

# Lecture 12: Learning Parameters of Multi-layer Perceptrons with Backpropagation

---

**COMP90049**

**Introduction to Machine Learning**

Semester 1, 2021

Lea Frermann, CIS

Copyright @ University of Melbourne 2021. All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.



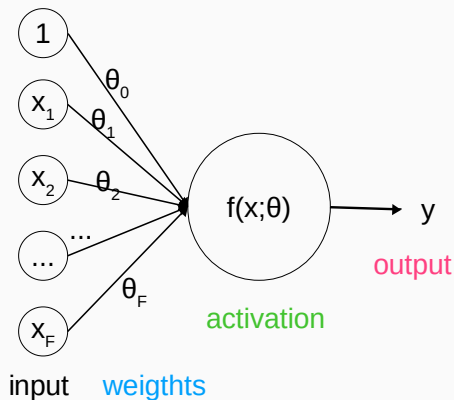
## Last lecture

- From perceptrons to neural networks
- multilayer perceptron
- some examples
- features and limitations

## Today

- Learning parameters of neural networks
- The Backpropagation algorithm

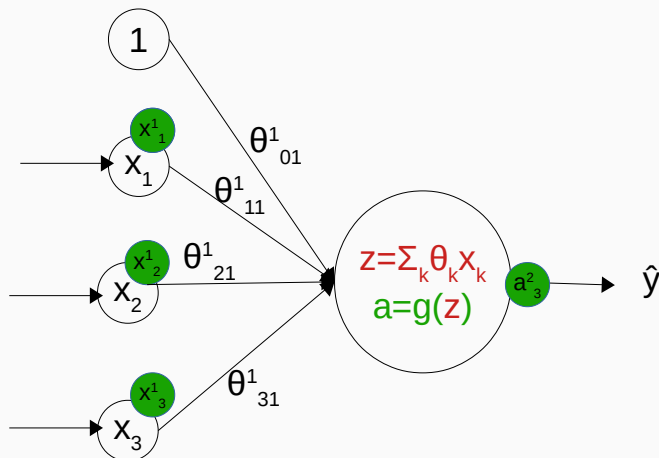
## Recap: Multi-layer perceptrons



$$y = f(\theta^T x)$$

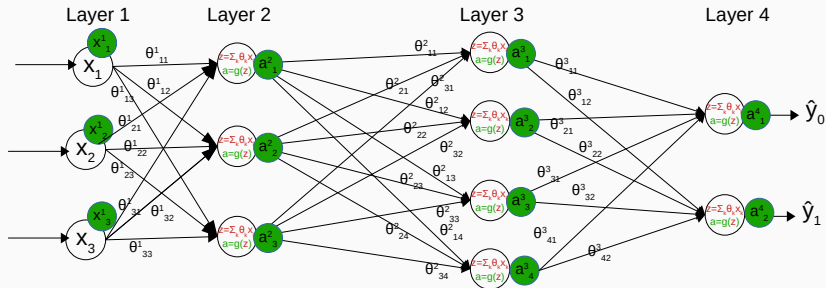
- Linearly separable data
- Perceptron learning rule

## Recap: Multi-layer perceptrons

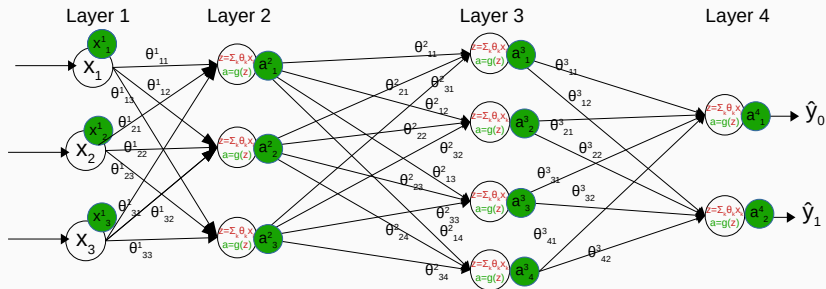


- Linearly separable data
- Perceptron learning rule

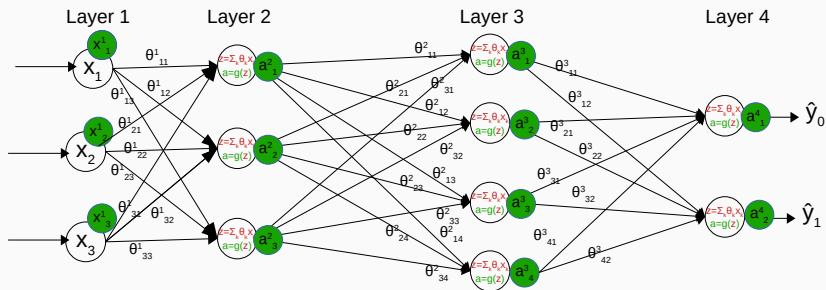
# Recap: Multi-layer perceptrons



## Recall: Supervised learning



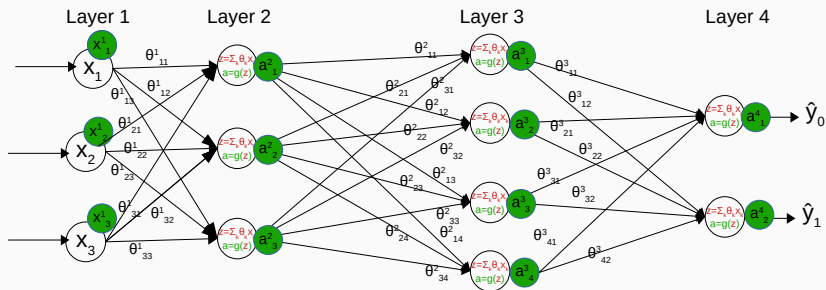
# Recall: Supervised learning



## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP

# Recall: Supervised learning

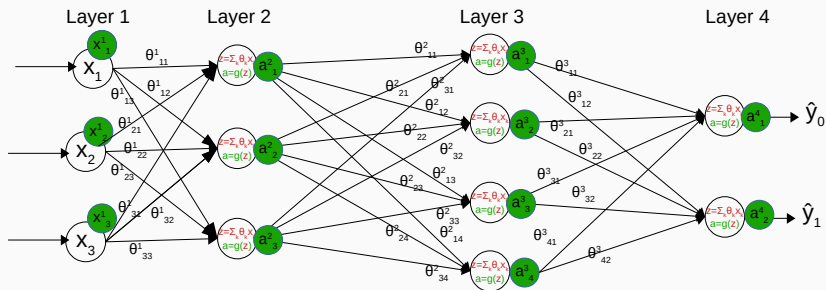


## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**



# Recall: Supervised learning

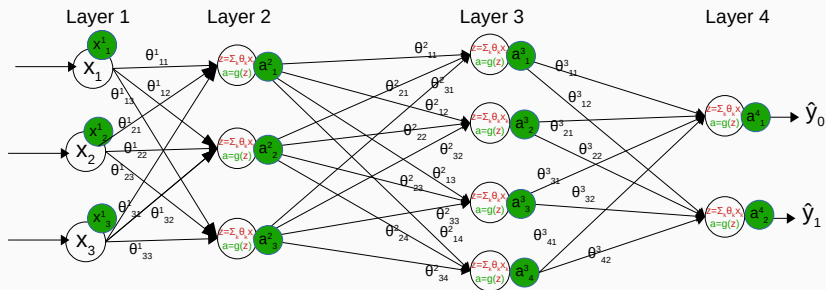


## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**
4. **Modify each weight** such that the error decreases in future predictions (e.g., by applying **gradient descent**)



# Recall: Supervised learning

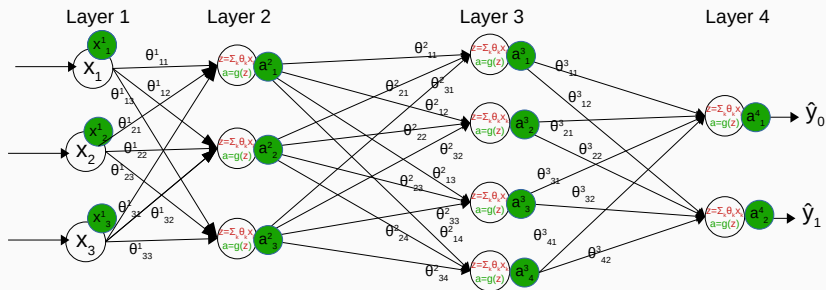


## Recipe

1. Forward propagate an input  $x$  from the **training set**
2. Compute the output  $\hat{y}$  with the MLP
3. Compare predicted output  $\hat{y}$  against true output  $y$ ; compute the **error**
4. **Modify each weight** such that the error decreases in future predictions (e.g., by applying **gradient descent**)
5. Repeat.



# Recall: Optimization with Gradient Descent



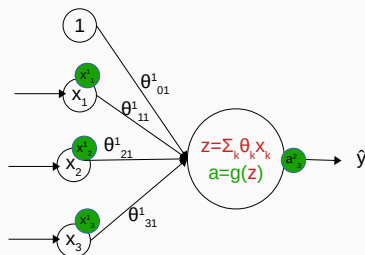
We want to

1. Find the best parameters, which lead to the smallest error  $E$
2. Optimize each model parameter  $\theta_{ij}^l$
3. We will use **gradient descent** to achieve that
4.  $\theta_{ij}^{l,(t+1)} \leftarrow \theta_{ij}^{l,(t)} + \Delta \theta_{ij}^l$



## Recall Perceptron learning:

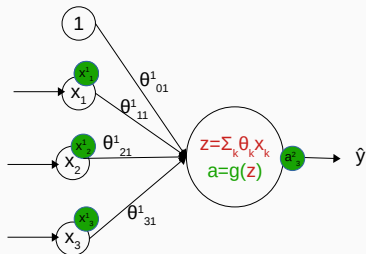
- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



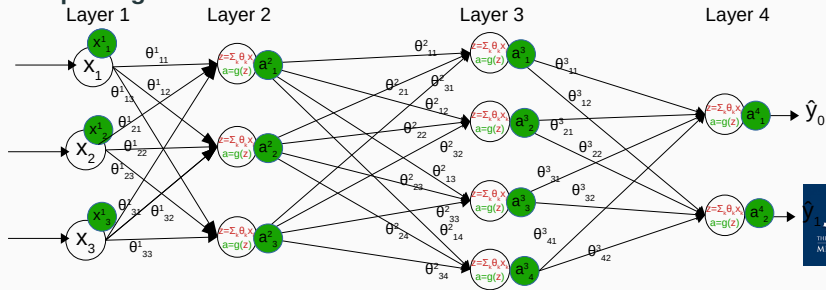
# Towards Backpropagation

## Recall Perceptron learning:

- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$

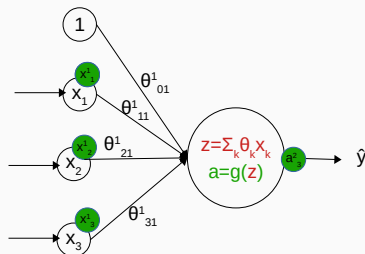


## Compare against the MLP:



## Recall Perceptron learning:

- Pass an input through and compute  $\hat{y}$
- Compare  $\hat{y}$  against  $y$
- Weight update  $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



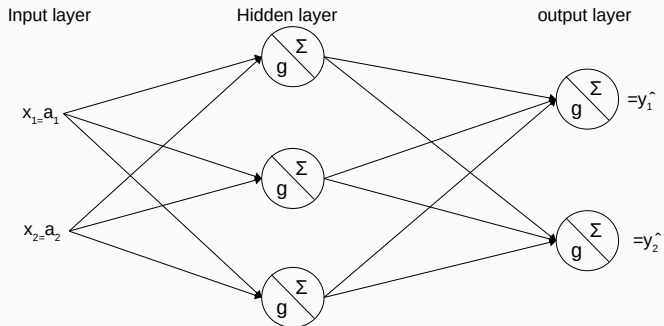
## Problems

- This update rule depends on **true target outputs**  $y$
- We only have access to true outputs for the **final layer**
- We do not know the **true activations** for the **hidden layers**. Can we **generalize** the above rule to also update the hidden layers?

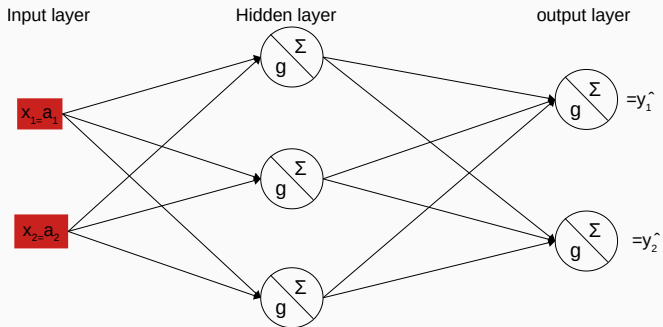
**Backpropagation provides us with an efficient way of computing partial derivatives of the error of an MLP wrt. each individual weight.**



# Backpropagation: Demo



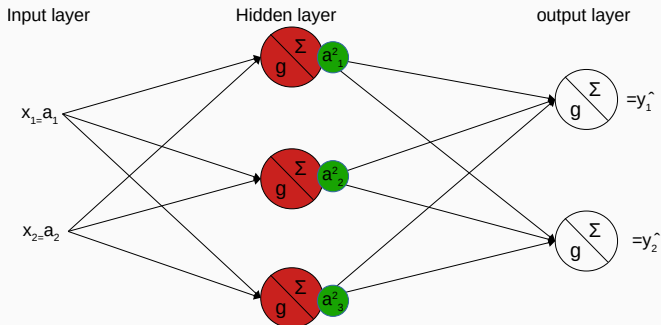
# Backpropagation: Demo



- Receive input

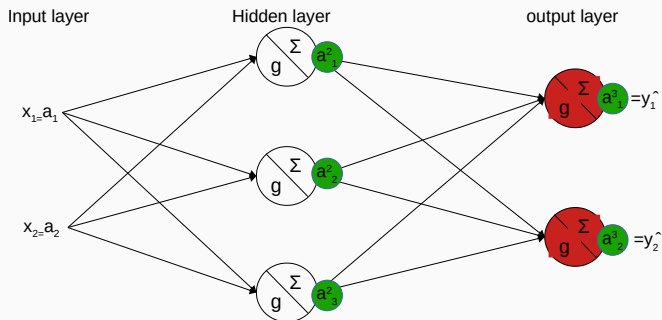


# Backpropagation: Demo



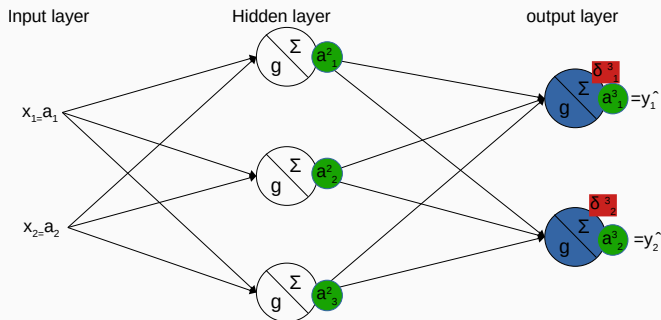
- Receive input
- Forward pass: propagate activations through the network

# Backpropagation: Demo



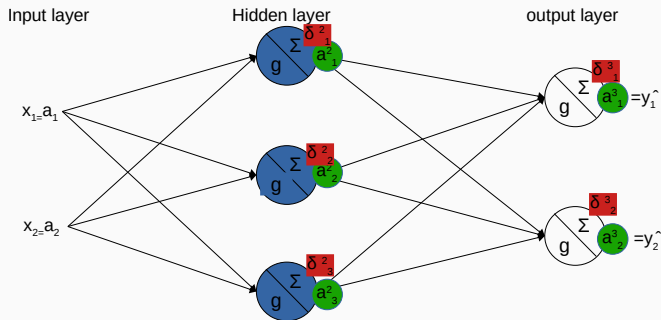
- Receive input
- Forward pass: propagate activations through the network

# Backpropagation: Demo



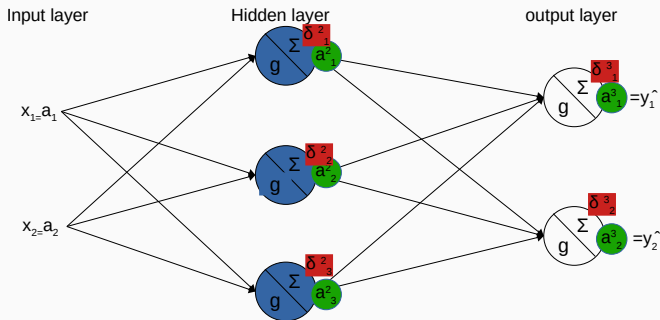
- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate **error terms** through the network

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate **error terms** through the network
- Calculate  $\Delta\theta_{ij}^l$  for all  $\theta_{ij}^l$
- Update weights  $\theta_{ij}^l \leftarrow \theta_{ij}^l + \Delta\theta_{ij}^l$

- We recall what a MLP is
- We recall that we want to learn its parameters such that our prediction error is minimized
- We recall that gradient descent gives us a rule for updating the weights

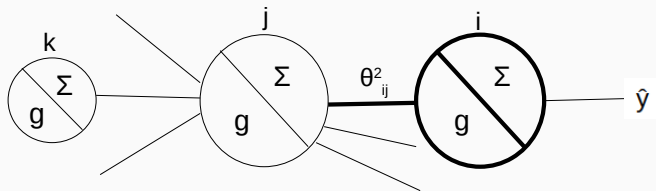
$$\theta_i \leftarrow \theta_i + \Delta\theta_i \text{ with } \Delta\theta_i = -\eta \frac{\partial E}{\partial \theta_i}$$

- But how do we compute  $\frac{\partial E}{\partial \theta_i}$ ?
- **Backpropagation** provides us with an **efficient way** of computing partial derivatives of the error of an MLP wrt. each individual weight.

## The (Generalized) Delta Rule

---

# Backpropagation 1: Model definition



- Assuming a sigmoid activation function, the output of neuron  $i$  (or its activation  $a_i$ ) is

$$a_i = g(z_i) = \frac{1}{1 + e^{-z_i}}$$

- And  $z_i$  is the input of all incoming activations into neuron  $i$

$$z_i = \sum_j \theta_{ij} a_j$$

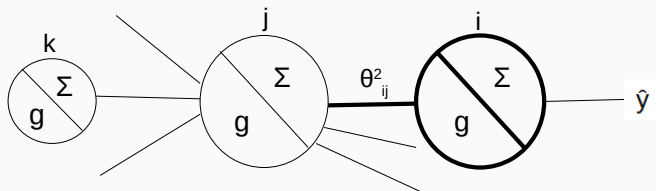
- And Mean Squared Error (MSE) as **error function**  $E$

$$E = \frac{1}{2N} \sum_{i=1}^N (y^i - \hat{y}^i)^2$$





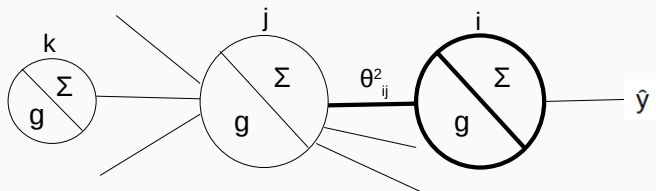
## Backpropagation 2: Error of the final layer



- Apply gradient descent for input  $p$  and weight  $\theta_{ij}^2$  connecting node  $j$  with node  $i$

$$\Delta \theta_{ij}^2 = -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (y^p - \hat{y}^p) g'(z_i) a_j$$

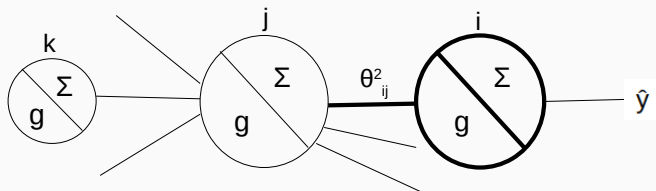
## Backpropagation 2: Error of the final layer



- Apply gradient descent for input  $p$  and weight  $\theta_{ij}^2$  connecting node  $j$  with node  $i$

$$\Delta \theta_{ij}^2 = -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (y^p - \hat{y}^p) g'(z_i) a_j$$

## Backpropagation 2: Error of the final layer

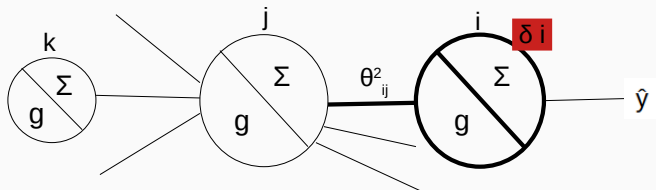


- Apply gradient descent for input  $p$  and weight  $\theta_{ij}^2$  connecting node  $j$  with node  $i$

$$\begin{aligned}\Delta \theta_{ij}^2 &= -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (y^p - \hat{y}^p) g'(z_i) a_j \\ &= \eta \delta_i a_j\end{aligned}$$

- The weight update corresponds to an error term ( $\delta_i$ ) scaled by the incoming activation
- We attach a  $\delta$  to **node**  $i$

# Backpropagation: The Generalized Delta Rule



- The **Generalized Delta Rule**

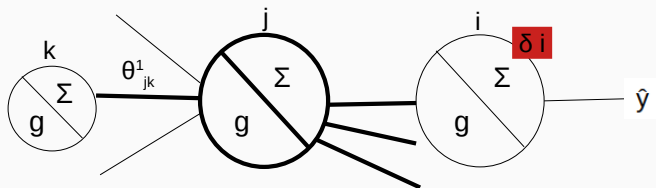
$$\Delta \theta_{ij}^2 = -\eta \frac{\partial E}{\partial \theta_{ij}^2} = \eta (y^p - \hat{y}^p) g'(z_i) a_j = \eta \delta_i a_j$$

$$\delta_i = (y^p - \hat{y}^p) g'(z_i)$$

- The above  $\delta_i$  can only be applied to output units, because it relies on the **target outputs**  $y^p$ .
- We do not have target outputs  $y$  for the intermediate layers



# Backpropagation: The Generalized Delta Rule

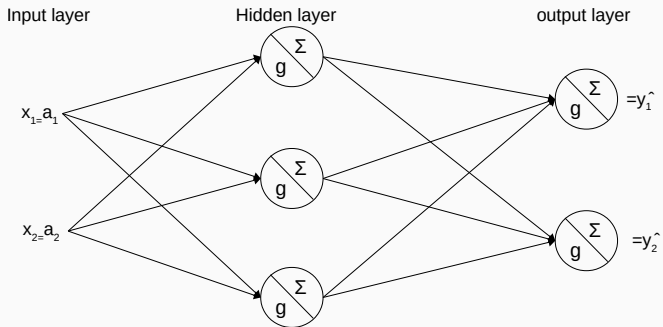


- Instead, we **backpropagate** the errors ( $\delta$ s) from right to left through the network

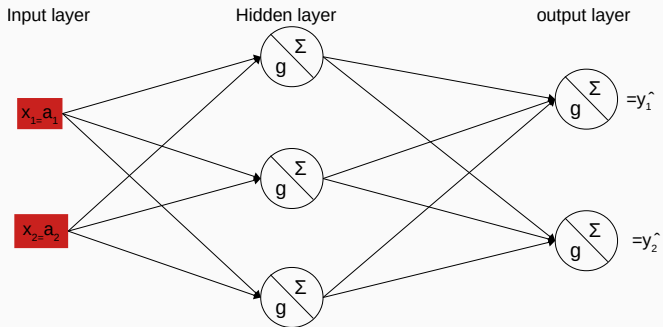
$$\Delta\theta_{jk}^1 = \eta \delta_j a_k$$

$$\delta_j = \sum_i \theta_{ij}^1 \delta_i g'(z_j)$$

# Backpropagation: Demo

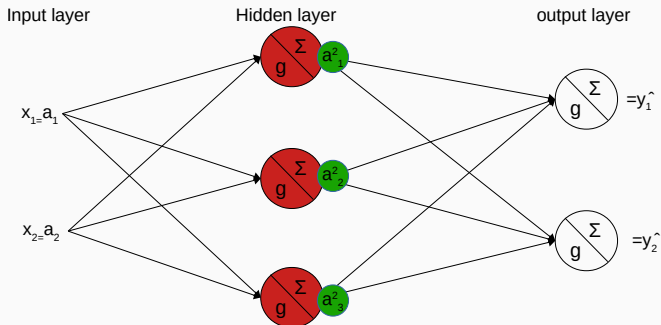


# Backpropagation: Demo



- Receive input

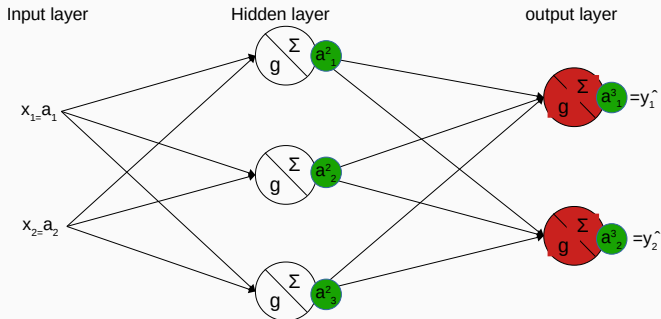
# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network

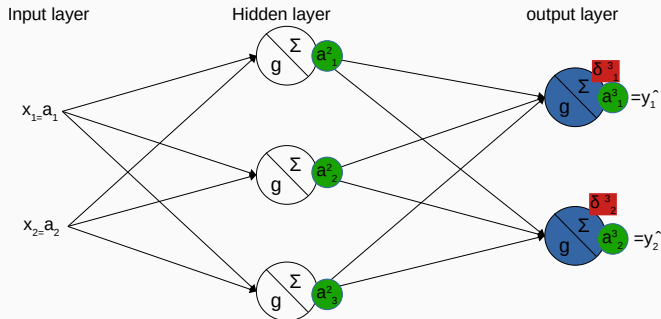


# Backpropagation: Demo



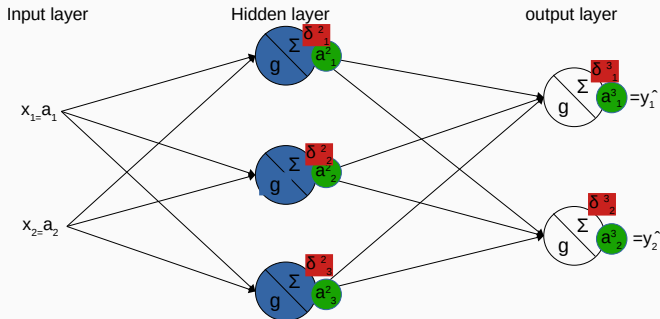
- Receive input
- Forward pass: propagate activations through the network

# Backpropagation: Demo



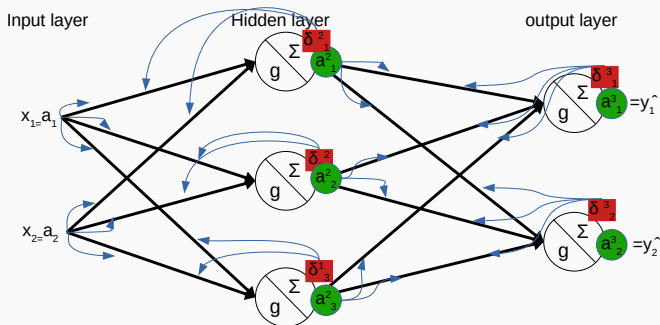
- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate error terms through the network

# Backpropagation: Demo



- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output  $\hat{y}$  against true  $y$
- Backward pass: propagate error terms through the network
- Calculate  $\frac{\partial E}{\partial \theta^l_{ij}}$  for all  $\theta^l_{ij}$
- Update weights  $\theta^l_{ij} \leftarrow \theta^l_{ij} + \Delta \theta^l_{ij}$

# Backpropagation Algorithm

---

Design your neural network

Initialize parameters  $\theta$

**repeat**

**for** training instance  $x_i$  **do**

1. **Forward pass** the instance through the network, compute activations, determine output
2. Compute the **error**
3. Propagate error **back** through the network, and compute for all weights between nodes  $ij$  in all layers  $l$

$$\Delta\theta_{ij}^l = -\eta \frac{\partial E}{\partial \theta_{ij}^l} = \eta \delta_i a_j$$

4. Update **all** parameters **at once**

$$\theta_{ij}^l \leftarrow \theta_{ij}^l + \Delta\theta_{ij}^l$$

**until** stopping criteria reached.

---



## Derivation of the update rules

... optional slides after the next (summary) slide, for those who are interested!

## After this lecture, you be able to understand

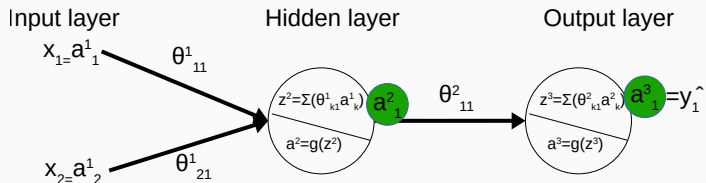
- Why estimation of the MLP parameters is difficult
- How and why we use Gradient Descent to optimize the parameters
- How Backpropagation is a special instance of gradient descent, which allows us to efficiently compute the gradients of all weights wrt. the error
- The mechanism behind gradient descent
- The mathematical justification of gradient descent

## Good job, everyone!

- You now know what (feed forward) neural networks are
- You now know what to consider when designing neural networks
- You now know how to estimate their parameters
- That's more than the average 'data scientist' out there!

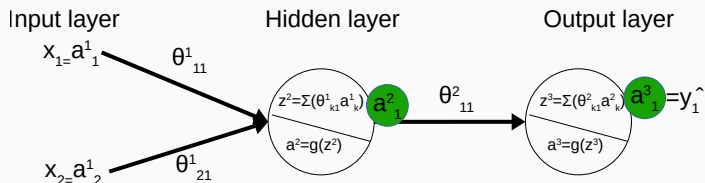


# Backpropagation: Derivation





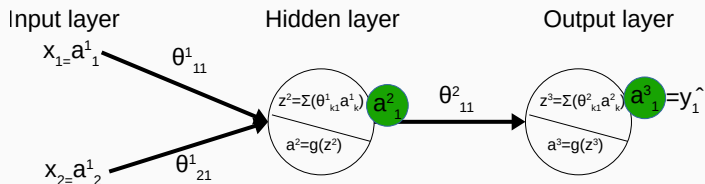
# Backpropagation: Derivation



**Chain of reactions** in the forward pass, focussing on the **output layer**

- varying  $a^2$  causes a change in  $z^3$
- varying  $z^3$  causes a change in  $a^3_1 = g(z^3)$
- varying  $a^3_1 = \hat{y}$  causes a change in  $E(y, \hat{y})$

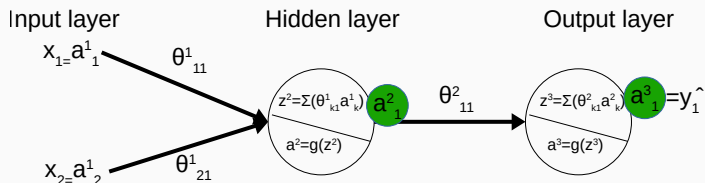
# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

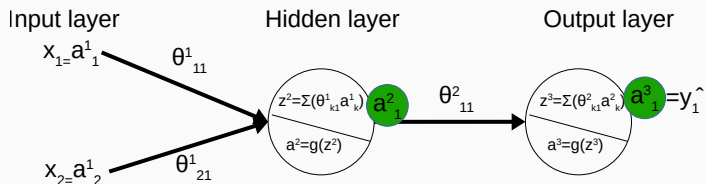
$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

Let's look at each term individually

$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

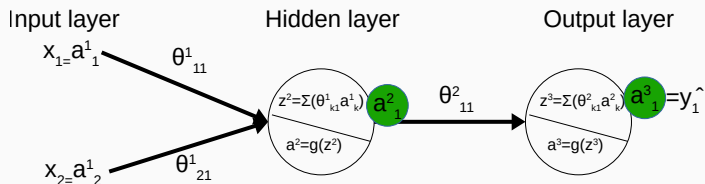
Let's look at each term individually

$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta\theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) =$$

Let's look at each term individually

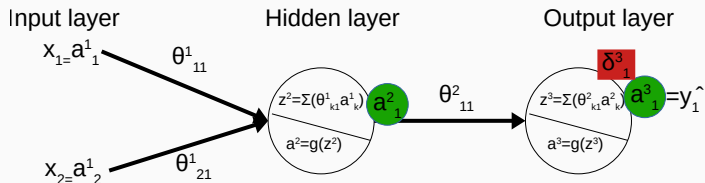
$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$

$$\frac{\partial z}{\partial \theta_{ij}} = \frac{\partial}{\partial \theta_{ij}} \sum_{i'} \theta_{i'j} a_{i'} = \sum_{i'} \frac{\partial}{\partial \theta_{ij}} \theta_{i'j} a_{i'} = a_i$$



# Backpropagation: Derivation



We can use the **chain rule** to capture the behavior of  $\theta^2_{11}$  wrt  $E$

$$\Delta \theta^2 = -\eta \frac{\partial E}{\partial \theta^2} = -\eta \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial \theta^2} \right) = \underbrace{\eta (y - a^3_1) (g'(z^3))}_{= \delta^3_1} (a^2_1) = \eta \delta^3_1 a^2_1$$

Let's look at each term individually

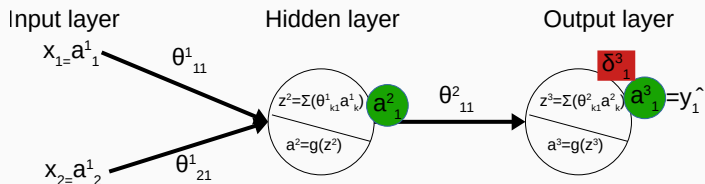
$$\frac{\partial E}{\partial a_i} = -(y_i - a_i) \quad \text{recall that } E = \sum_i^N \frac{1}{2} (y_i - a_i)^2$$

$$\frac{\partial a}{\partial z} = \frac{\partial g(z)}{\partial z} = g'(z)$$

$$\frac{\partial z}{\partial \theta_{ij}} = \frac{\partial}{\partial \theta_{ij}} \sum_{i'} \theta_{i'j} a_{i'} = \sum_{i'} \frac{\partial}{\partial \theta_{ij}} \theta_{i'j} a_{i'} = a_i$$



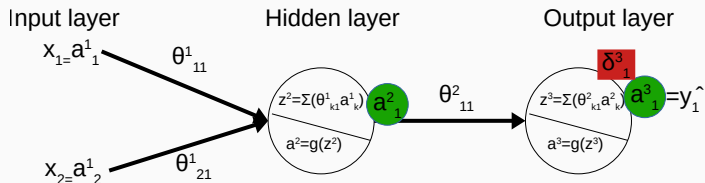
# Backpropagation: Derivation



We have another **chain reaction**. Let's consider **layer 2**

- varying any  $\theta^1_{k1}$  causes a change in  $z^2$
- varying  $z^2$  causes a change in  $a^2_1 = g(z^2)$
- varying  $a^2_1$  causes a change in  $z^3$  (we consider  $\theta^2$  fixed for the moment)
- varying  $z^3$  causes a change in  $a^3_1 = g(z^3)$
- varying  $a^3_1 = \hat{y}$  causes a change in  $E(y, \hat{y})$

# Backpropagation: Derivation

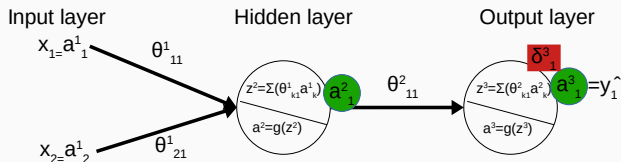


Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

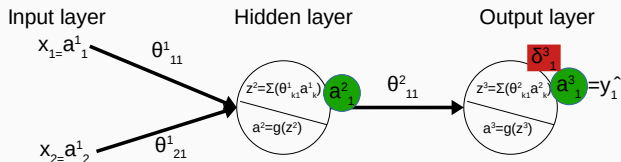
We already know that

$$\frac{\partial E}{\partial a^3_1} = -(y - a^3_1)$$

$$\frac{\partial a^3_1}{\partial z^3} = g'(z^3)$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

And following the previous logic, we can calculate that

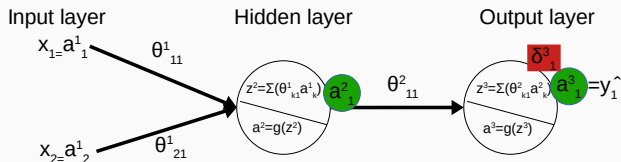
$$\frac{\partial z^3}{\partial a^2_1} = \frac{\partial \theta^2_{11} a^2_1}{\partial a^2_1} = \theta^2_{11}$$

$$\frac{\partial a^2_1}{\partial z^2} = \frac{\partial g(z^2)}{\partial z^2} = g'(z^2)$$

$$\frac{\partial z^2}{\partial \theta^1_{k1}} = a_k$$



# Backpropagation: Derivation



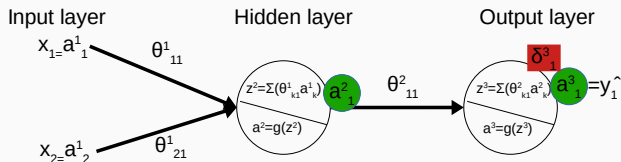
Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$-\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( - (y - a^3_1) g'(z^3) \theta^2_{11} \right) g'(z^2) a_k$$

# Backpropagation: Derivation



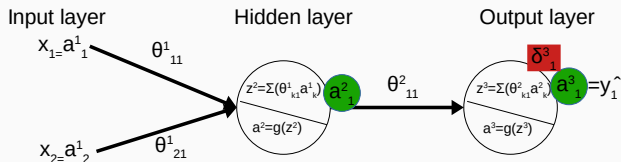
Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( - (y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \left( (y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \end{aligned}$$

# Backpropagation: Derivation



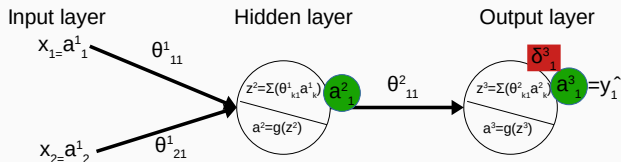
Formulating this again as the chain rule

$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( - (y - a^3_1) g'(z^3) \theta^2_{11} \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{(y - a^3_1) g'(z^3) \theta^2_{11}}_{= \delta^3_1} \right) g'(z^2) a_k \\ &= \delta^3_1 \end{aligned}$$

# Backpropagation: Derivation



Formulating this again as the chain rule

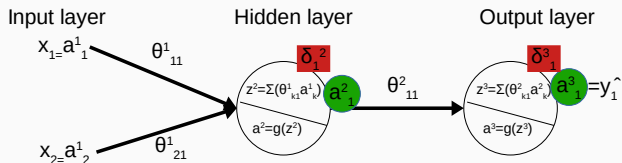
$$\Delta\theta^1_{k1} = -\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= -\eta \left( - (y - a^3_1) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{(y - a^3_1) g'(z^3) \theta^2}_{=\delta^3_1} \right) g'(z^2) a_k = \eta \left( \delta^3_1 \theta^2 \right) g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

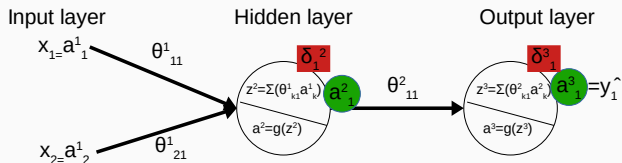
$$\Delta\theta_{k1}^1 = -\eta \frac{\partial E}{\partial \theta_{k1}^1} = -\eta \left( \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a_1^2} \right) \right) \left( \frac{\partial a_1^2}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta_{k1}^1} \right)$$

Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta_{k1}^1} &= -\eta \left( - (y - a_1^3) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{(y - a_1^3) g'(z^3) \theta^2}_{= \delta_1^3} \right) g'(z^2) a_k = \eta \left( \underbrace{\delta_1^3 \theta^2}_{= \delta_1^2} \right) g'(z^2) a_k \end{aligned}$$



# Backpropagation: Derivation



Formulating this again as the chain rule

$$\Delta \theta_{k1}^1 = -\eta \frac{\partial E}{\partial \theta_{k1}^1} = -\eta \left( \left( \frac{\partial E}{\partial a_1^3} \right) \left( \frac{\partial a_1^3}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a_1^2} \right) \right) \left( \frac{\partial a_1^2}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta_{k1}^1} \right)$$

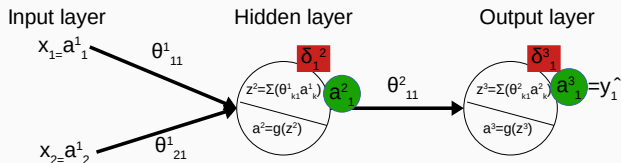
Plugging these into the above we get

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta_{k1}^1} &= -\eta \left( - (y - a_1^3) g'(z^3) \theta^2 \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{(y - a_1^3) g'(z^3) \theta^2}_{= \delta_1^3} \right) g'(z^2) a_k = \eta \left( \underbrace{\delta_1^3 \theta^2}_{= \delta_1^2} \right) g'(z^2) a_k = \eta \delta_1^2 a_k \end{aligned}$$





# Backpropagation: Derivation



Formulating this again as the chain rule

$$-\eta \frac{\partial E}{\partial \theta^1_{k1}} = -\eta \left( \left( \frac{\partial E}{\partial a^3_1} \right) \left( \frac{\partial a^3_1}{\partial z^3} \right) \left( \frac{\partial z^3}{\partial a^2_1} \right) \right) \left( \frac{\partial a^2_1}{\partial z^2} \right) \left( \frac{\partial z^2}{\partial \theta^1_{k1}} \right)$$

If we had more than one weight  $\theta^2$

$$\begin{aligned} -\eta \frac{\partial E}{\partial \theta^1_{k1}} &= \eta \left( \sum_j \underbrace{(y_j - a^3_j) g'(z^3_j) \theta^2_{1j}}_{= \delta^3_j} \right) g'(z^2) a_k \\ &= \eta \left( \underbrace{\sum_j \delta^3_j \theta^2_{1j}}_{= \delta^2_1} \right) g'(z^2) a_k = \eta \delta^2_1 a_k \end{aligned}$$

