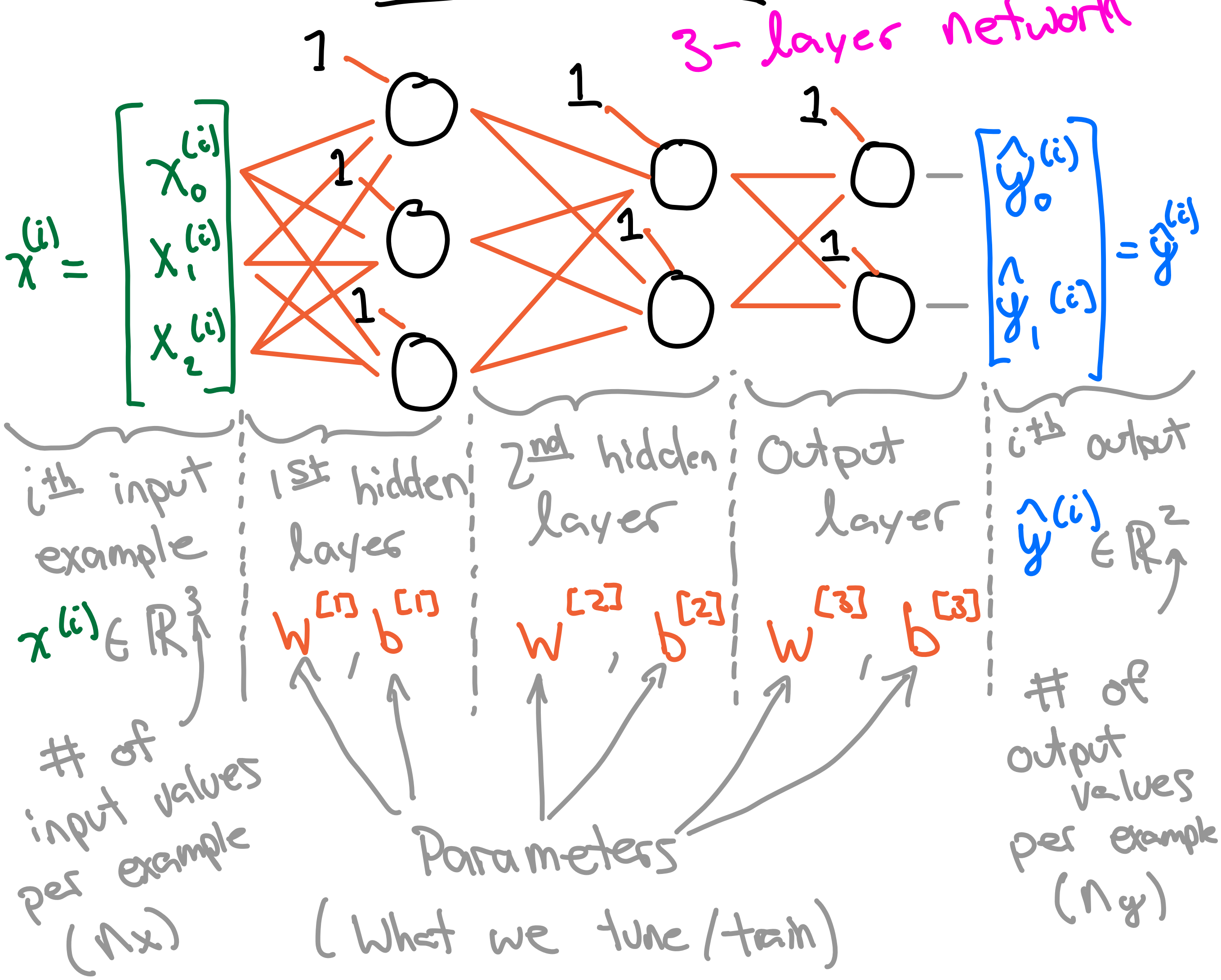
Notation



The input size (n_x) and output size (n_y) depend on the problem.

The number of layers (L) and the number of neurons per layer (n_l) are "Hyperparameters" chosen by you.

Summary

"(i)" parentheses are used to denote a specific input training example/sample

"[l]" brackets are used to denote a specific layer in the network

$x^{(i)}$: a single input example with a shape $\in \mathbb{R}^{n_x}$, which is a $(n_x, 1)$ column vector

$\hat{y}^{(i)}$: a single output example prediction with a shape $\in \mathbb{R}^{n_y}$, which is a $(n_y, 1)$ column vector

$W^{[l]}$: each hidden and output layer includes a matrix of parameter values (sometimes called the "weight" matrix). This matrix $\in \mathbb{R}^{n_l \times n_{l-1}}$ the shape is (n_l, n_{l-1})

n_l: # of neurons in l^{th} layer

n_{l-1}: # of neurons in $l-1$ layer

$b^{[l]}$: each hidden and output layer includes a vector of parameter values (sometimes called the "bias" vector). This vector $\in \mathbb{R}^{n_l}$, the shape is $(n_l, 1)$.

Forward Compute

We use $x^{(i)}$ and all $W^{[\ell]}$, $b^{[\ell]}$ values to compute $\hat{y}^{(i)}$.

Specifically, we compute the output of every neuron as follows:

$$z^{[\ell](i)} = W^{[\ell]} a^{[\ell-1](i)} + b^{[\ell]}$$

Linear output of all neurons in layer ℓ given example i. The shape $\in \mathbb{R}^{n_\ell}$.

$$a^{[\ell](i)} = g^{[\ell]}\left(z^{[\ell](i)}\right)$$

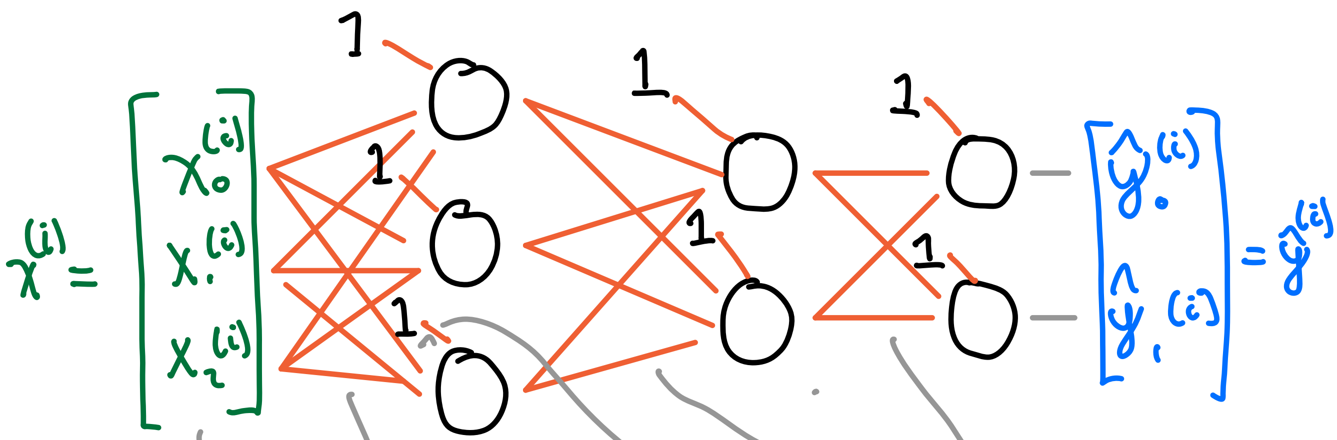
Activation function

"Activation" (output) of all neurons in layer ℓ for the example i. The shape $\in \mathbb{R}^{n_\ell}$.

$$a^{[0](i)} = x^{(i)}$$ "Output" of input neurons

$$\hat{y}^{(i)} = a^{[L](i)}$$ Output of output neurons

We can now compute the output for the example network.



$$x^{(i)} = a^{[0](i)}$$

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = g\left(z^{[1](i)}\right)$$

Feeding forward

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

$$a^{[2](i)} = g\left(z^{[2](i)}\right)$$

Feeding forward

$$z^{[3](i)} = W^{[3]} a^{[2](i)} + b^{[3]}$$

$$\hat{y}^{(i)} = g\left(z^{[3](i)}\right)$$

$$\hat{y}^{(i)} = a^{[3](i)}$$

Loss

How do we compute performance?

We want some function that we can optimize. We need to know how to update the parameters $w^{[l]}, b^{[l]}$ so that we can drive our prediction to the true value

hat → $\hat{y}^{(i)} \approx y^{(i)}$ ← no hat

$\hat{y}^{(i)}$: "prediction" $y^{(i)}$: "target" / label

Lets use the "mean-squared-error", aka MSE.

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = \underbrace{\frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2}_{\text{MSE}}$$

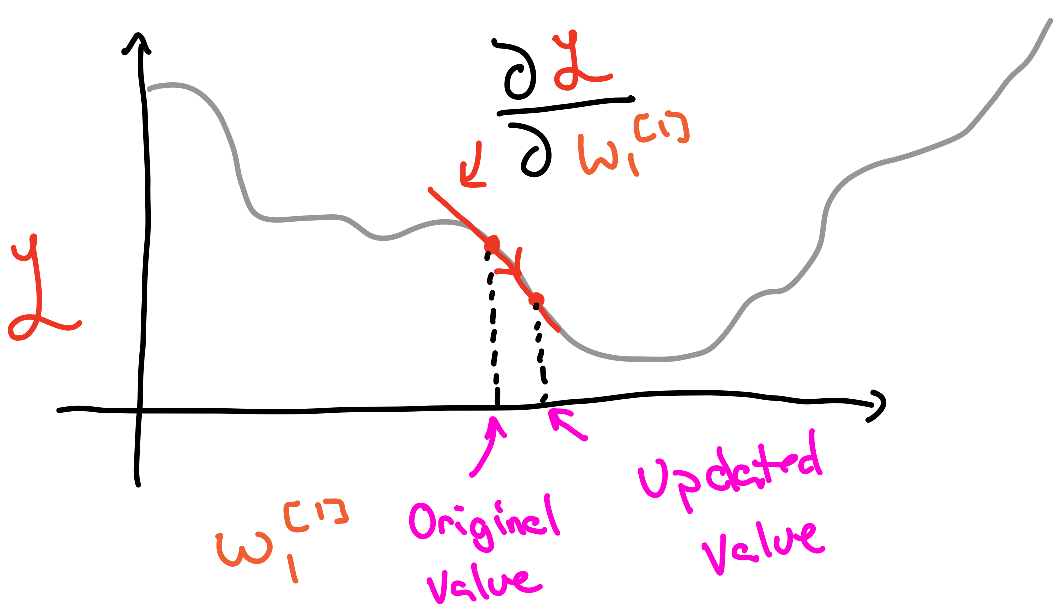
$\mathcal{L}$: "Loss"; $\hat{y}^{(i)}$: prediction; $y^{(i)}$: true target

Backpropagation

We want to find values for $W^{[l]}$, $b^{[l]}$ to minimize $\mathcal{L}$. Effectively, we want $\hat{y}^{(i)} \approx y^{(i)}$.

We can do this by starting with random values for $W^{[l]}$, $b^{[l]}$ and then updating them in such a way that we continually reduce $\mathcal{L}$.

In effect, we want to follow the slope of $\mathcal{L}$ to smaller and smaller values.



So, we need to compute:

$\frac{\partial \mathcal{L}}{\partial W^{[1]}}$, $\frac{\partial \mathcal{L}}{\partial b^{[1]}}$, Parameters for layer 1

$\frac{\partial \mathcal{L}}{\partial W^{[2]}}$, $\frac{\partial \mathcal{L}}{\partial b^{[2]}}$, Parameters for layer 2

$\frac{\partial \mathcal{L}}{\partial W^{[3]}}$, $\frac{\partial \mathcal{L}}{\partial b^{[3]}}$, Parameters for layer 3

Derive equations for these six quantities using the example 3-layer network, the half-MSE loss function, and assuming that the activation function $g(\cdot)$ is the sigmoid function.

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