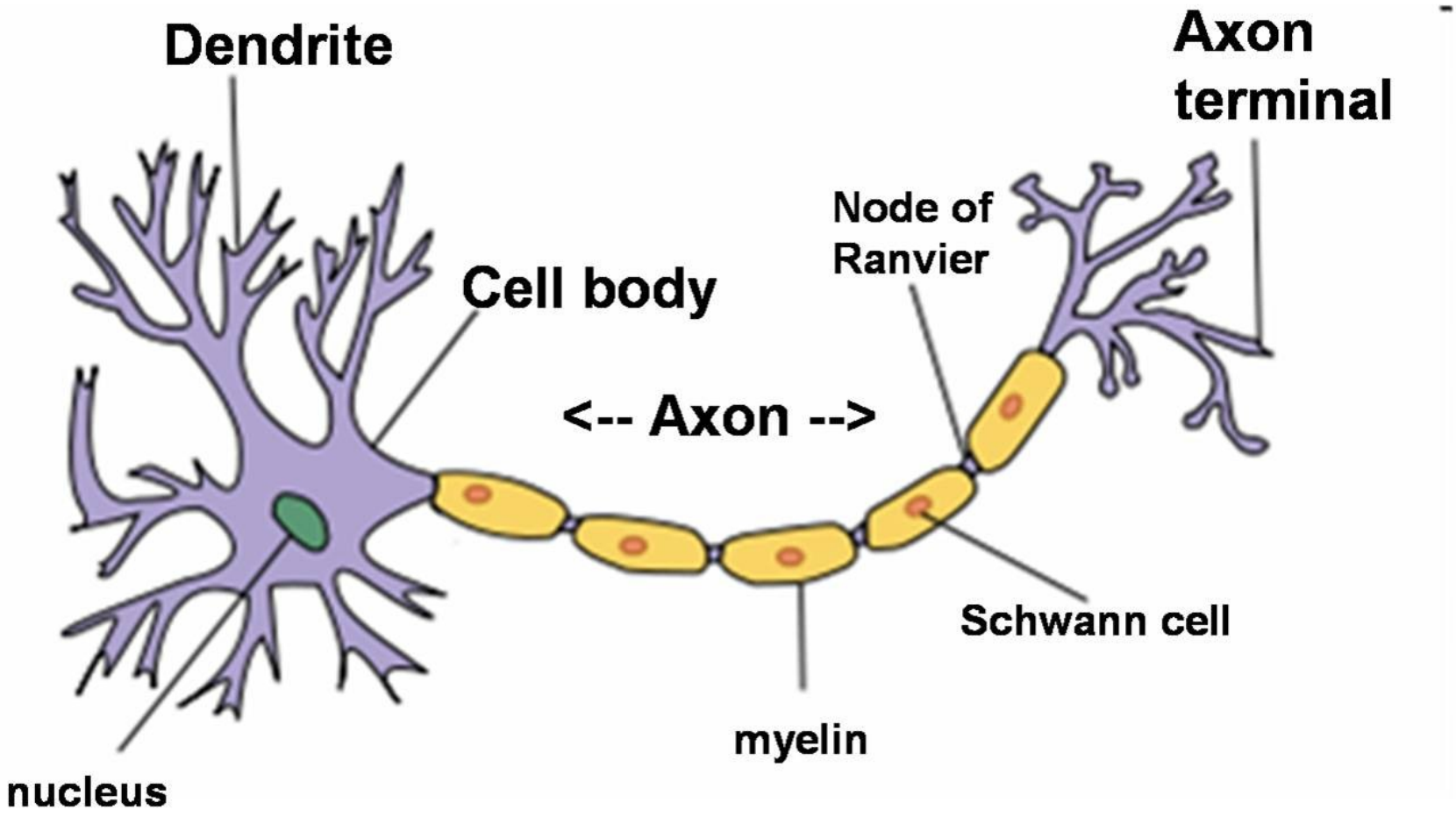
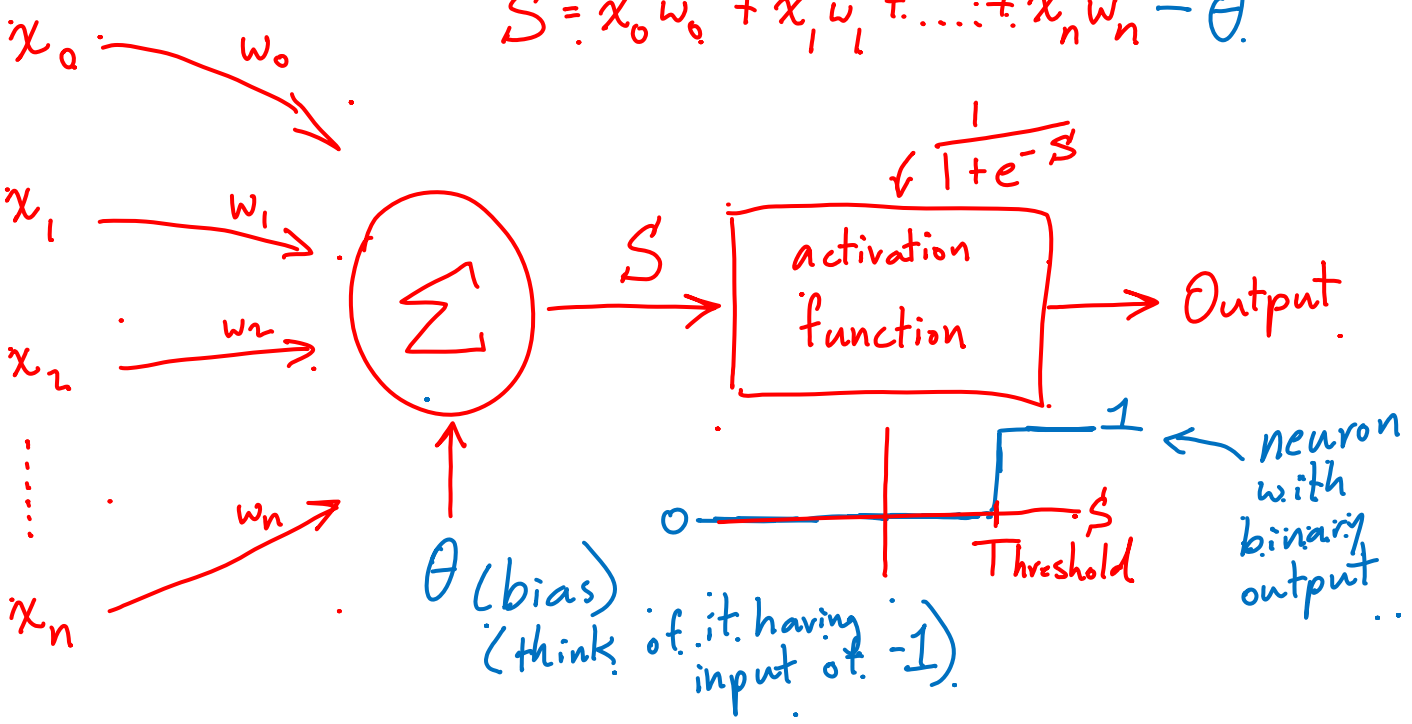
- Lab 4

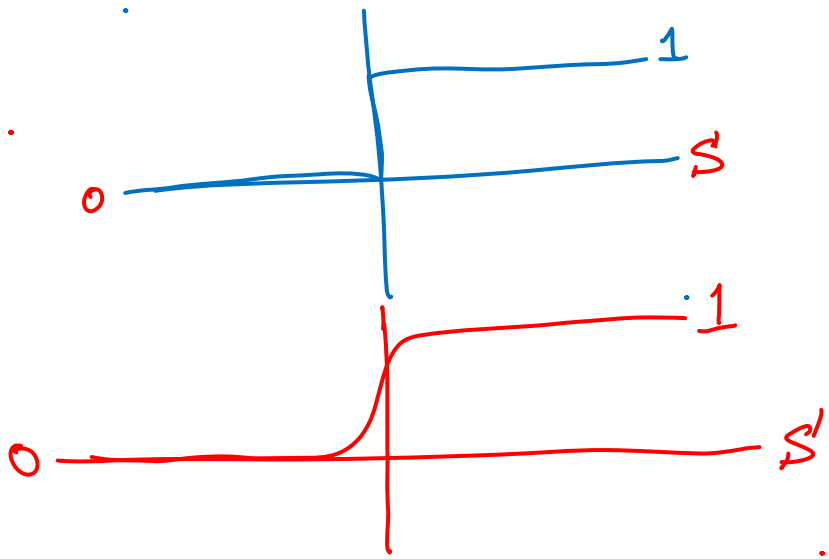

Dendrite
Axon terminal
Node of Ranvier
Cell body
<-- Axon -->
Schwann cell
myelin
nucleus

Neuron Model

$$S = x_0 w_0 + x_1 w_1 + \ldots + x_n w_n - \theta$$



Activation Function



$$s(t) = \frac{1}{1 + e^{-t}}$$

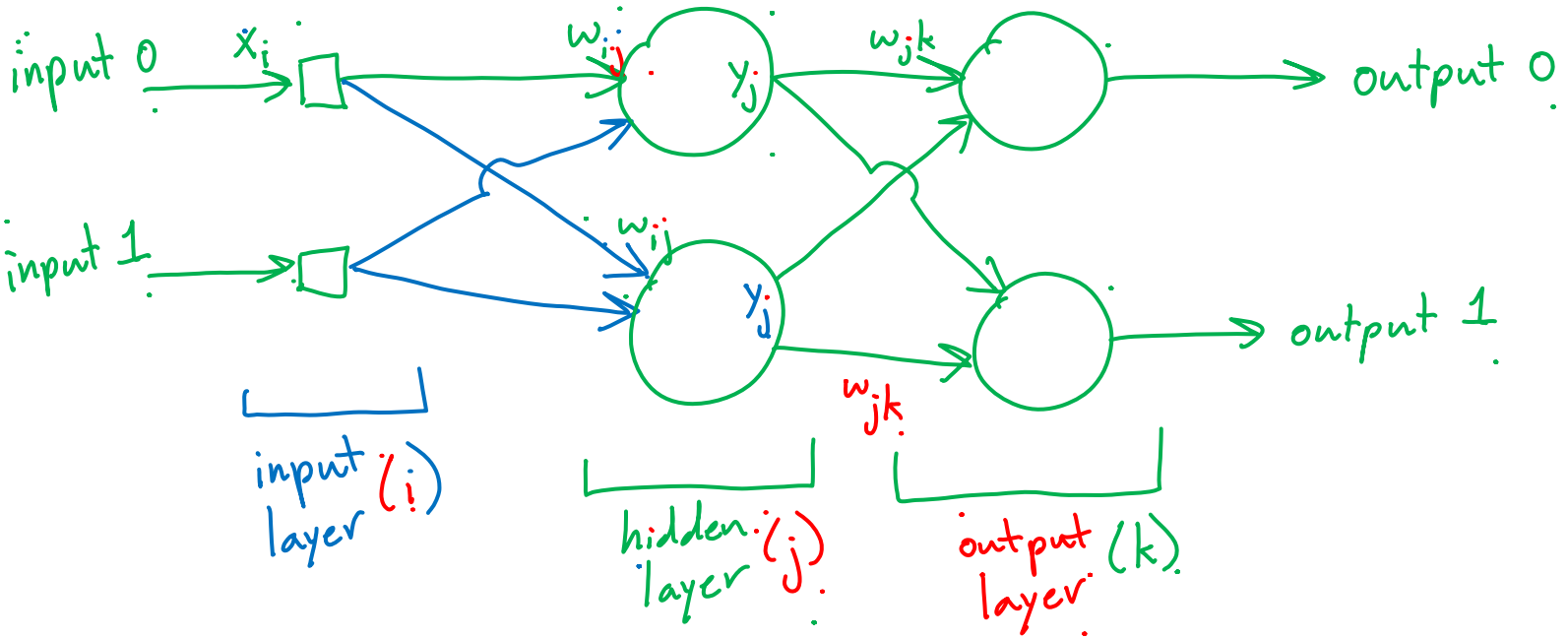
sigmoid

out 1 1 0

16 16 16

16 A 16 16 B

Artificial Neural Network



- Training a neural network (Back Propagation)

$e_k(p)$

- Compute the error at the output neurons first and then work on hidden neurons

the actual neuron output $= y_k(p)$ (output of neuron k at training iteration p)

the desired neuron output $= y_{d,k}(p)$

the error at iteration p for neuron k $= y_{d,k}(p) - y_k(p)$

$e_k(p)$

1. Compute $e_k(p)$ for each output neuron.

2. Adjust the weights in the output neurons.

α = learning rate

$w_{jk}(p)$ ← weight of output neuron

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

$$\Delta w_{jk}(p) = \alpha \cdot y_j(p) \cdot \delta_k(p)$$

$$\delta_k(p) = y_k(p) \cdot (1 - y_k(p)) \cdot e_k(p)$$

$\delta_k(p)$: error gradient

$y_k(p)$: output of neuron

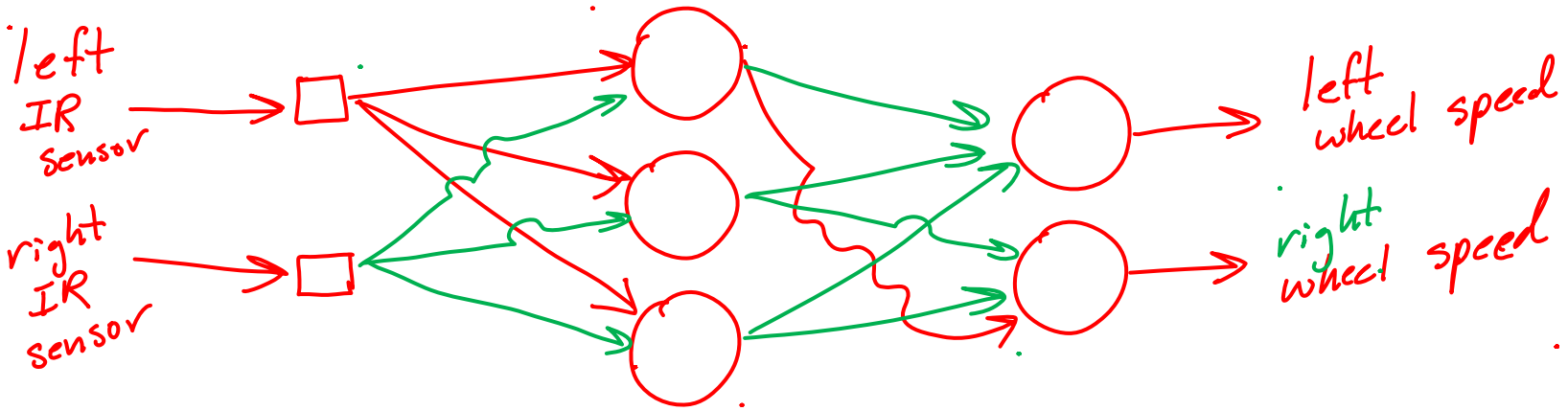
3. Adjust the weights for hidden neurons.

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

$$\Delta w_{ij}(p) = \alpha \cdot x_i(p) \cdot \delta_j(p)$$

$x_i(p)$: input value

$$\delta_j(p) = y_j(p) \cdot (1 - y_j(p)) \cdot \sum_{\text{over all output neurons}} (\delta_k(p) \cdot w_{jk}(p))$$



Part 2

1. Turn on robot
   - run using proportional control
2. Push button
   - print "Data"
   - store sensor readings while moving robot side to side
3. Push button
   - Training mode
   - Have the # of training iterations as user input

Part 3

- Train network while line following using Proportional