Imperial College London

Image Classification II Supplementary Notes

Dr Wenjia Bai

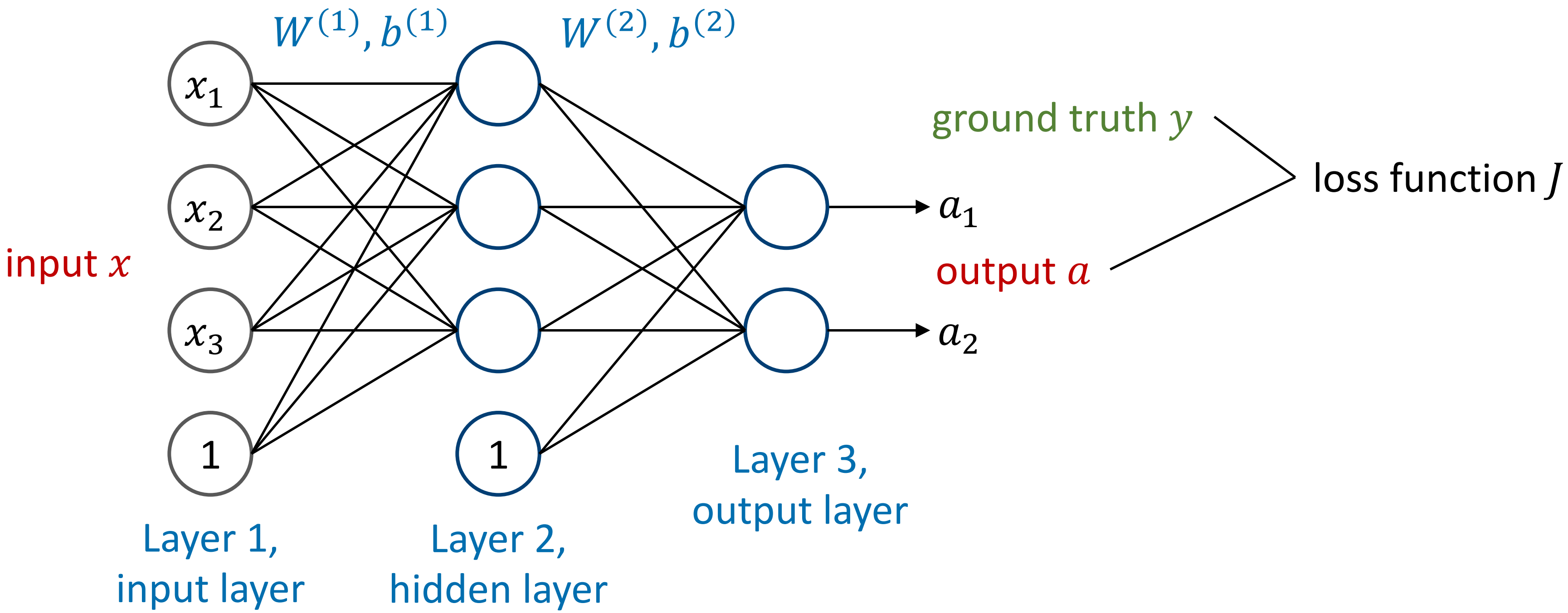
Department of Computing & Brain Sciences

Neural networks

- The mathematics of forward propagation and backpropagation is already covered in other machine learning and deep learning modules, which you can refer to.
- Therefore in the lecture, we only focus on those techniques specific to computer vision.
- Here, we provide some details of the mathematics in case you are still interested in.

Neural networks

- A neural network is formed by putting many neurons into connection, where the output of a neuron can be the input to another.
- It often consists of several layers of neurons.



Multi-layer perceptron (MLP), which is a fully connected multi-layer network.

Neural networks

- To train a neural network, we need a training set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m), \cdots, (x_M, y_M)\}$, which are paired data and ground truth labels.
- We would like to find parameters W and b so that given input x, the output of the network a matches y as much as possible.
- For example, we can define a loss function like this,

$$J(W, b) = \frac{1}{M} \underbrace{\sum_{m=1}^{M}}_{m\text{-th sample}} \frac{1}{2} \|a_m - y_m\|^2$$

Optimising a neural network

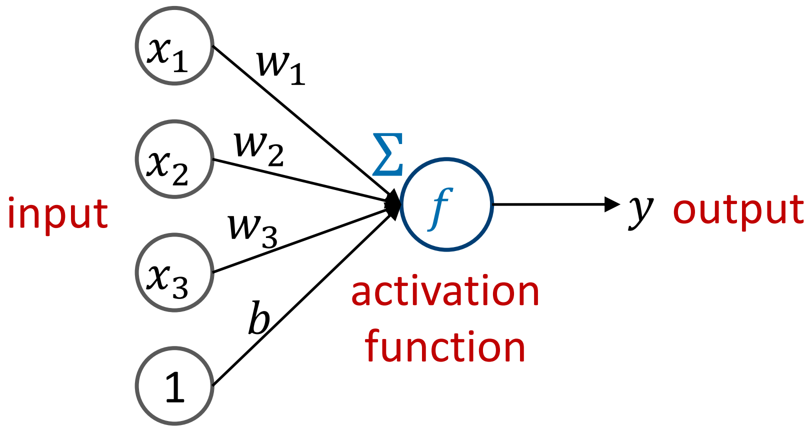
- There are two technical questions here.
- Q1: How do we calculate the output of the network a given input x?
  - A1: We use **forward propagation**.

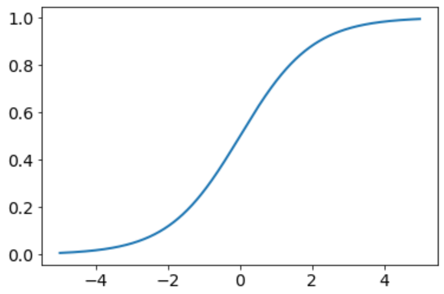
- Q2: How do we find parameters W and b that minimise the loss function, so that a matching ground truth y as much as possible?
  - A2: We use gradient descent for optimisation and **backpropagation** to calculate the required gradient.

We know how a single neuron works

- The neuron is a computational unit that takes an input, applies an activation function and generates an output.

$$y = f(\sum_{i=1}^{3} w_i x_i + b)$$



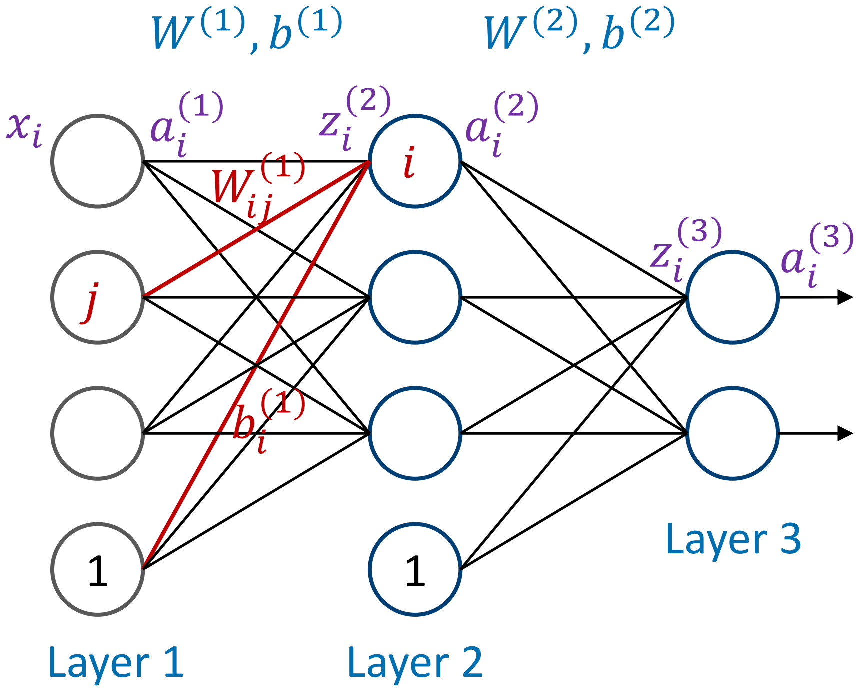


Sigmoid activation function

It works the same for multi-layer perceptron

- At Layer 1, the circles represent the input

$$a_i^{(1)} = x_i$$



It works the same for multi-layer perceptron

- At Layer 2, we calculate the input to each neuron, then apply the activation function.

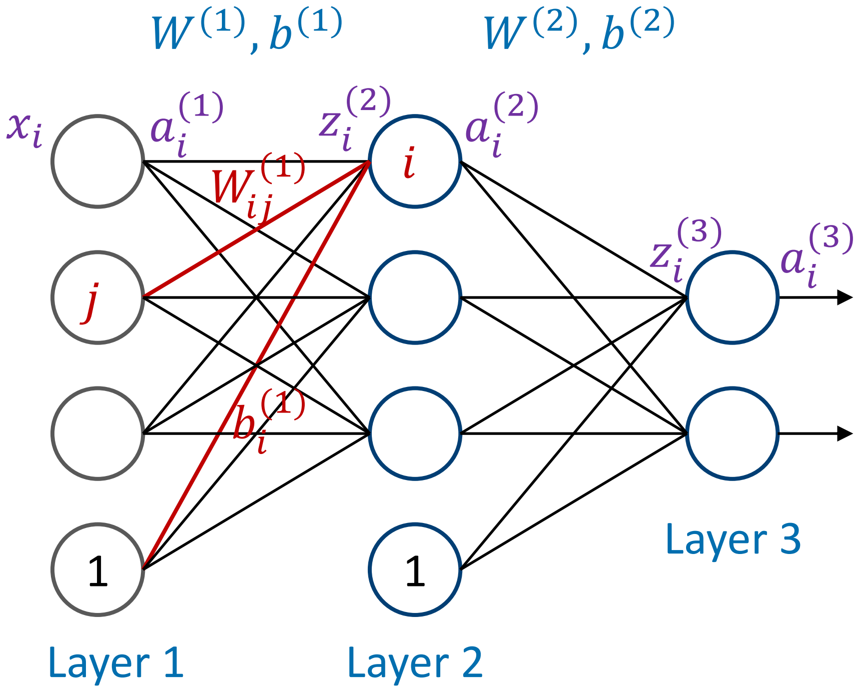
$$z_i^{(2)} = W_{i1}^{(1)} a_1^{(1)} + W_{i2}^{(1)} a_2^{(1)} + W_{i3}^{(1)} a_3^{(1)} + b_i^{(1)}$$

$$a_i^{(2)} = f(z_i^{(2)})$$

- Using matrix notation, we have

$$z^{(2)} = W^{(1)} a^{(1)} + b^{(1)}$$

$$a^{(2)} = f(z^{(2)})$$ Apply activation function element-wise.



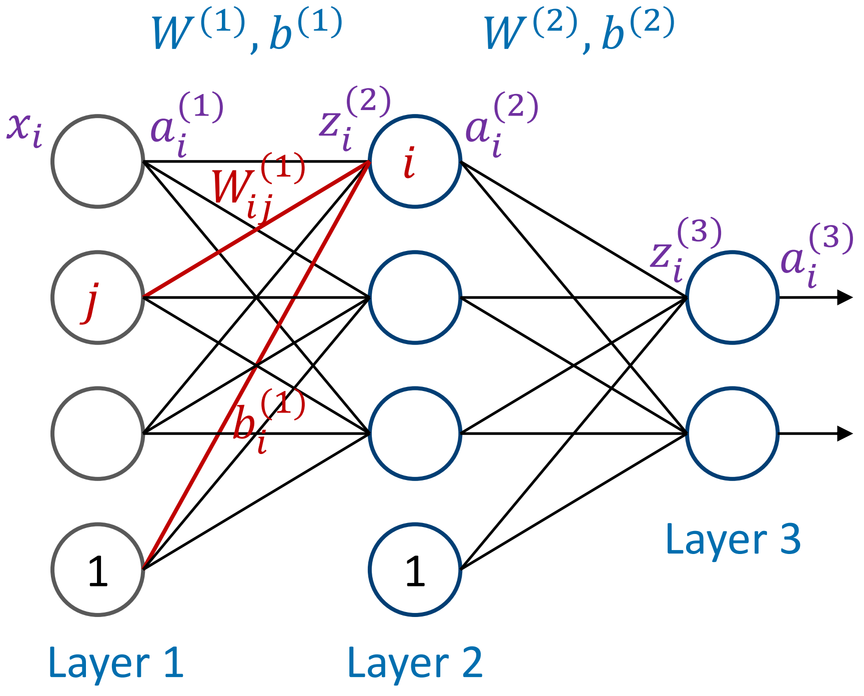
We do the same for following layers

- At Layer 3, we calculate the input to each neuron, then apply the activation function.

$$z_i^{(3)} = W_{i1}^{(2)} a_1^{(2)} + W_{i2}^{(2)} a_2^{(2)} + W_{i3}^{(2)} a_3^{(2)} + b_i^{(2)}$$
$$a_i^{(3)} = f(z_i^{(3)})$$

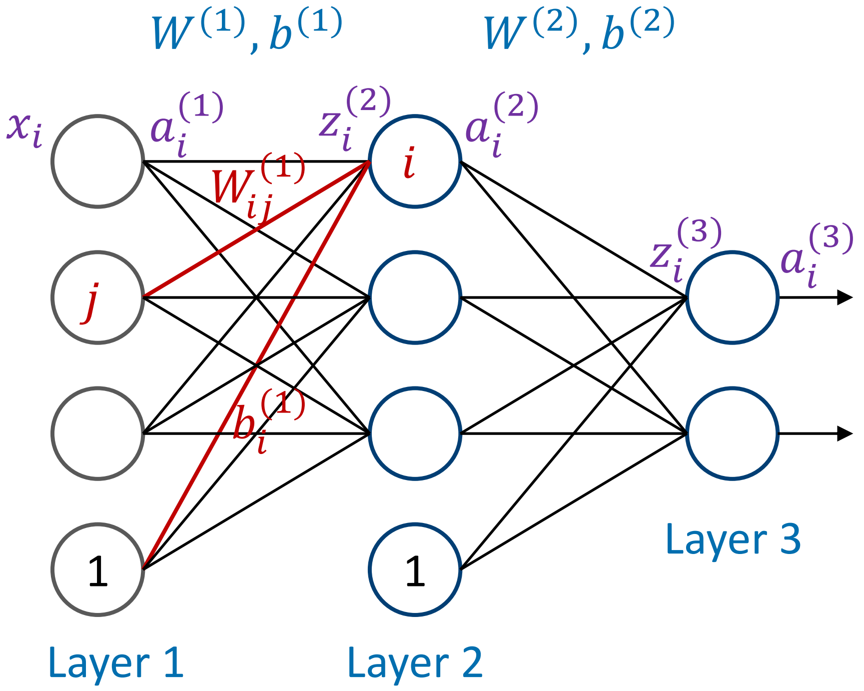
- Using matrix notation, we have

$$z^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$$
$$a^{(3)} = f(z^{(3)})$$



Notations

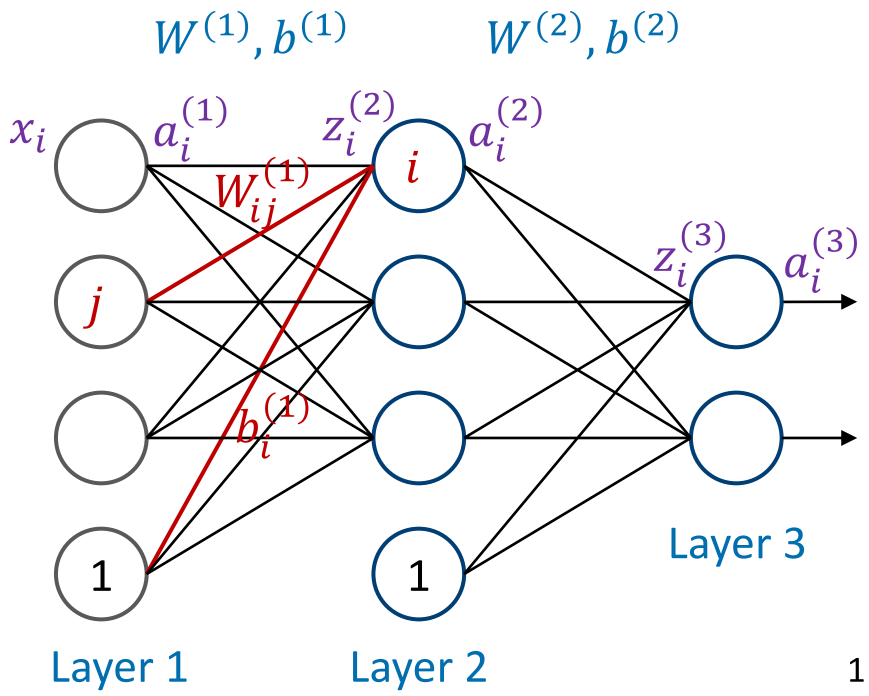
- $W^{(l)}$: weight matrix at layer l
- $W_{ij}^{(l)}$: weight for the connection between neuron j at layer l and neuron i at layer $l+1$
- $b^{(l)}$: bias vector at layer l
- $b_i^{(l)}$: bias at layer l connecting to neuron i at layer $l+1$
- $z_i^{(l)}$: total input to neuron i at layer l
- $a_i^{(l)}$: activation of neuron i at layer l
- $a_i^{(1)}$: input i at the first layer, $a_i^{(1)} = x_i$



In this example,
$W^{(1)}$ is a 3x3 matrix, $b^{(1)}$ is a 3x1 vector,
$W^{(2)}$ is a 2x3 matrix, $b^{(2)}$ is a 2x1 vector.

Forward propagation

- Given a fixed setting of parameters W and b, the neural network computes the output given input x.
- At Layer 1, $a^{(1)} = x$
- At Layer 2, $z^{(2)} = W^{(1)}a^{(1)} + b^{(1)}$
  $$a^{(2)} = f(z^{(2)})$$
- At Layer 3, $z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$
  $$a^{(3)} = f(z^{(3)})$$
- In general, for Layer l, we have
  $$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
  $$a^{(l+1)} = f(z^{(l+1)})$$

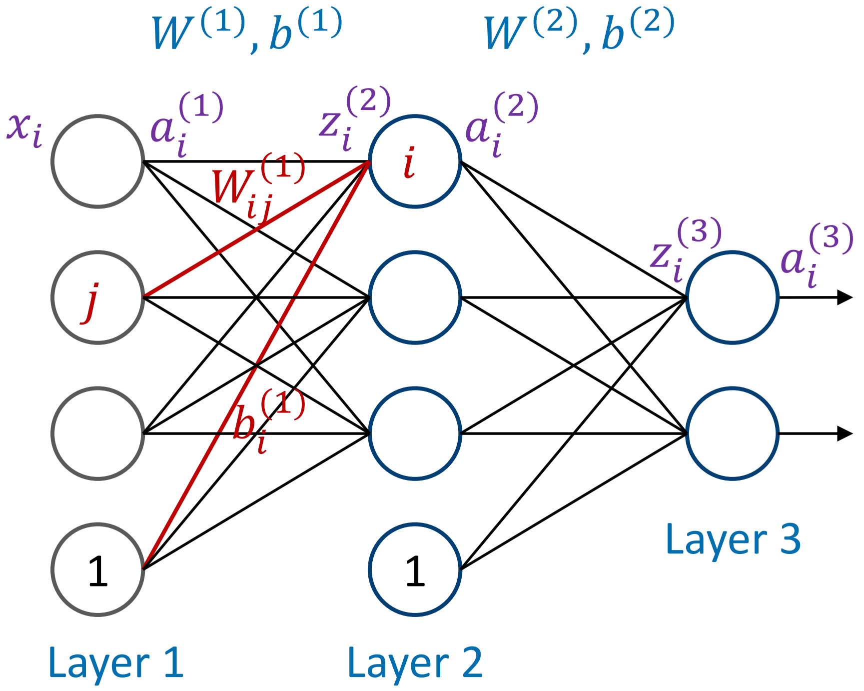


Forward propagation

- The calculation for each layer is the same.
- For Layer $l + 1$, we have

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)})$$

- We do this layer by layer.
- This is called forward propagation.



Estimate parameters

- The first question is solved.
- The second question is to find parameters W and b that minimise the loss function $J(W, b)$.

$$J(W, b) = \frac{1}{M}\sum_{m=1}^{M} \frac{1}{2} ||a_m - y_m||^2$$

Gradient descent

- Gradient descent is a technique for optimising a function with respect to some parameters.

$$W = W - \alpha \frac{\partial J}{\partial W}$$

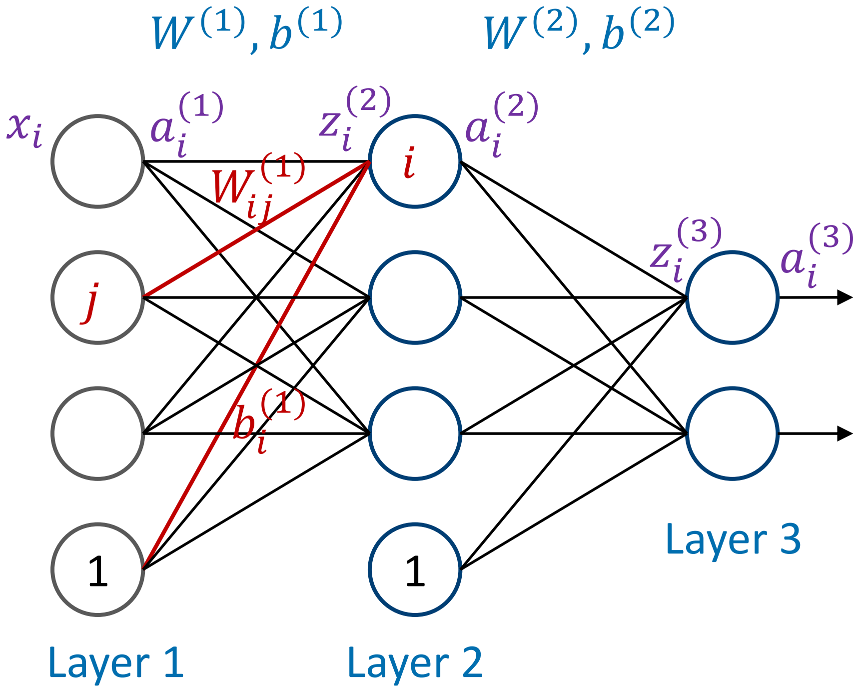
$$b = b - \alpha \frac{\partial J}{\partial b}$$

  where α is the learning rate or step size.

- As long as we know the gradient $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial b}$, the second question is also solvable.

Gradient descent

- The gradient $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial b}$ can be calculated using the backpropagation algorithm.
- It is based on the chain rule in differentiation.



Chain rule

$$x \longrightarrow y \longrightarrow z$$

- Suppose we have a composition of two functions $z = g(f(x))$, where

$$y = f(x)$$
$$z = g(y)$$

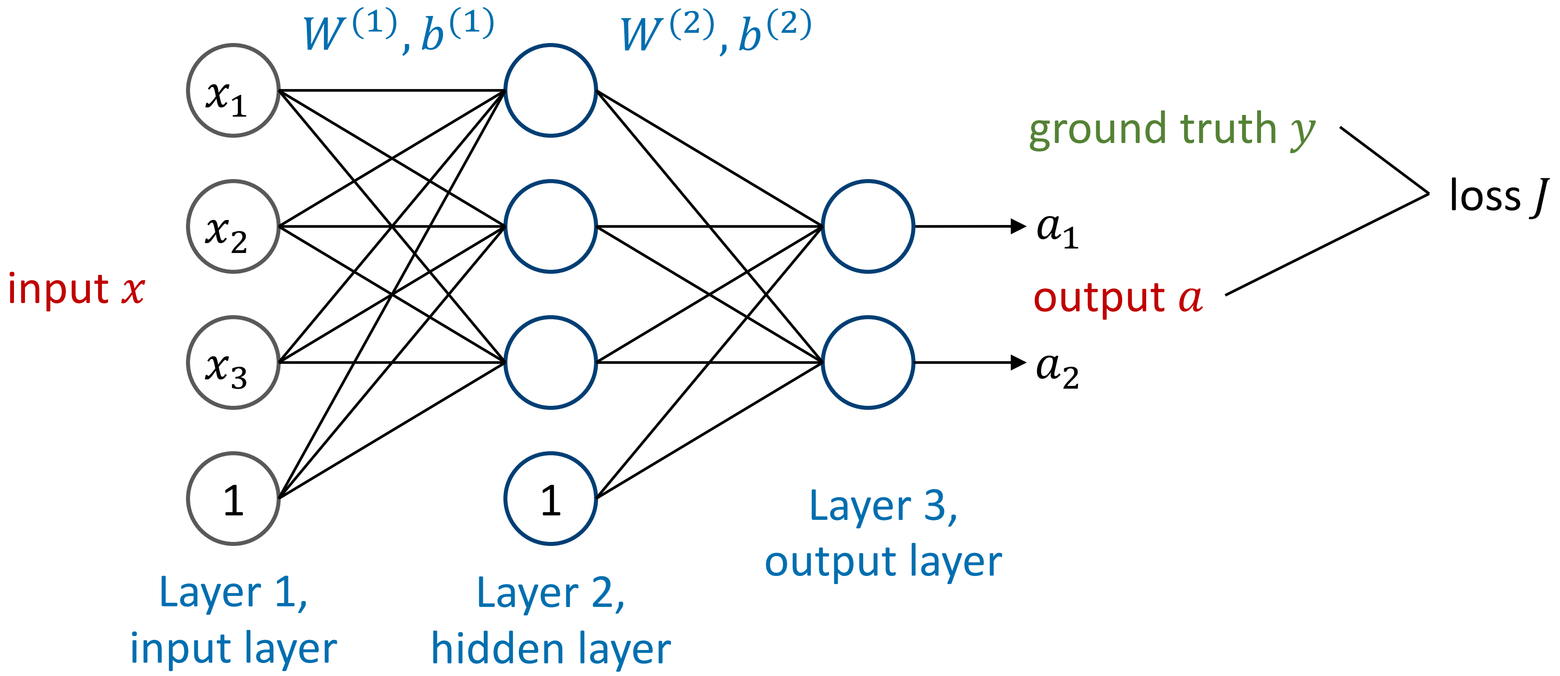
- The chain rule expresses the derivative of the composition in terms of the derivative for each single function.

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

Backpropagation

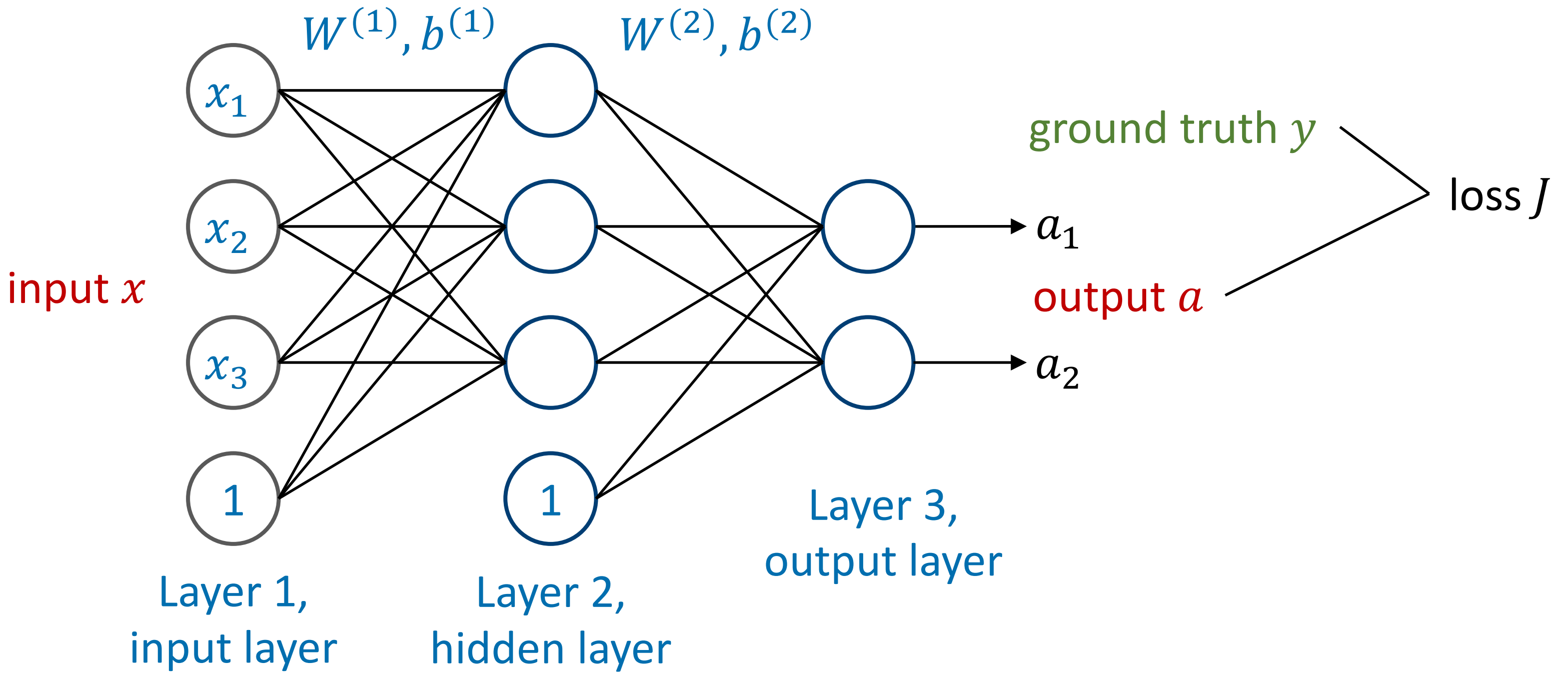
- The relation between J and a is simple,

$$J(W,b) = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{2}\|a_m - y_m\|^2$$



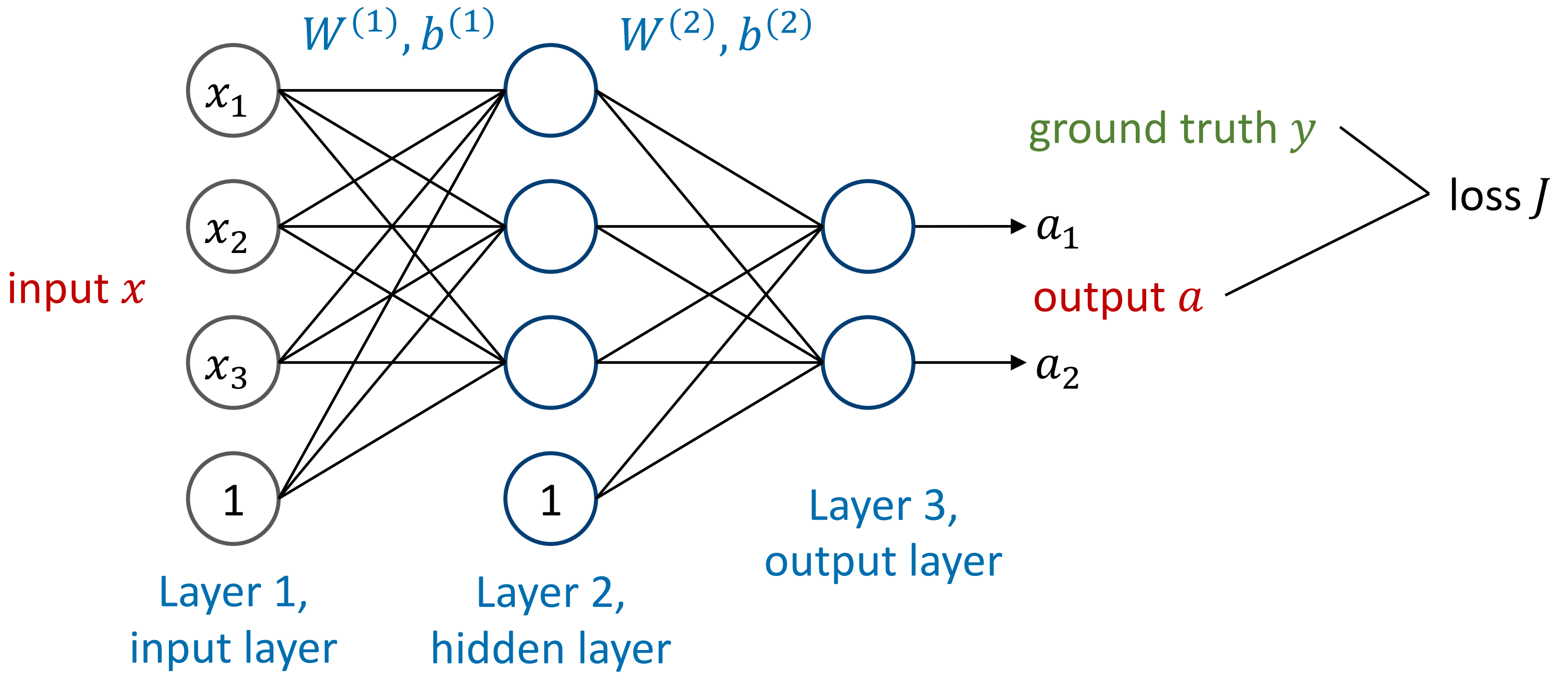
Backpropagation

- We can easily derive $\frac{\partial J}{\partial a}$.



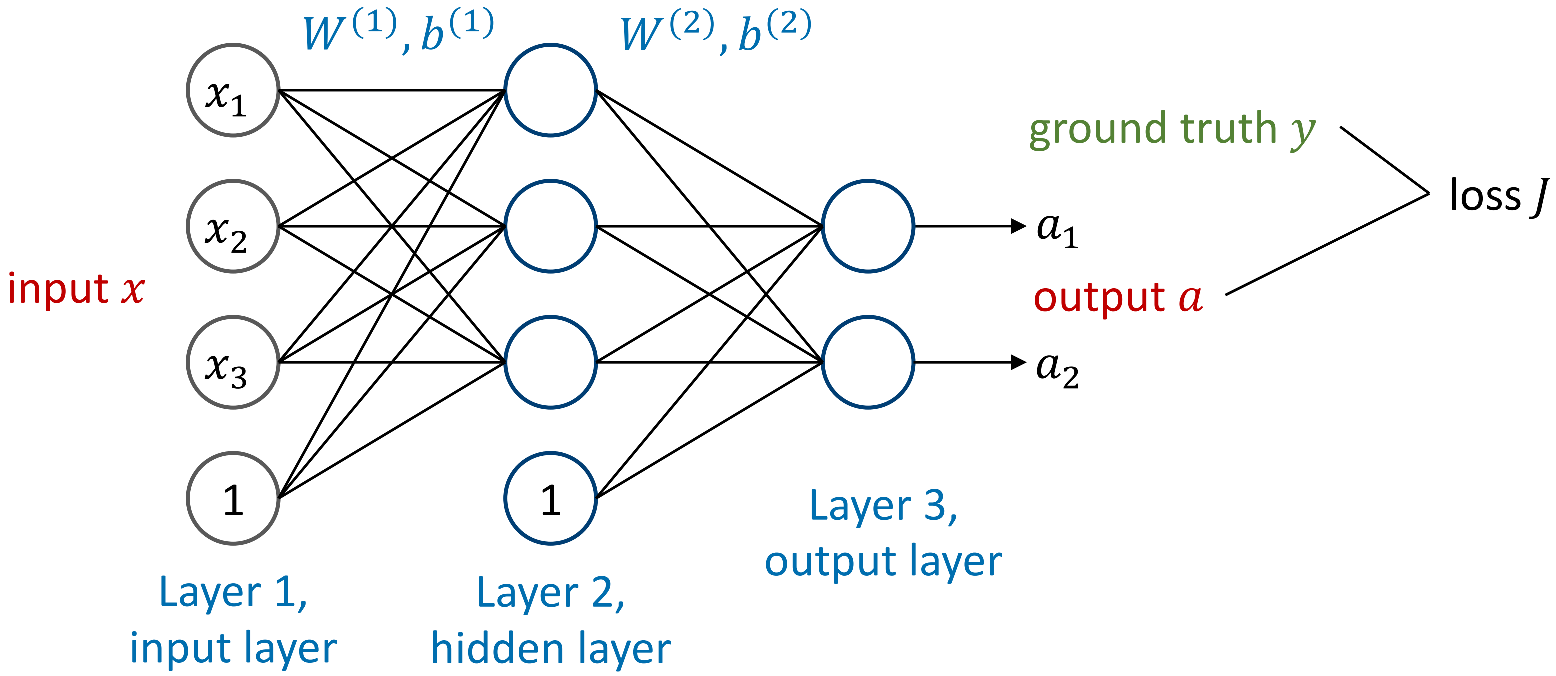
Backpropagation

- The relation between $W^{(2)}$ and a is also simple.



Backpropagation

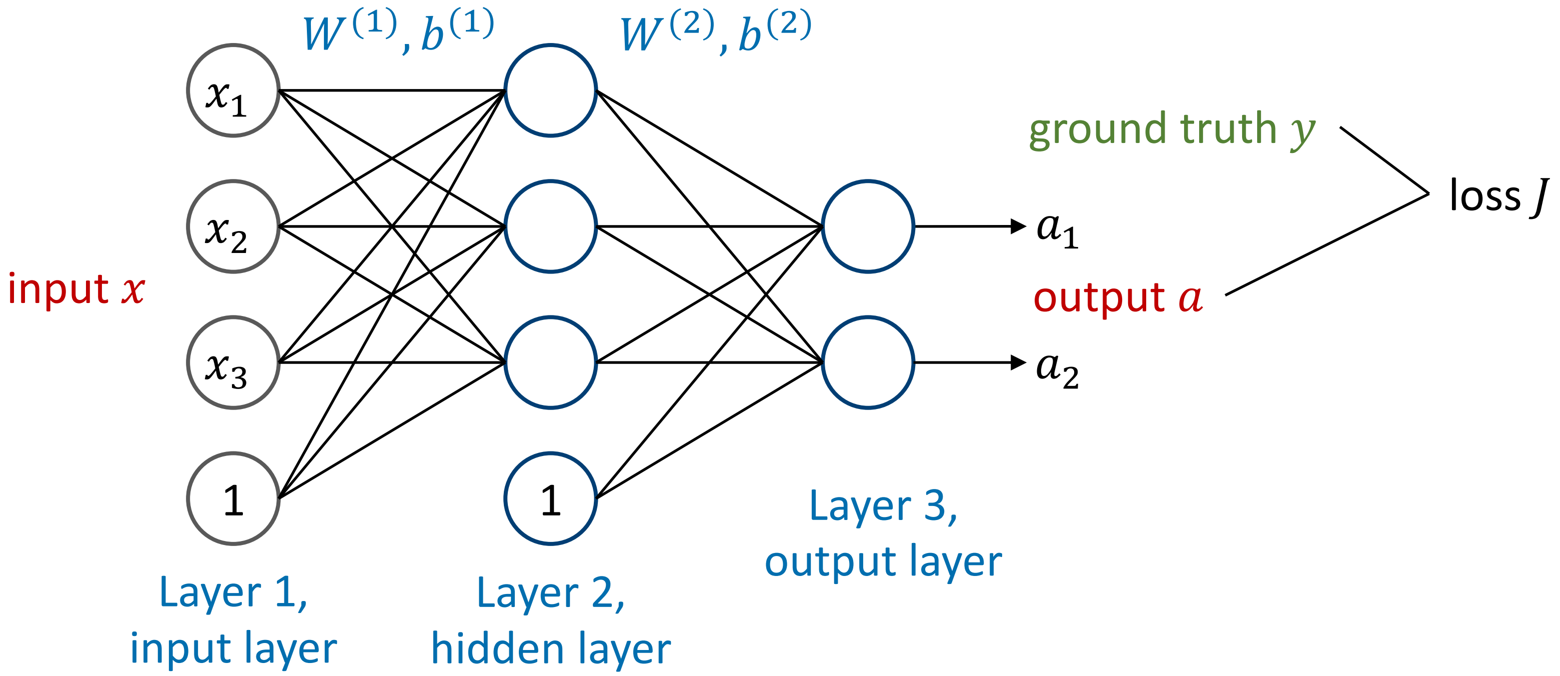
- We can easily derive $\frac{\partial a}{\partial W^{(2)}}$.



Backpropagation

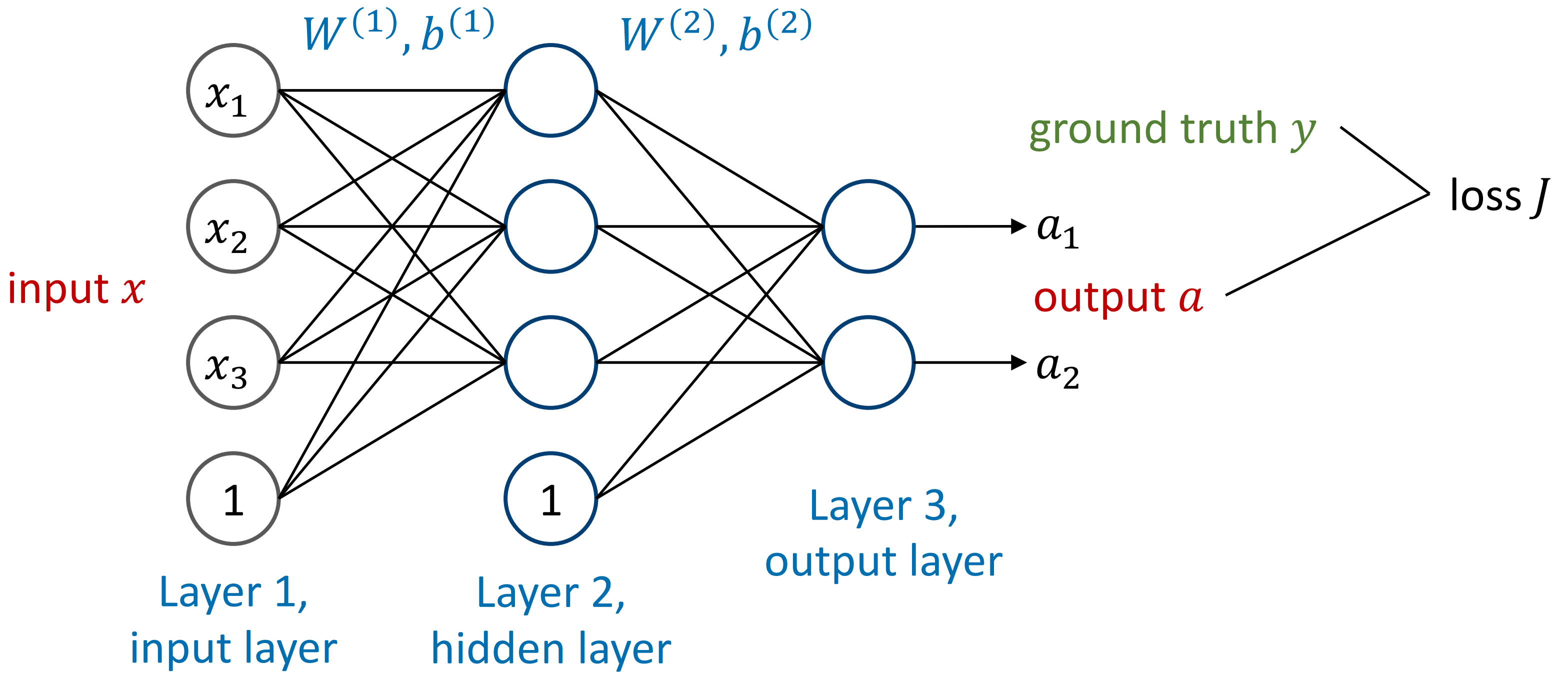
- Now using the chain rule, we can derive $\frac{\partial J}{\partial W^{(2)}}$ as

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



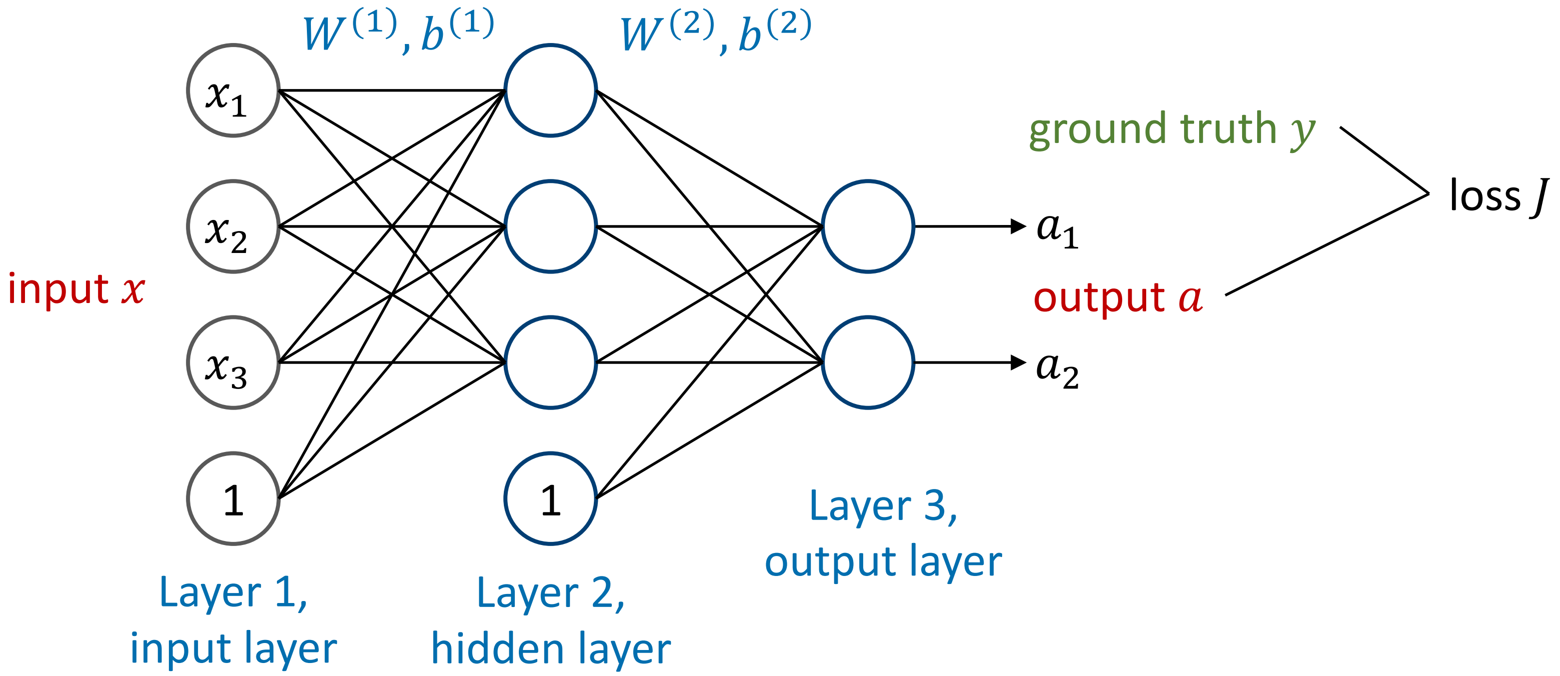
Backpropagation

- In the same way, we can propagate the gradient back layer by layer.
- As the end, we can calculate $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial b}$ for all layers.

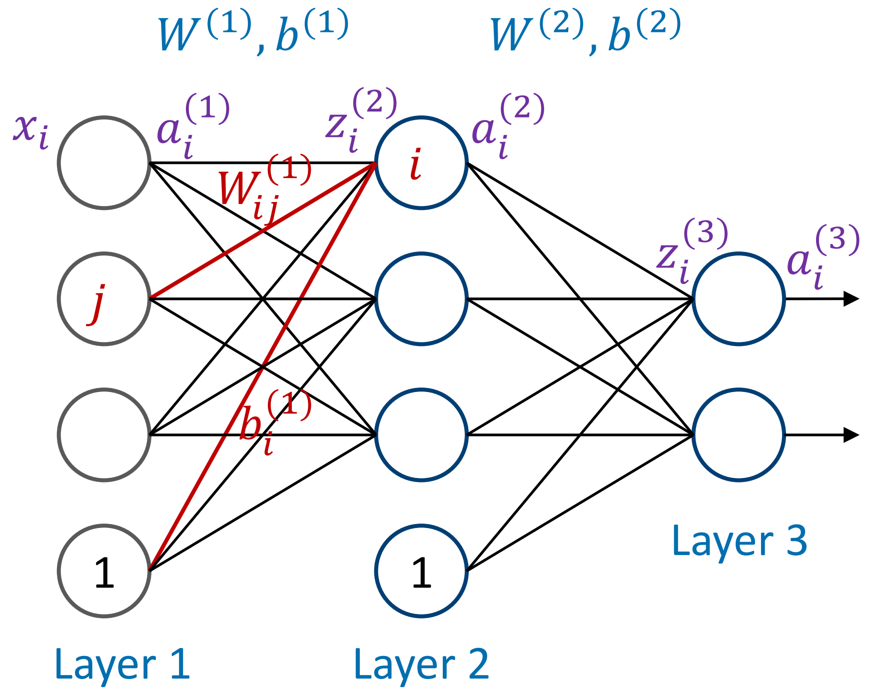


Backpropagation

- The detailed mathematical derivation is slightly longer.
- But the idea is simple, which is to use the chain rule.



Backpropagation



- Let us look at the loss function

$$J(W,b) = \frac{1}{M}\sum_{m=1}^{M} \frac{1}{2}\left\|a_m^{(3)} - y_m\right\|^2$$

- For each sample, the loss is

$$J(W,b;x_m,y_m) = \frac{1}{2}\left\|a_m^{(3)} - y_m\right\|^2$$

- Let us calculate the derivative for just one sample and ignore the sample index m for simplicity,

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

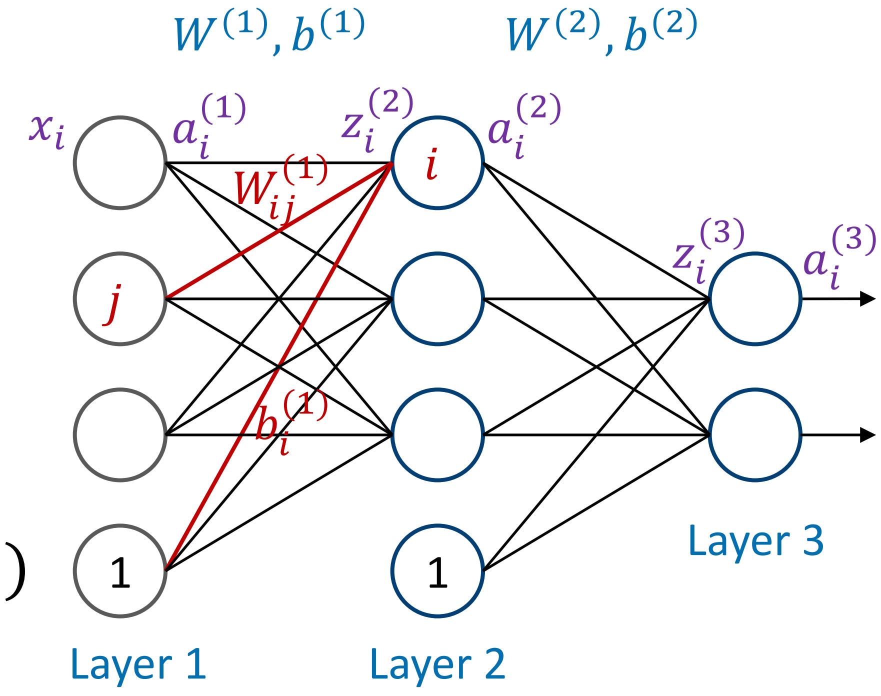
$$\frac{\partial J}{\partial z^{(3)}} = \frac{\partial J}{\partial a^{(3)}}\frac{\partial a^{(3)}}{\partial z^{(3)}} = \left(a^{(3)} - y\right) \circ f'(z^{(3)})$$

Element-wise multiplication

A vector, length equal to the number of neurons on this layer

Derivative of activation function

Backpropagation



- For Layer 3,

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

$$\frac{\partial J}{\partial z^{(3)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} = \left(a^{(3)} - y\right) \circ f'\left(z^{(3)}\right)$$

- For the previous layer, Layer 2,

$$\frac{\partial J}{\partial W_{ij}^{(2)}} = \frac{\partial J}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial W_{ij}^{(2)}} = \frac{\partial J}{\partial z_i^{(3)}} a_j^{(2)}$$

- We can use the matrix notation,

$$\underbrace{\frac{\partial J}{\partial W^{(2)}}}_{\text{I x J matrix}} = \underbrace{\frac{\partial J}{\partial z^{(3)}}}_{\text{I x 1 vector}} \underbrace{\left(a^{(2)}\right)^T}_{\text{1 x J vector}}$$

Backpropagation

- For Layer 3,

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

$$\frac{\partial J}{\partial z^{(3)}} = \frac{\partial J}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(3)}} = \left(a^{(3)} - y\right) \circ f'(z^{(3)})$$

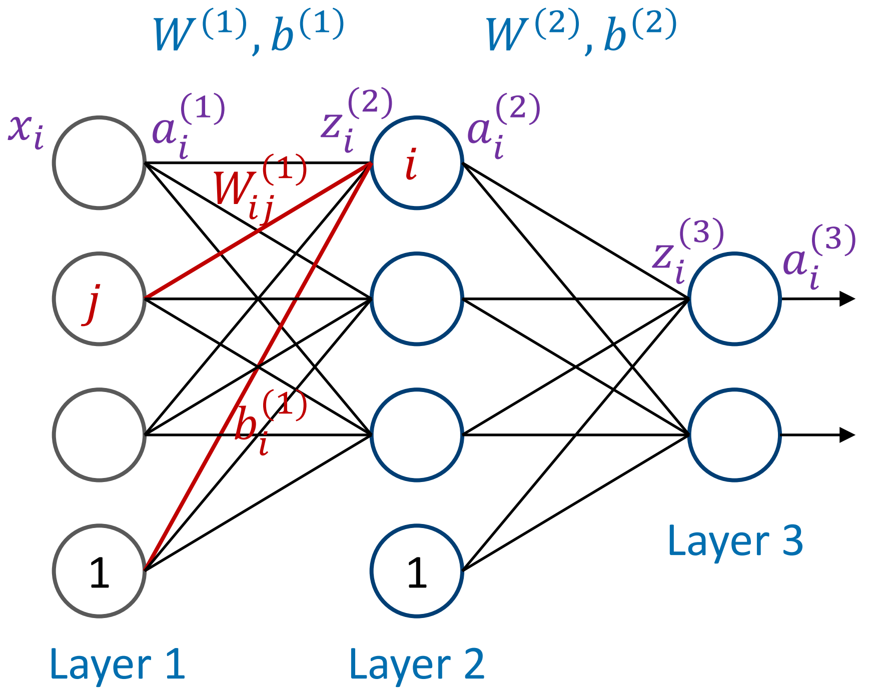
- For the previous layer, Layer 2,

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

$$\frac{\partial J}{\partial b^{(2)}} = \frac{\partial J}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial b^{(2)}} = \frac{\partial J}{\partial z^{(3)}}$$

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

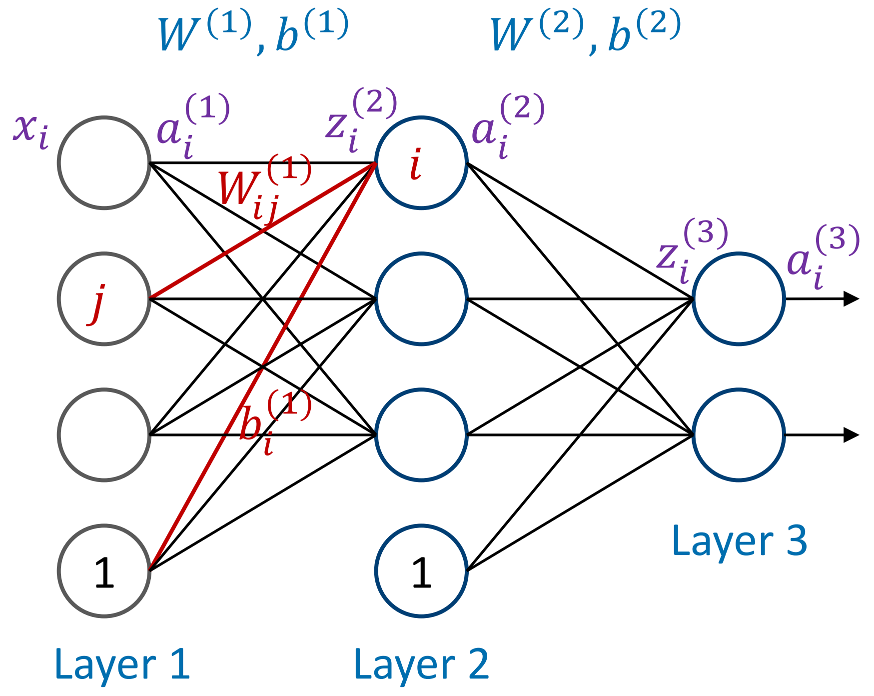
$$\frac{\partial J}{\partial z^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} = ((W^{(2)})^T \frac{\partial J}{\partial z^{(3)}}) \circ f'(z^{(2)})$$



Key observation:

To calculate the derivatives at Layer 2, we only need the information of $\frac{\partial J}{\partial z^{(3)}}$.

Backpropagation



- Similarly, for Layer 1,

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

$$\frac{\partial J}{\partial b^{(1)}} = \frac{\partial J}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial b^{(1)}} = \frac{\partial J}{\partial z^{(2)}}$$

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

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

- To calculate the derivatives at Layer l, we only need the derivatives at Layer $l+1$, in particular $\frac{\partial J}{\partial z^{(l+1)}}$.

Backpropagation algorithm

- Perform forward propagation, calculate the neuron inputs $z^{(l)}$, activations $a^{(l)}$.
- At the output layer, calculate the derivative $\frac{\partial J}{\partial z^{(L)}}$,

$$\frac{\partial J}{\partial z^{(L)}} = (a^{(L)} - y) \circ f'(z^{(L)})$$

- For layer $l = L-1, L-2, \cdots, 1$, calculate the propagated derivatives,

$$\frac{\partial J}{\partial W^{(l)}} = \frac{\partial J}{\partial z^{(l+1)}} (a^{(l)})^T$$

$$\frac{\partial J}{\partial b^{(l)}} = \frac{\partial J}{\partial z^{(l+1)}}$$

$$\frac{\partial J}{\partial z^{(l)}} = ((W^{(l)})^T \frac{\partial J}{\partial z^{(l+1)}}) \circ f'(z^{(l)})$$

Gradient descent

- Now with the gradient $\frac{\partial J}{\partial W}$ and $\frac{\partial J}{\partial b}$, we can perform gradient descent

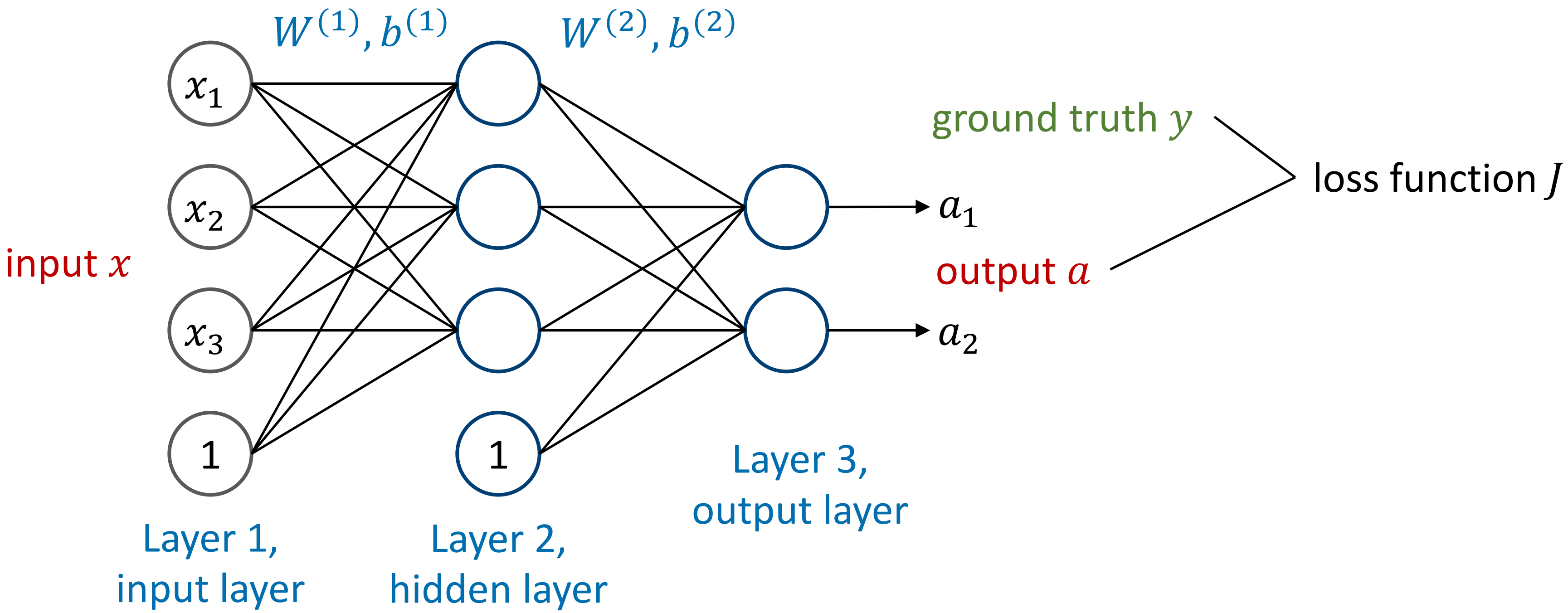
$$W = W - \alpha \frac{\partial J}{\partial W}$$

$$b = b - \alpha \frac{\partial J}{\partial b}$$

- So the network parameters W and b can be optimised after some iterations. The second question is also solved.

Neural networks

- Now we know how a neural network works.
  - Given input x, we know how to calculate output a.
  - We also know to how optimise parameters W and b to minimise the loss function J.



Multi-layer perceptron (MLP), which is a fully connected multi-layer network.

References

- Ch. 6, Deep Feedforward Networks. Ian Goodfellow et al. Deep Learning (https://www.deeplearningbook.org/).