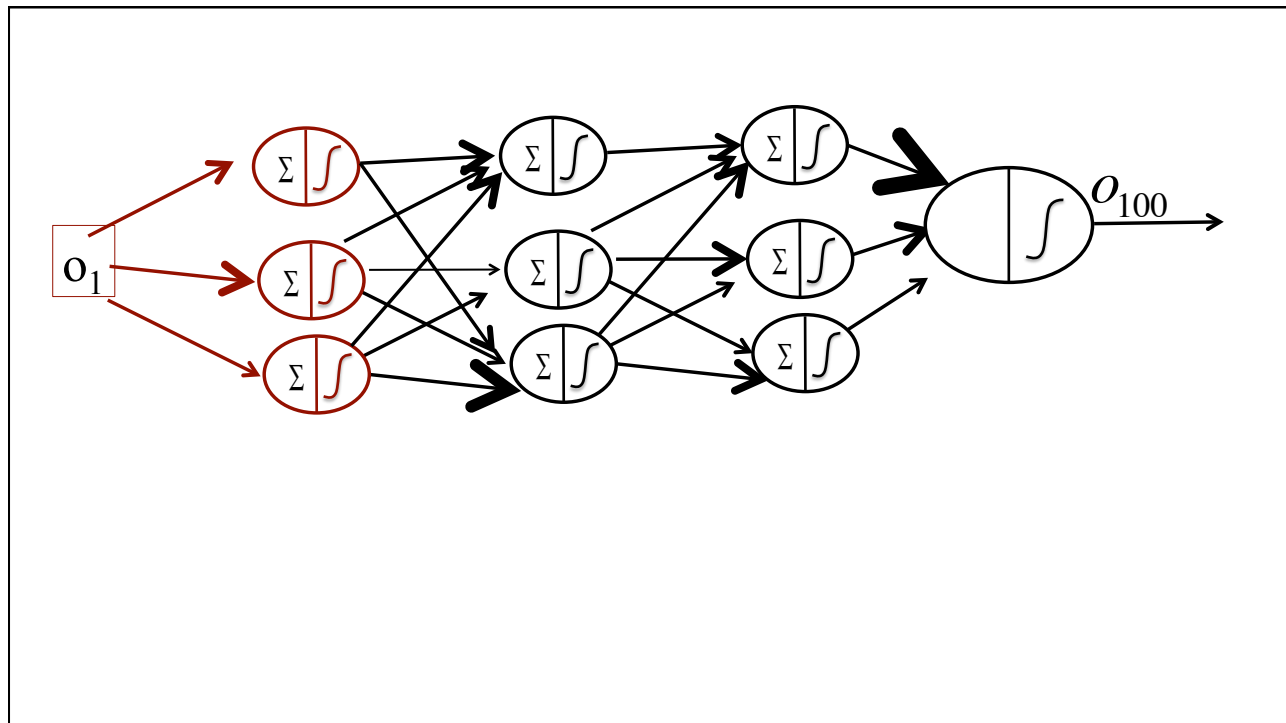


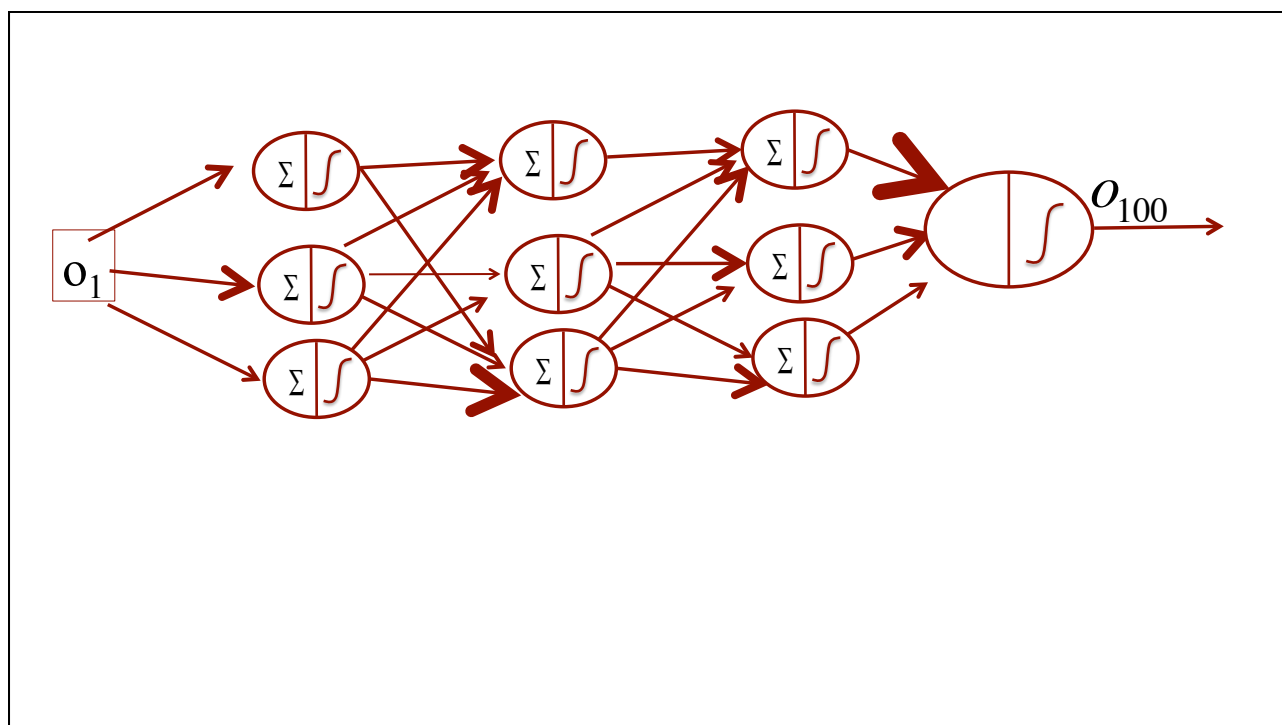
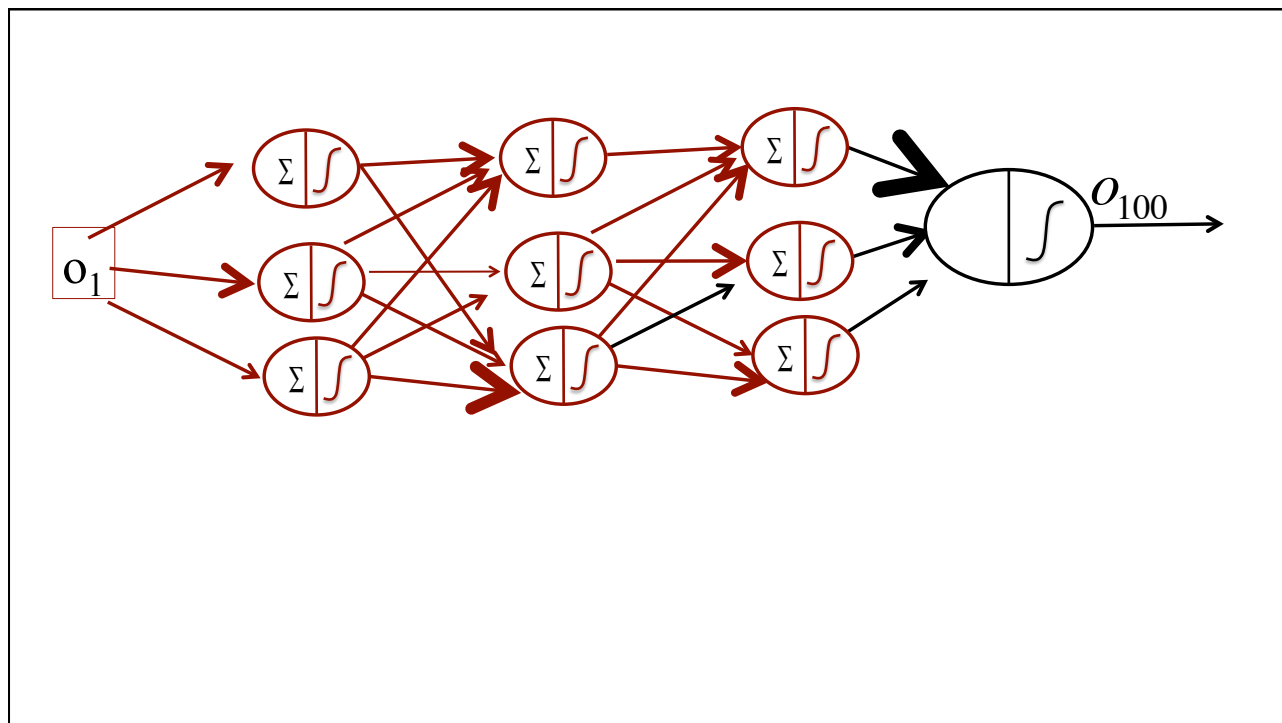
## Backpropagation

- Now we know how to compute  $\frac{dE}{dw_{a,b}}$  for all of the  $w_{a,b}$ 's.
- Let's do gradient descent.

$$w_{a,b} \longleftarrow w_{a,b} - \alpha \frac{dE}{dw_{a,b}}$$

- $\alpha$  is between 0 and 1. Called the “learning rate”.
- Now we know how to propagate errors back through the network.
- Remember how to go forward?







## Backpropagation

- Repeat going backwards (to calculate the gradients), adjusting the weights, and going forwards (to calculate the errors) over and over in order to learn.

## Neural networks

- Advantages:
  - highly expressive nonlinear models
  - have advances in computer vision and speech that other methods have not achieved
  - can capture latent structure within the hidden layers
- Disadvantages
  - can get stuck in local optima, could produce bad solutions
  - black box
  - lots of tuning parameters (e.g., the structure of the network)



©2014 Microsoft Corporation. All rights reserved. Microsoft, Windows, Office, Azure, System Center, Dynamics and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries. The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.