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

COMP90049
Introduction to Machine Learning

Semester 1, 2021

Lea Frermann, CIS

Roadmap

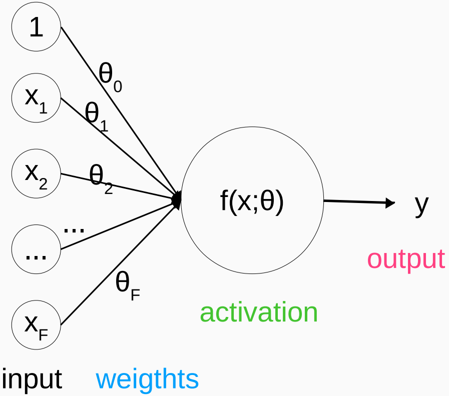
Last lecture

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

Today

- Learning parameters of neural networks
- The Backpropagation algorithm

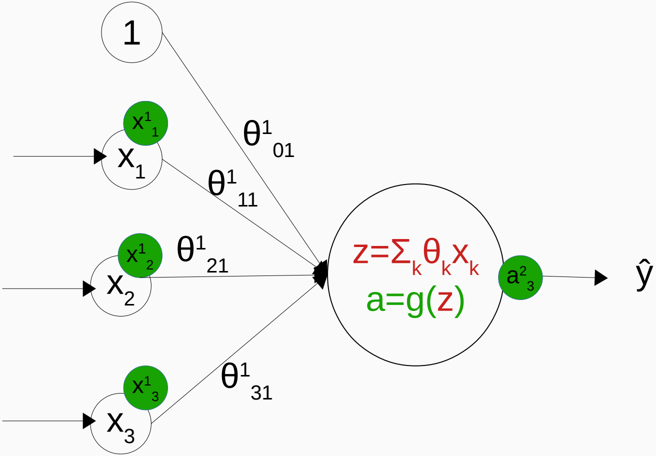
Recap: Multi-layer perceptrons



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

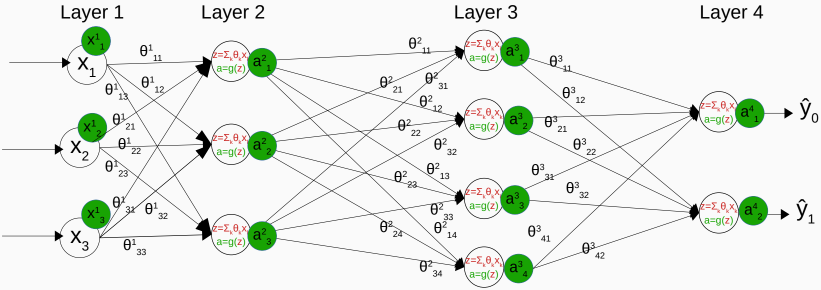
- Linearly separable data
- Perceptron learning rule

Recap: Multi-layer perceptrons

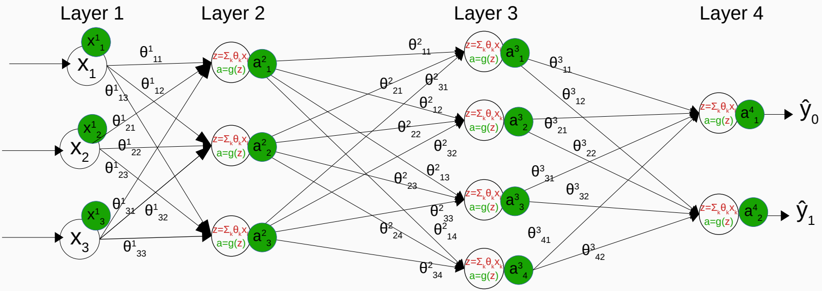


- Linearly separable data
- Perceptron learning rule

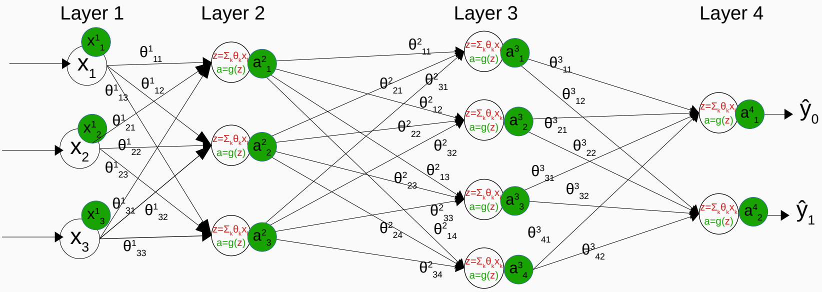
Recap: Multi-layer perceptrons



Recall: Supervised learning



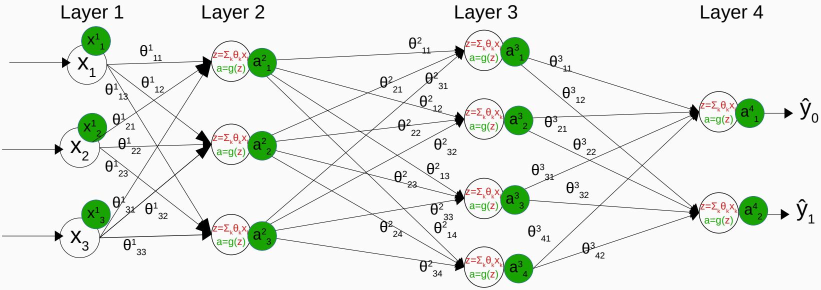
Recall: Supervised learning



Recipe

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

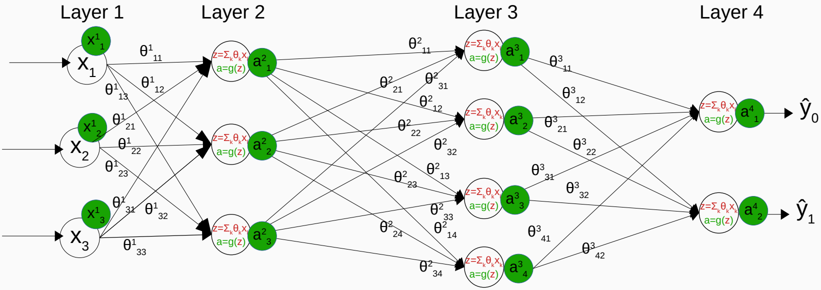
Recall: Supervised learning



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**

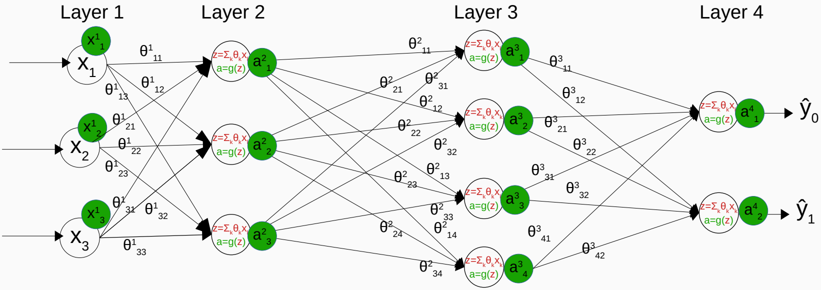
Recall: Supervised learning



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**)

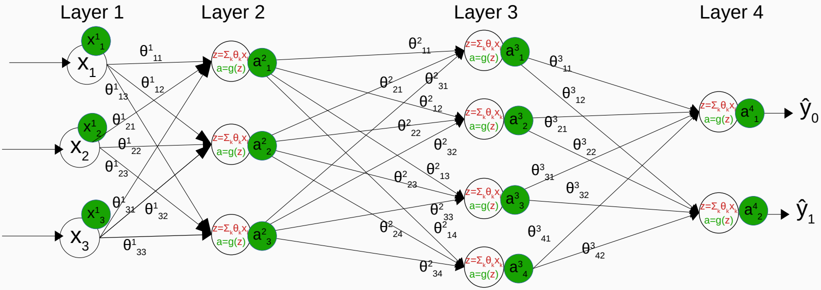
Recall: Supervised learning


Layer 1
Layer 2
Layer 3
Layer 4
x^1_1
x^1_2
x^1_3
X_1
X_2
X_3
θ^1_{11} θ^1_{12} θ^1_{13} θ^1_{21} θ^1_{22} θ^1_{23} θ^1_{31} θ^1_{32} θ^1_{33}
$z=\Sigma_k\theta_k x_k$
$a=g(z)$
a^2_1 a^2_2 a^2_3
θ^2_{11} θ^2_{21} θ^2_{31} θ^2_{12} θ^2_{22} θ^2_{32} θ^2_{13} θ^2_{23} θ^2_{33} θ^2_{24} θ^2_{14} θ^2_{34}
a^3_1 a^3_2 a^3_3 a^3_4
θ^3_{11} θ^3_{12} θ^3_{21} θ^3_{22} θ^3_{31} θ^3_{32} θ^3_{41} θ^3_{42}
a^4_1 a^4_2
$\hat{y}_0$
$\hat{y}_1$

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



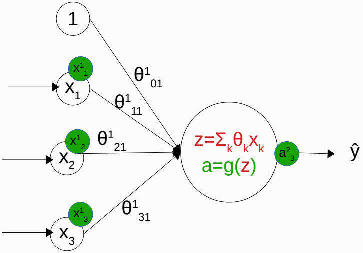
We want to

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

Towards Backpropagation

Recall Perceptron learning:

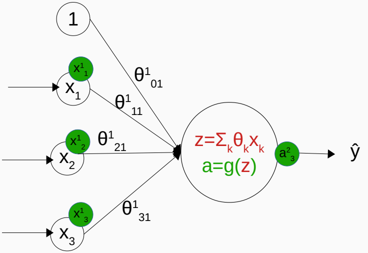
- Pass an input through and compute $\hat{y}$
- Compare $\hat{y}$ against y
- Weight update $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



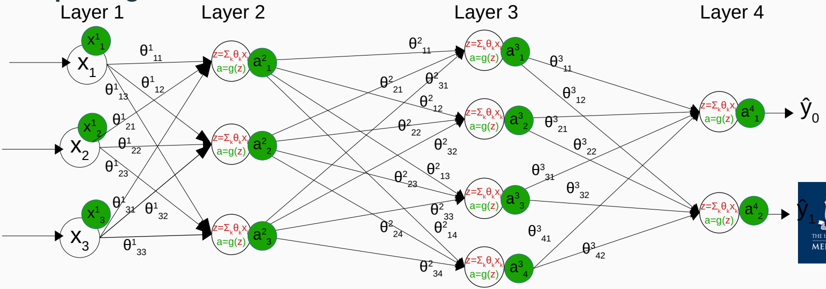
Towards Backpropagation

Recall Perceptron learning:

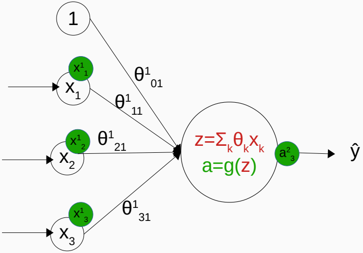
- Pass an input through and compute $\hat{y}$
- Compare $\hat{y}$ against y
- Weight update $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$



Compare against the MLP:



Towards Backpropagation



Recall Perceptron learning:

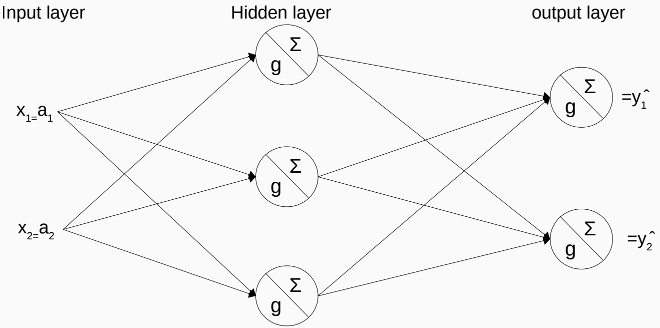
- Pass an input through and compute $\hat{y}$
- Compare $\hat{y}$ against y
- Weight update $\theta_i \leftarrow \theta_i + \eta(y - \hat{y})x_i$

Problems

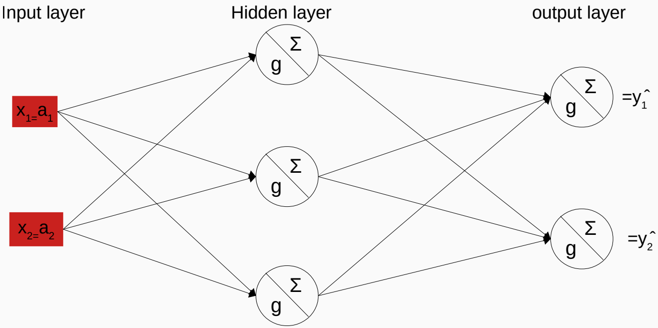
- This update rule depends on **true target outputs** y
- We only have access to true outputs for the **final layer**
- We do not know the **true activations** for the **hidden layers**. Can we **generalize** the above rule to also update the hidden layers?

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

Backpropagation: Demo

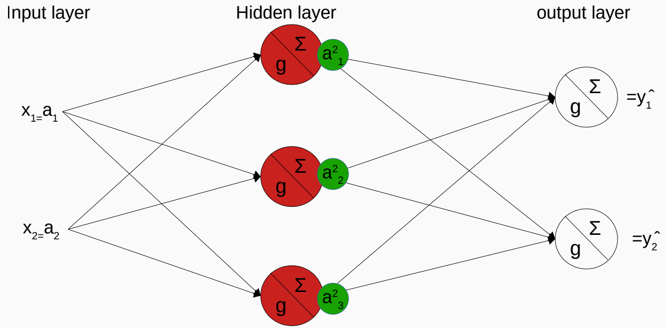


Backpropagation: Demo



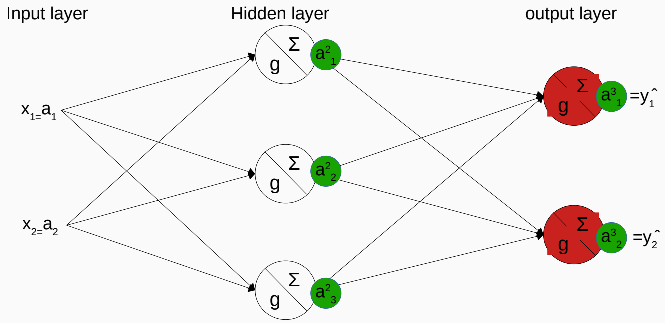
- Receive input

Backpropagation: Demo



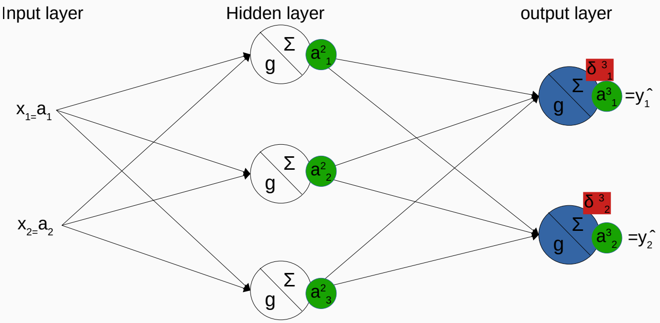
- Receive input
- Forward pass: propagate activations through the network

Backpropagation: Demo



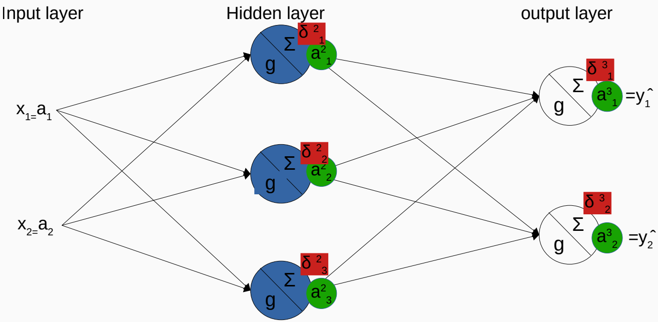
- Receive input
- Forward pass: propagate activations through the network

Backpropagation: Demo



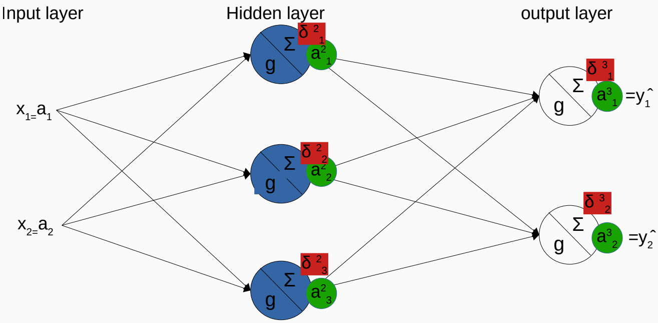
- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output $\hat{y}$ against true y

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 $\Delta\theta^l_{ij}$ for all θ^l_{ij}
- Update weights $\theta^l_{ij} \leftarrow \theta^l_{ij} + \Delta\theta^l_{ij}$

Interim Summary

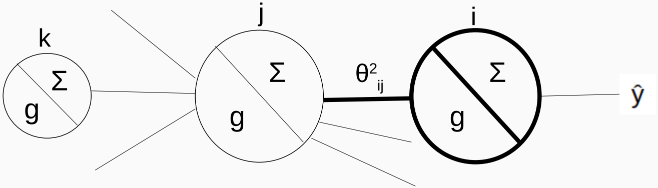
- We recall what a MLP is
- We recall that we want to learn its parameters such that our prediction error is minimized
- We recall that gradient descent gives us a rule for updating the weights

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

- 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

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

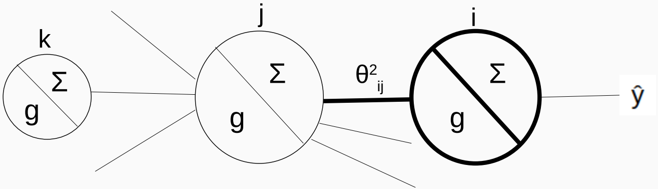
- And z_i is the input of all incoming activations into neuron *i*

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

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

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

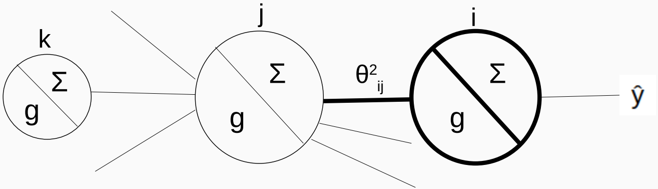
Backpropagation 2: Error of the final layer



- Apply gradient descend for input p and weight θ^2_{ij} connecting node j with node i

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

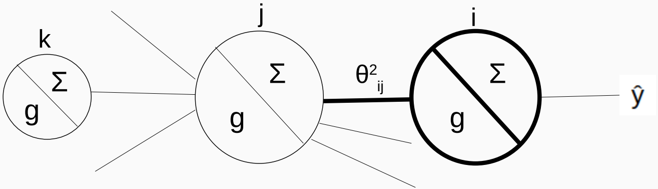
Backpropagation 2: Error of the final layer



- Apply gradient descend for input p and weight θ^2_{ij} connecting node j with node i

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

Backpropagation 2: Error of the final layer



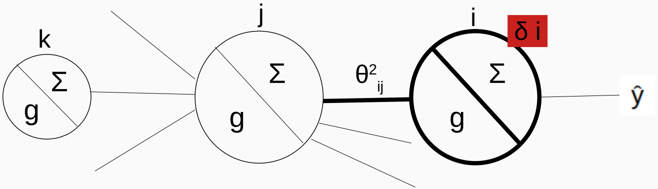
- Apply gradient descend for input *p* and weight θ^2_{ij} connecting node *j* with node *i*

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

$$= \eta\ \delta_i\ a_j$$

- The weight update corresponds to an error term (δ_i) scaled by the incoming activation
- We attach a δ to **node** *i*

Backpropagation: The Generalized Delta Rule



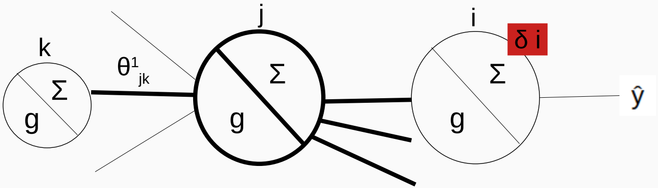
- The Generalized Delta Rule

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

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

- The above δ_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

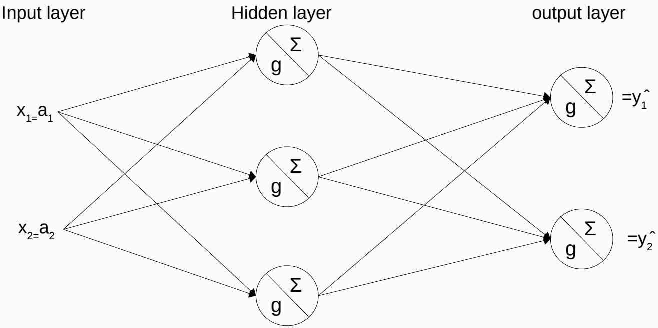


- Instead, we **backpropagate** the errors (δs) from right to left through the network

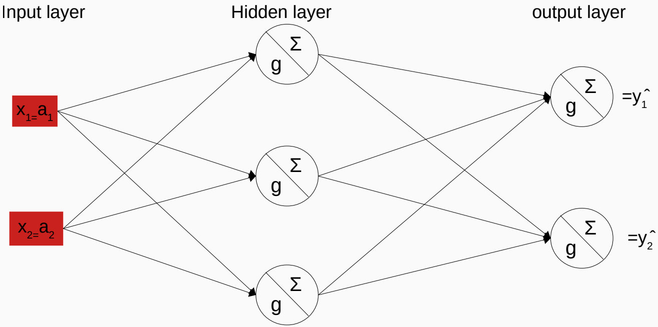
$$\triangle\theta^1_{jk} = \eta\ \delta_j\ a_k$$

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

Backpropagation: Demo

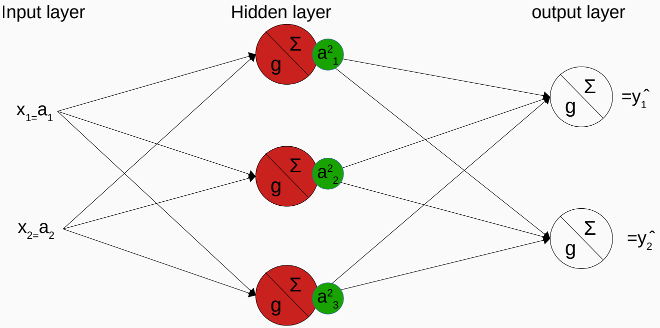


Backpropagation: Demo

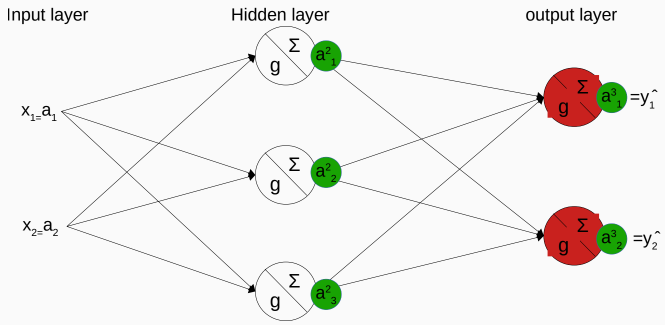


- Receive input

Backpropagation: Demo

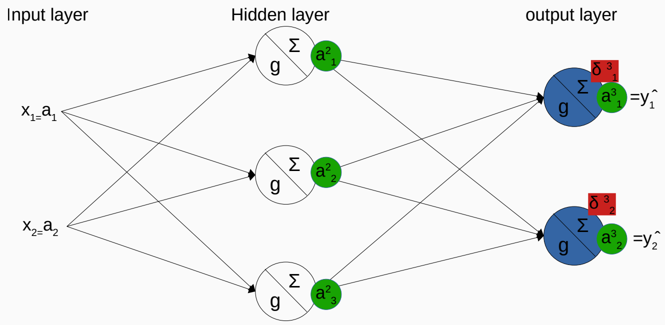


- Receive input
- Forward pass: propagate activations through the network



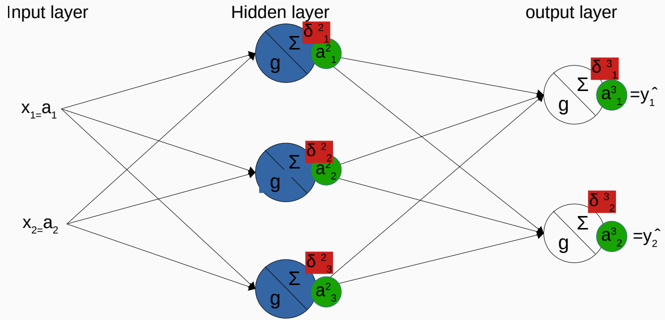
- Receive input
- Forward pass: propagate activations through the network

Backpropagation: Demo



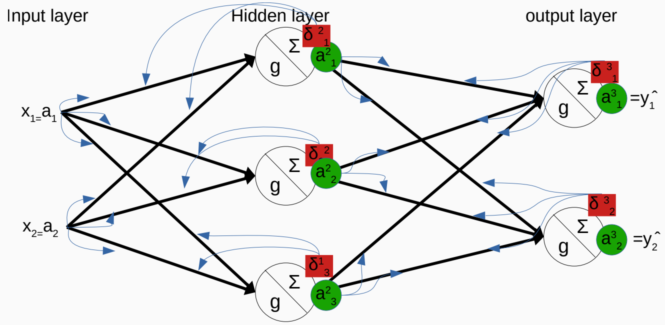
- Receive input
- Forward pass: propagate activations through the network
- Compute Error : compare output $\hat{y}$ against true y

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 θ^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!

Summary

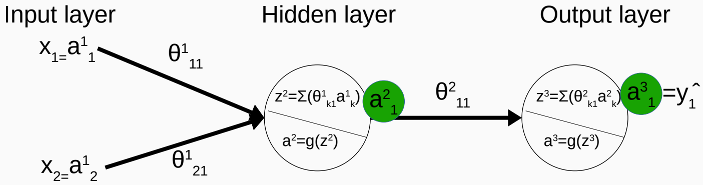
After this lecture, you be able to understand

- 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

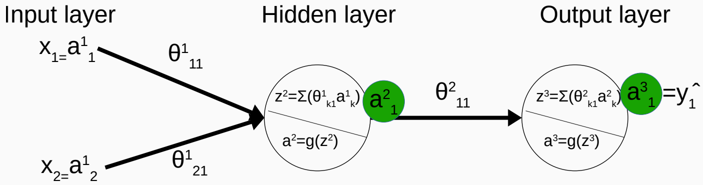
Good job, everyone!

- 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



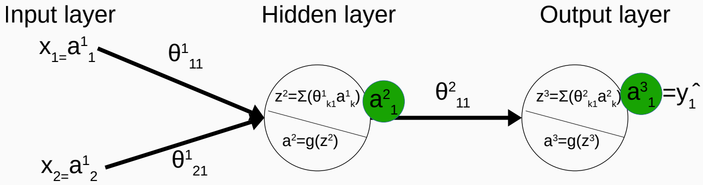
Backpropagation: Derivation



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 $E(y, \hat{y})$

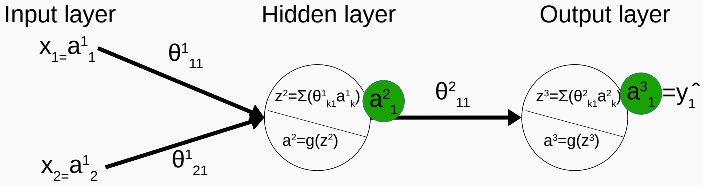
Backpropagation: Derivation



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

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

Backpropagation: Derivation



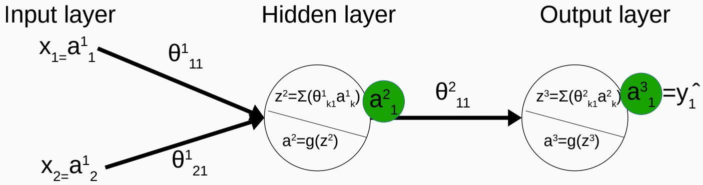
We can use the **chain rule** to capture the behavior of θ^2_{11} wrt E

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

Let’s look at each term individually

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

Backpropagation: Derivation



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

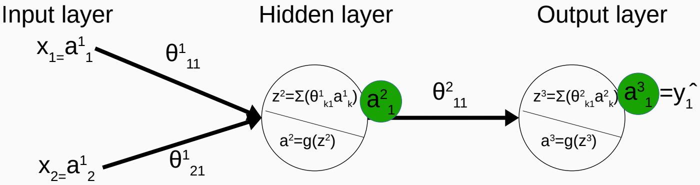
$$\Delta\theta^2 = -\eta\frac{\partial E}{\partial \theta^2} = -\eta\left(\frac{\partial E}{\partial a^3_1}\right)\left(\frac{\partial a^3_1}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial \theta^2}\right) =$$

Let's look at each term individually

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

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

Backpropagation: Derivation



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

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

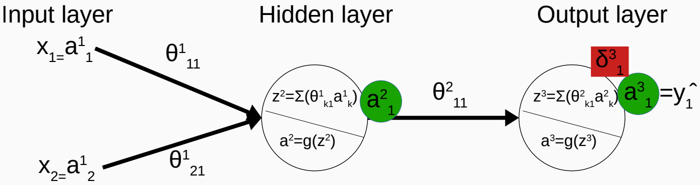
Let's look at each term individually

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

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

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

Backpropagation: Derivation



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

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

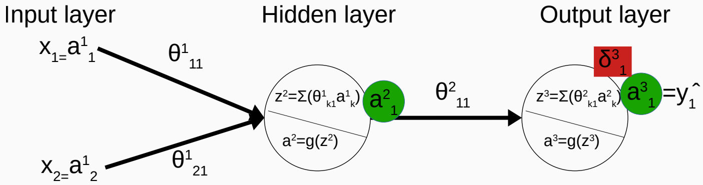
Let's look at each term individually

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

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

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

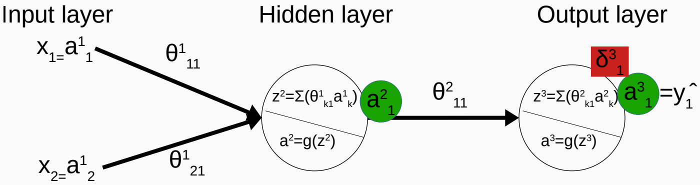
Backpropagation: Derivation



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

- varying any θ^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 θ^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 $E(y, \hat{y})$

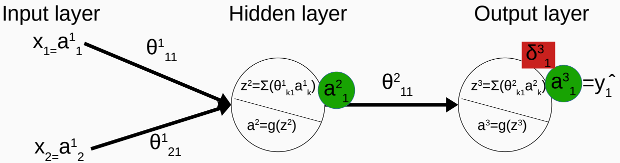
Backpropagation: Derivation



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

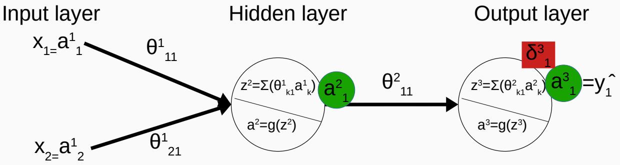
$$\Delta\theta^1_{k1} = -\eta\frac{\partial E}{\partial \theta^1_{k1}} = -\eta\left(\left(\frac{\partial E}{\partial a^3_1}\right)\left(\frac{\partial a^3_1}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial a^2_1}\right)\right)\left(\frac{\partial a^2_1}{\partial z^2}\right)\left(\frac{\partial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

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

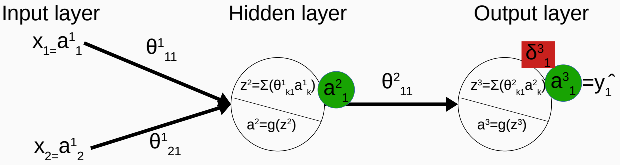
$$\Delta\theta^1_{k1} = -\eta\frac{\partial E}{\partial\theta^1_{k1}} = -\eta\left(\left(\frac{\partial E}{\partial a^3_1}\right)\left(\frac{\partial a^3_1}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial a^2_1}\right)\right)\left(\frac{\partial a^2_1}{\partial z^2}\right)\left(\frac{\partial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

And following the previous logic, we can calculate that

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

$$\frac{\partial a^2_1}{\partial z^2} = \frac{\partial g(z^2)}{\partial z^2} = g'(z^2) \qquad \frac{\partial z^2}{\partial\theta^1_{k1}} = a_k$$

Backpropagation: Derivation



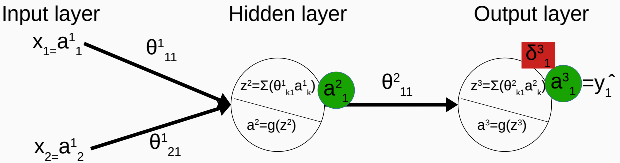
Formulating this again as the chain rule

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

Plugging these into the above we get

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

Backpropagation: Derivation



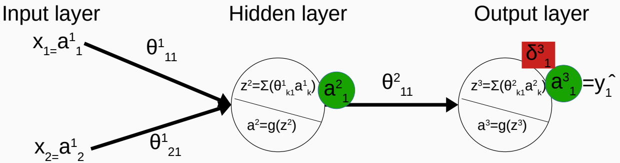
Formulating this again as the chain rule

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

Plugging these into the above we get

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

Backpropagation: Derivation



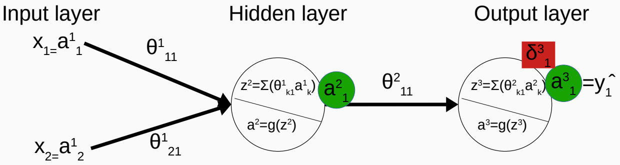
Formulating this again as the chain rule

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

Plugging these into the above we get

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

Backpropagation: Derivation



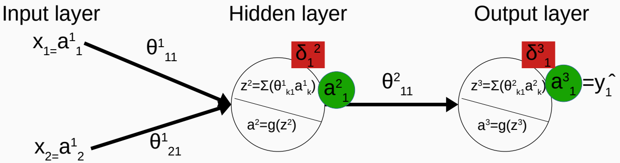
Formulating this again as the chain rule

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

Plugging these into the above we get

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

Backpropagation: Derivation



Formulating this again as the chain rule

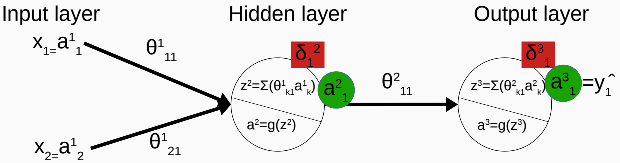
$$\Delta\theta^1_{k1} = -\eta\frac{\partial E}{\partial\theta^1_{k1}} = -\eta\left(\left(\frac{\partial E}{\partial a^3_1}\right)\left(\frac{\partial a^3_1}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial a^2_1}\right)\right)\left(\frac{\partial a^2_1}{\partial z^2}\right)\left(\frac{\partial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

Plugging these into the above we get

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

$$= \eta\Big(\underbrace{(y-a^3_1)g'(z^3)}_{=\delta^3_1}\theta^2\Big)g'(z^2)a_k = \eta\underbrace{\Big(\delta^3_1\theta^2\Big)g'(z^2)}_{=\delta^2_1}a_k$$

Backpropagation: Derivation



Formulating this again as the chain rule

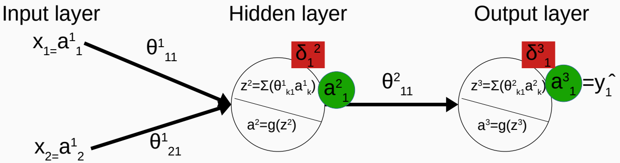
$$\Delta\theta^1_{k1} = -\eta\frac{\partial E}{\partial\theta^1_{k1}} = -\eta\left(\left(\frac{\partial E}{\partial a^3_1}\right)\left(\frac{\partial a^3_1}{\partial z^3}\right)\left(\frac{\partial z^3}{\partial a^2_1}\right)\right)\left(\frac{\partial a^2_1}{\partial z^2}\right)\left(\frac{\partial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

Plugging these into the above we get

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

$$= \eta\Big(\underbrace{(y-a^3_1)g'(z^3)}_{=\delta^3_1}\theta^2\Big)g'(z^2)a_k = \eta\underbrace{\Big(\delta^3_1\theta^2\Big)g'(z^2)}_{=\delta^2_1}a_k = \eta\delta^2_1 a_k$$

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 θ^2

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

$$= \eta\underbrace{\left(\sum_j \delta^3_j\theta^2_{1j}\right)g'(z^2)}_{=\,\delta^2_1}\,a_k \;= \eta\delta^2_1 a_k$$