# MIDS W207
# Applied Machine Learning

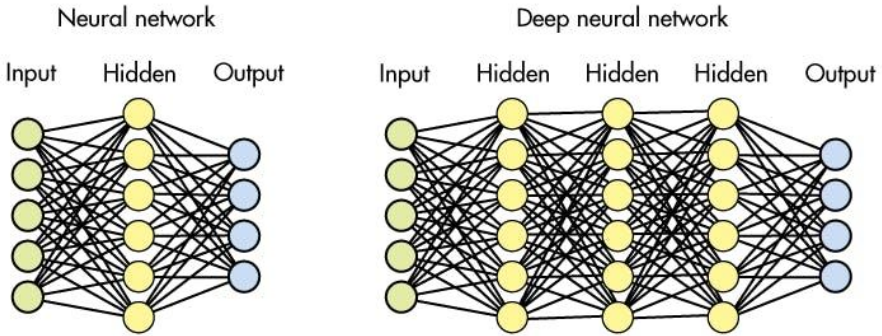
Week 7
Live Session Slides

# Neural Networks

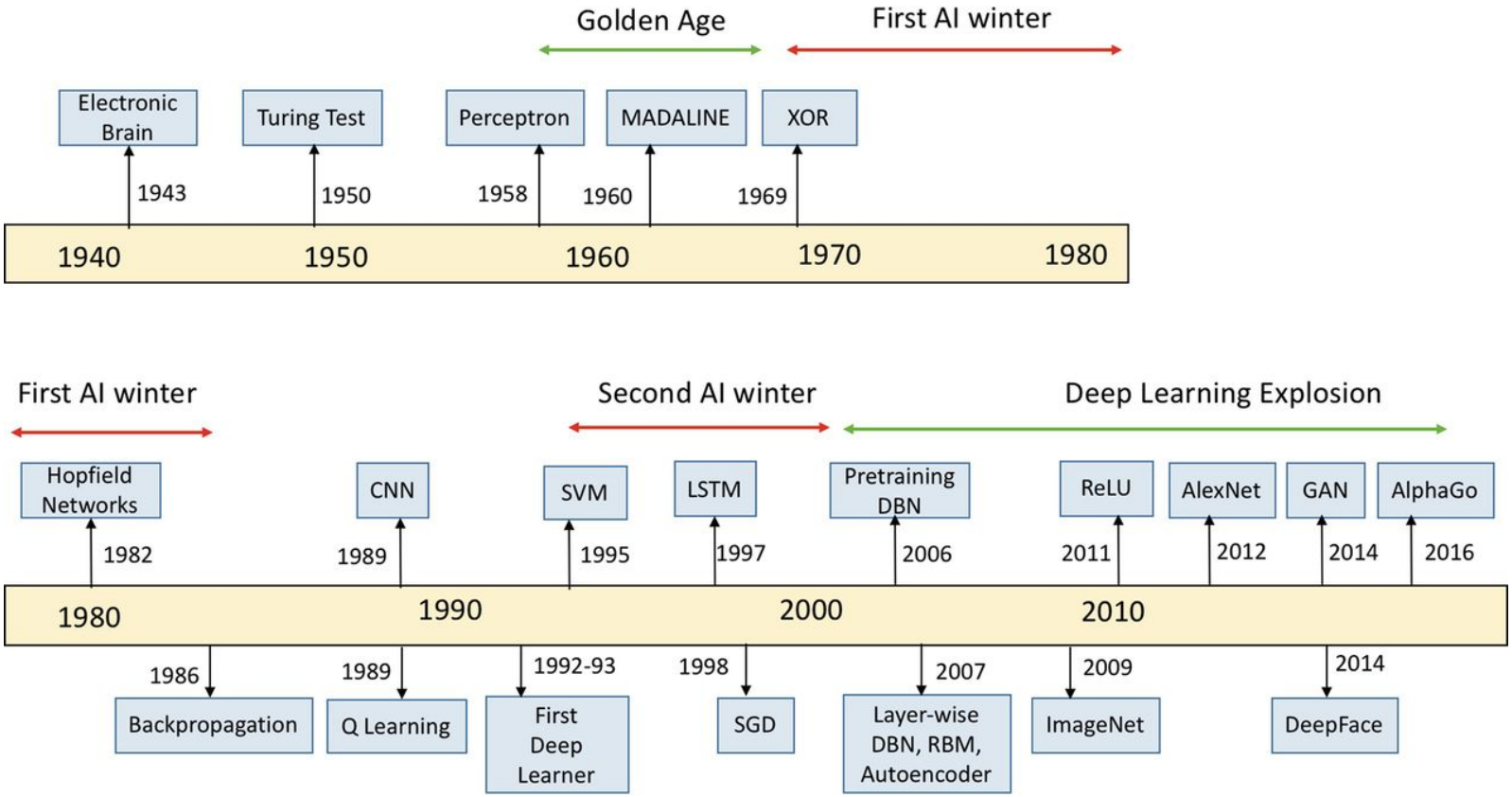
Deep learning algorithm structured similar to the organization of neurons in the brain

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

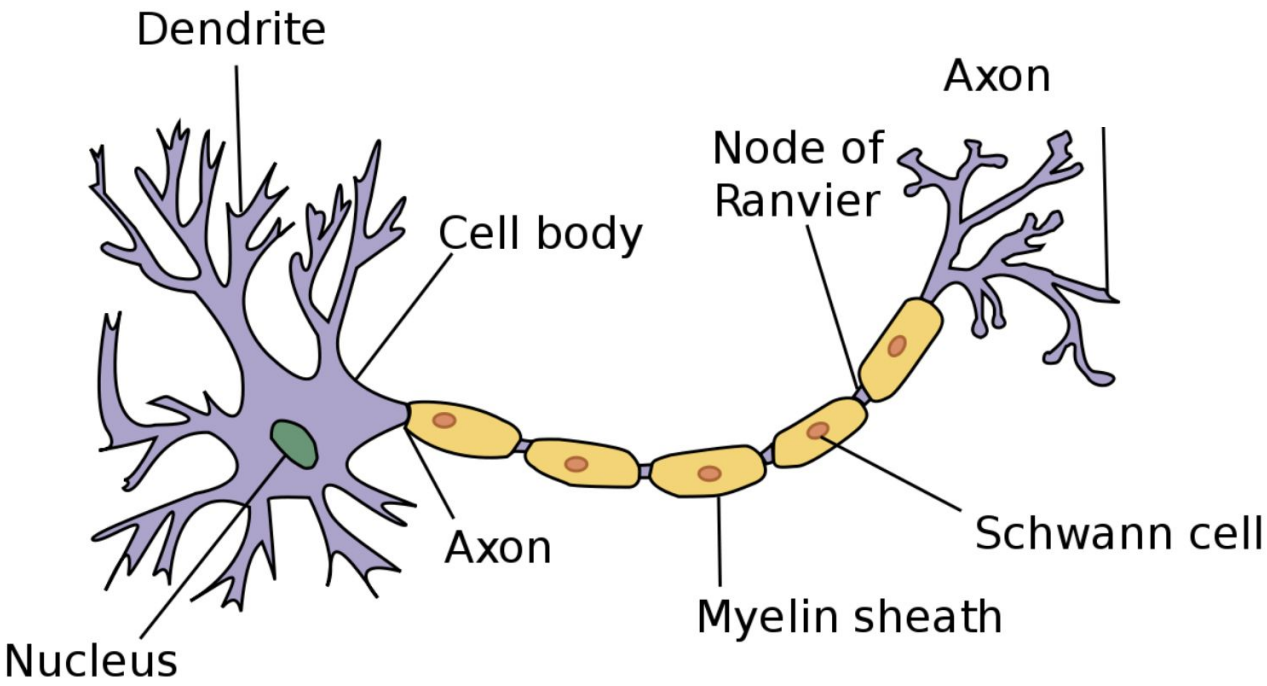
Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.



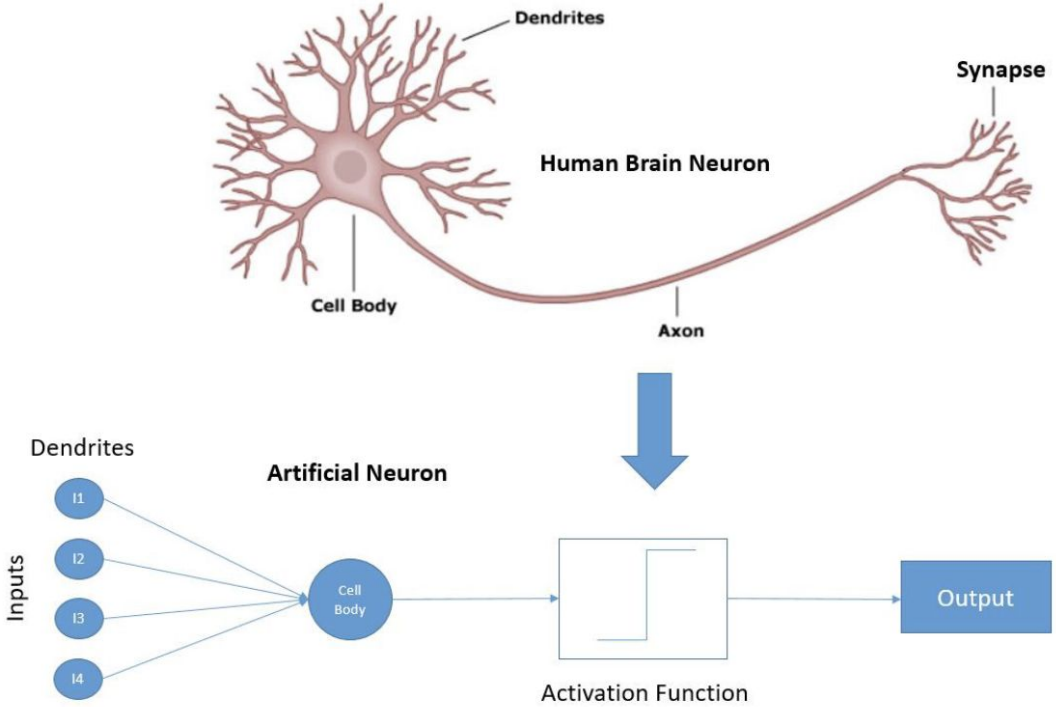
# Neural Networks: History and Timeline



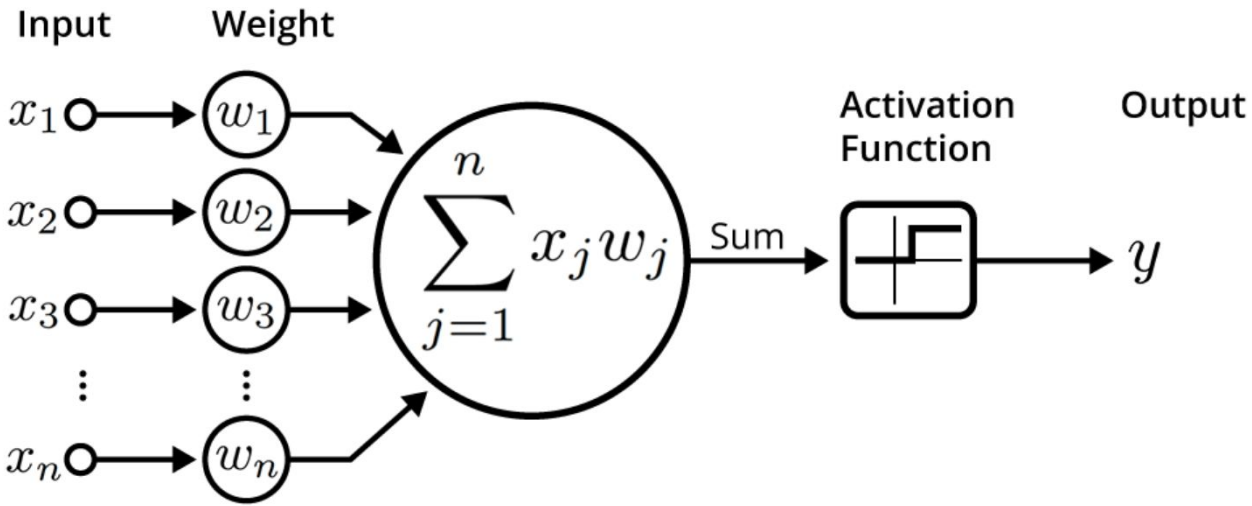
# Neural Networks



# Neural Networks

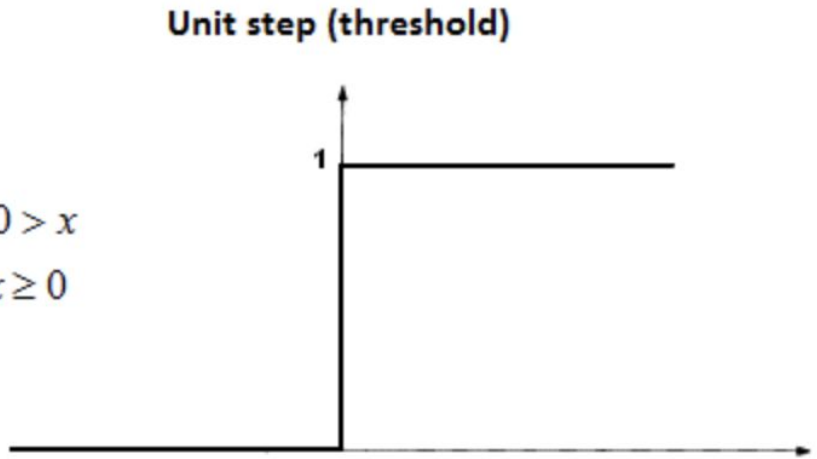


# Neural Networks

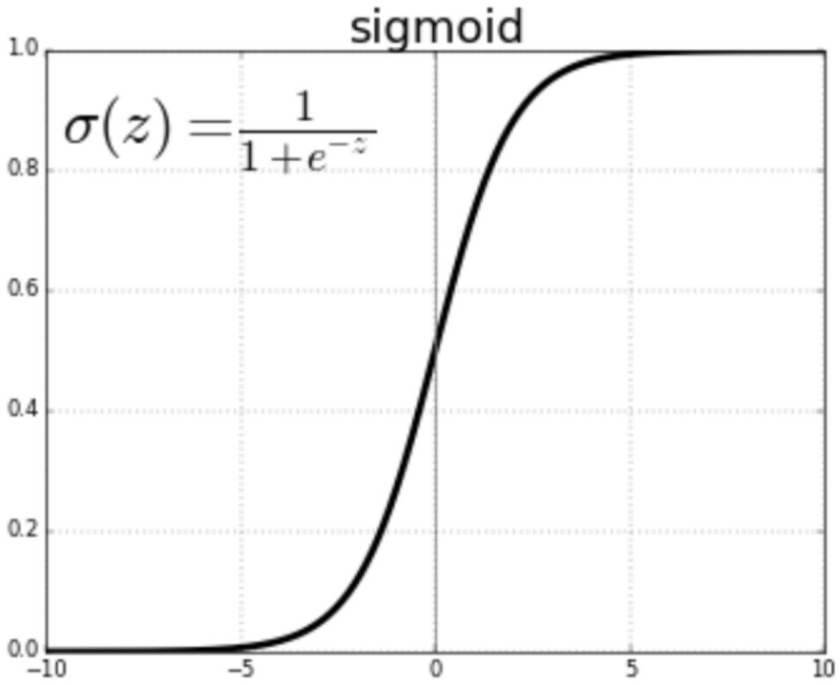


# Neural Networks: Activation Functions: Threshold

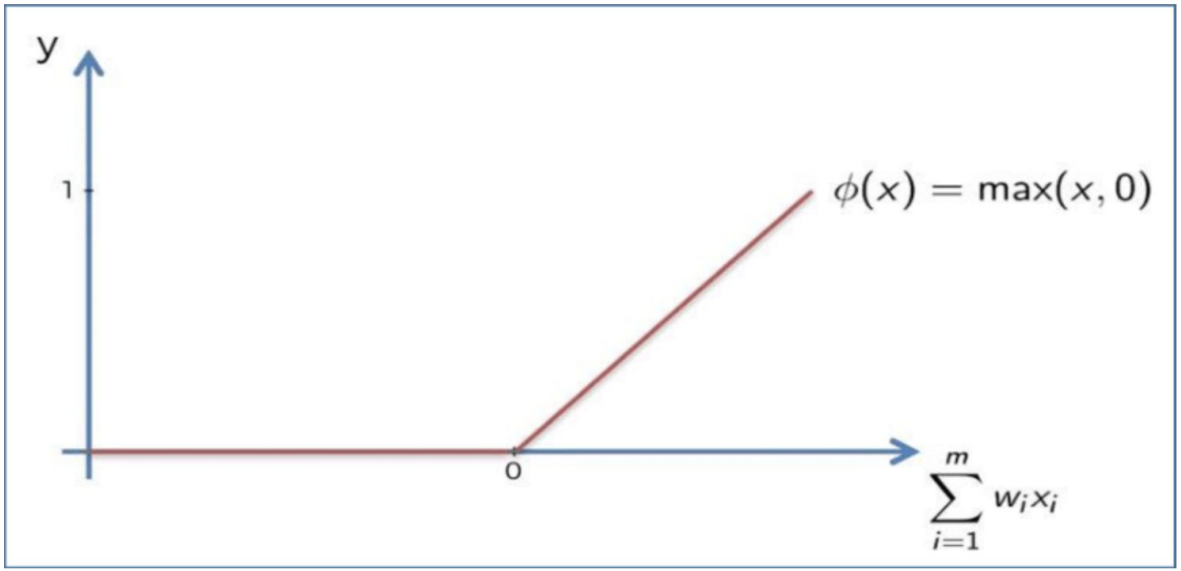
$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



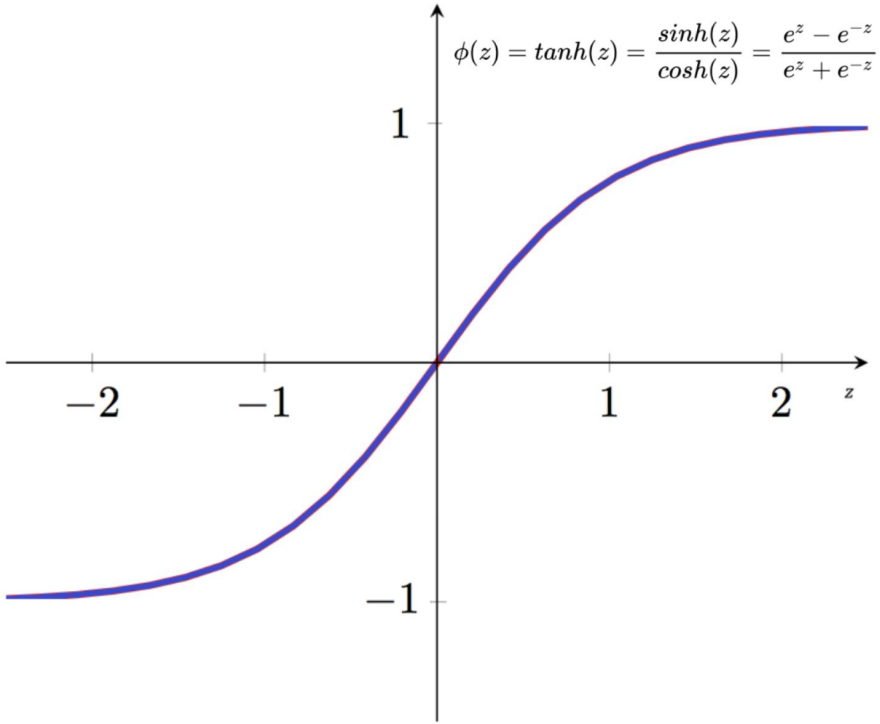
# Neural Networks: Activation Functions: Sigmoid



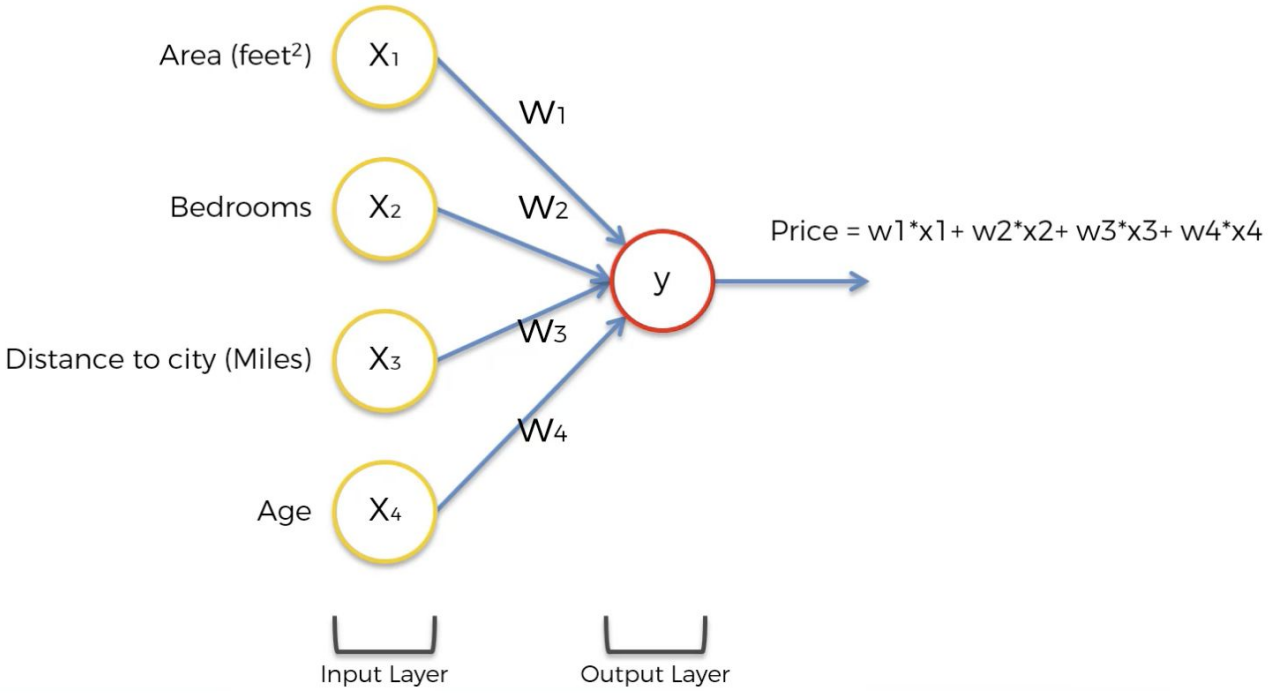
# Neural Networks: Activation Functions: Rectilinear



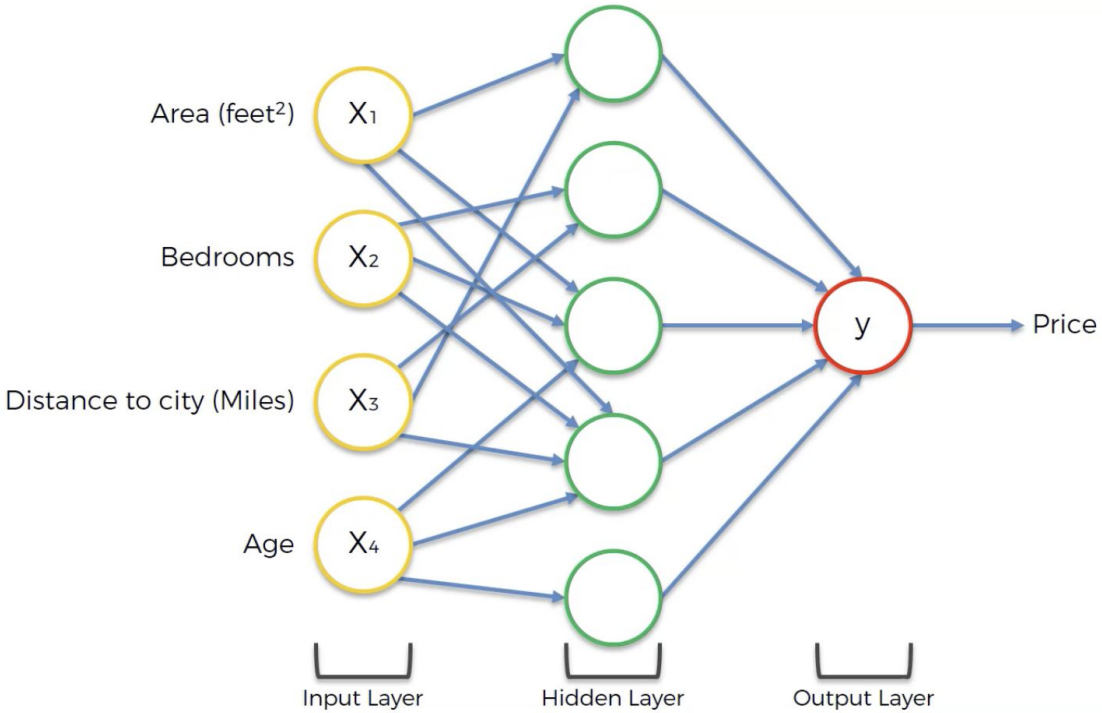
# Neural Networks: Activation Functions: Hyperbolic Tangent



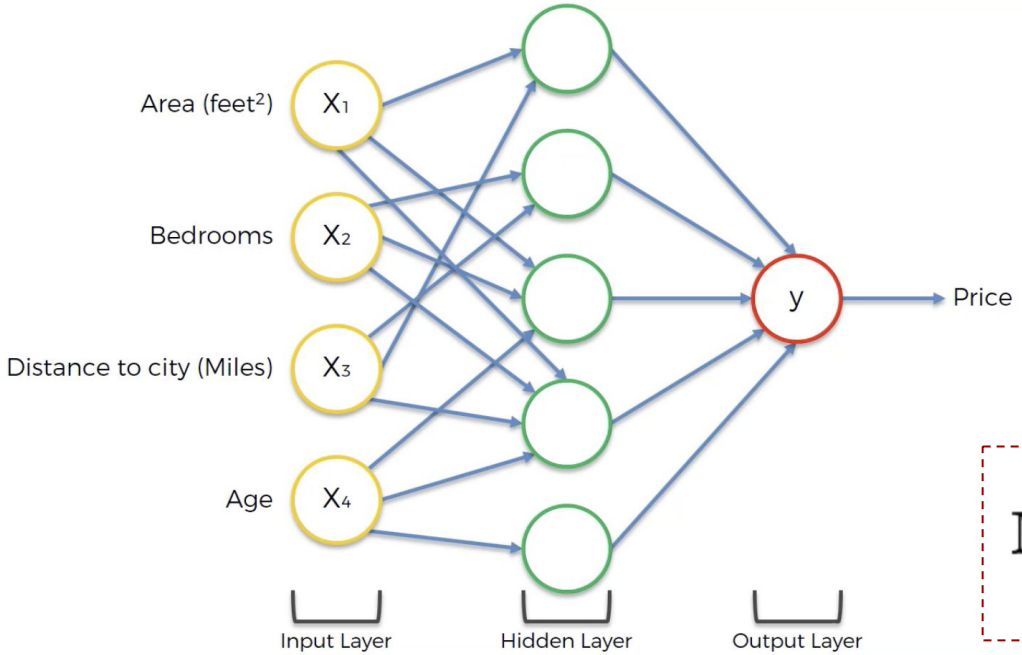
# Neural Networks: Example (Property Valuation)



# Neural Networks: Example (Property Valuation)



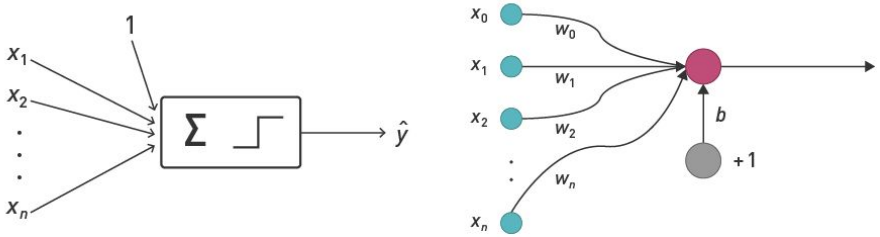
# Neural Networks: Example (Property Valuation)




Cost Function

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

# Neural Networks: Perceptron




- **Discriminative classifier:** learns decision boundary
- Perceptron fires if predicted value is positive

$$h(x_i) = b + \sum_{d=1}^{D} w_d x_{id}$$

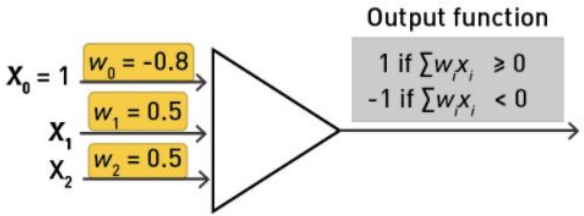
- *b* allows for nonzero threshold

$$output = \begin{cases} 0 \ if \ \sum_j w_j x_j < threshold \\ 1 \ if \ \sum_j w_j x_j > threshold \end{cases}$$

# Neural Networks: Perceptron (AND)

A perceptron for AND:

| $x_1$ | $x_2$ | $\Sigma w_1 x_1$ | Output |
|---|---|---|---|
| 1 | 1 | 0.2 | 1 |
| 1 | 0 | −0.3 | −1 |
| 0 | 1 | −0.3 | −1 |
| 0 | 0 | −0.8 | −1 |

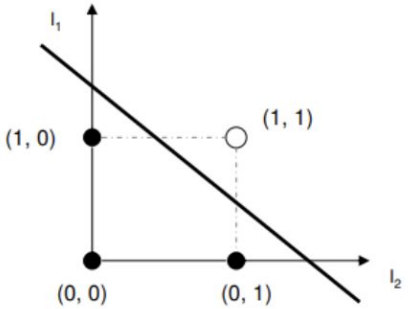



- Two weights and intercept

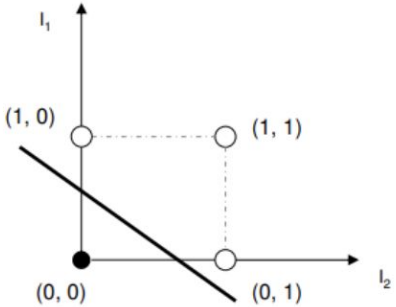
$$h(x_i) = b + w_1 x_{i1} + w_2 x_{i2}$$

# Neural Networks: Perceptron

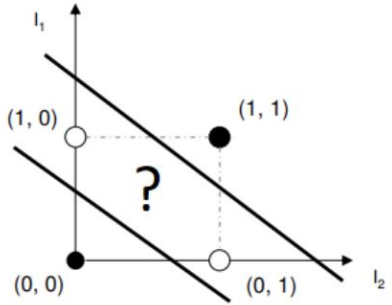
| AND | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




| OR | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




| XOR | | |
|---|---|---|
| $I_1$ | $I_2$ | out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



# Neural Networks: Perceptron (XOR)

| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

→ $b \leq 0$

→ $b + w2 > 0$

→ $b + w1 > 0$

→ $b + w1 + w2 \leq 0$

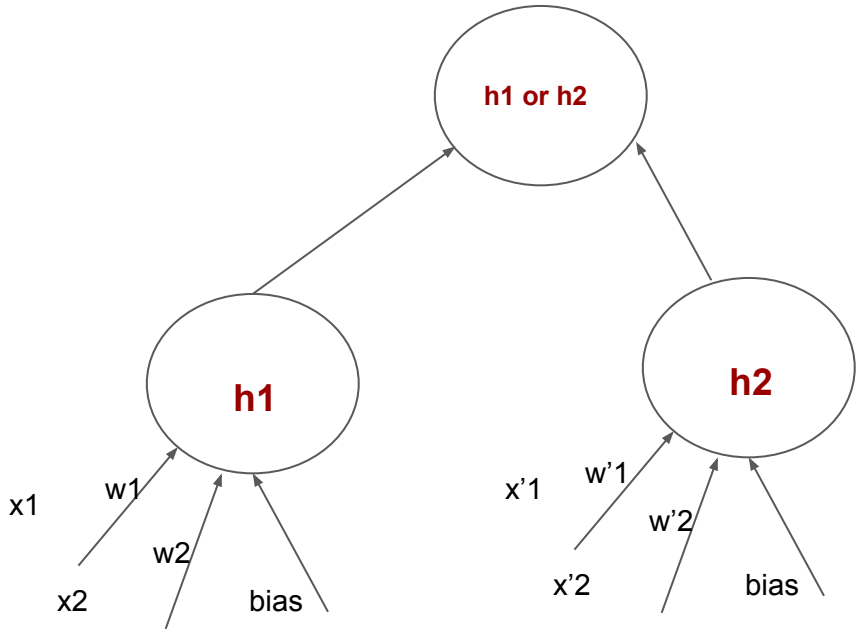
$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

# Neural Networks: Perceptron (XOR)

| | | | h1 | h2 | |
|---|---|---|---|---|---|
| X1 | X2 | Y | X1 AND ⌐X2 | ⌐X1 AND X2 | h1 OR h2 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |

$$y = \begin{cases} 0, & \text{if } w \cdot x + b \leq 0 \\ 1, & \text{if } w \cdot x + b > 0 \end{cases}$$

# Neural Networks: Perceptron (XOR)



# Perceptron: Rosenblatt's Algorithm

initialize weights randomly
while termination condition is not met:
    initialize $\Delta w_j = 0$
    for each training example $(X_i, Y_i)$:
        compute predicted output $\hat{Y}_i$
        for each weight $w_j$:
            $$\Delta w_j = \Delta w_j + \eta\left(Y_i - \hat{Y}_i\right) X_i$$
    for each weight $w_j$:
        $w_j = w_j + \Delta w_j$

# Neural Networks: Gradient Descent

- Training rule (Rosenblatt)

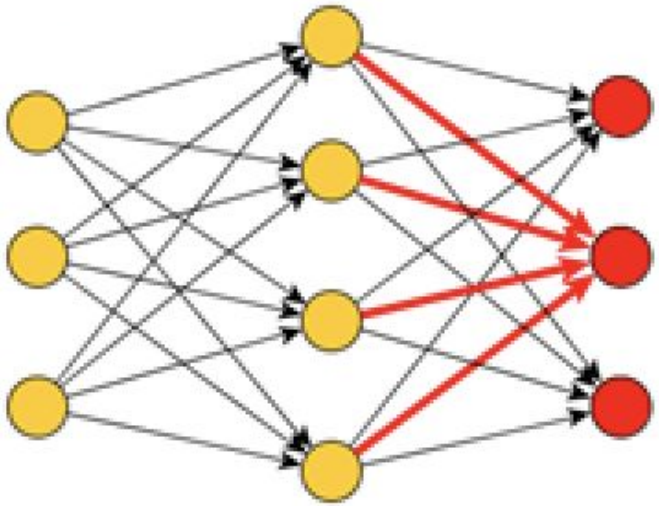
$$\Delta w_j = \Delta w_j + \eta(Y_i - \hat{Y})X_i$$

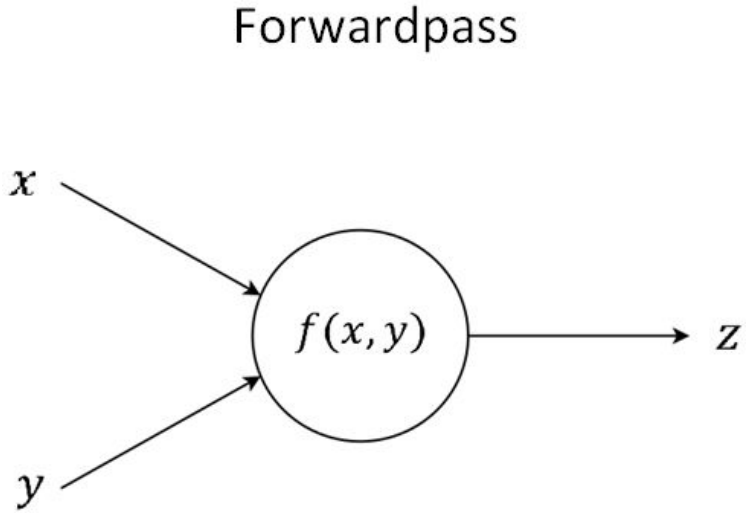
- Gradient descent (stochastic)

$$\beta <- \beta + R(Y_i - \hat{Y}_i)X_i$$

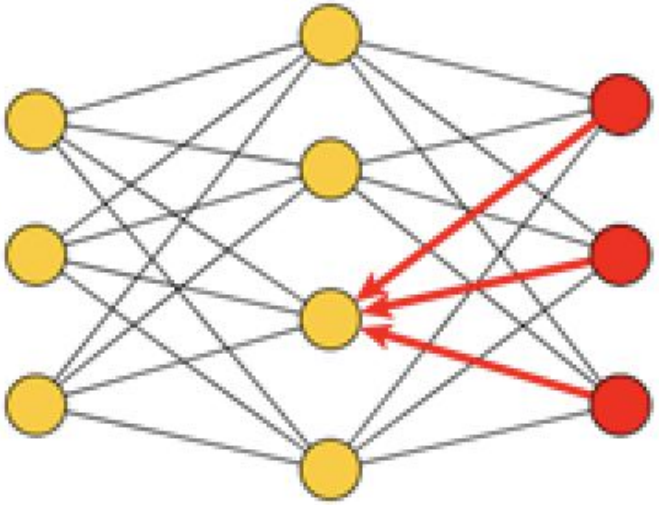
- Key is the $\hat{Y}_i$.
  - Perceptron: $\hat{Y}_i$ is step function; either 0 or 1.
  - Gradient descent: $\hat{Y}_i$ is smooth function; continuous.
  - Gradient provides continuous surface

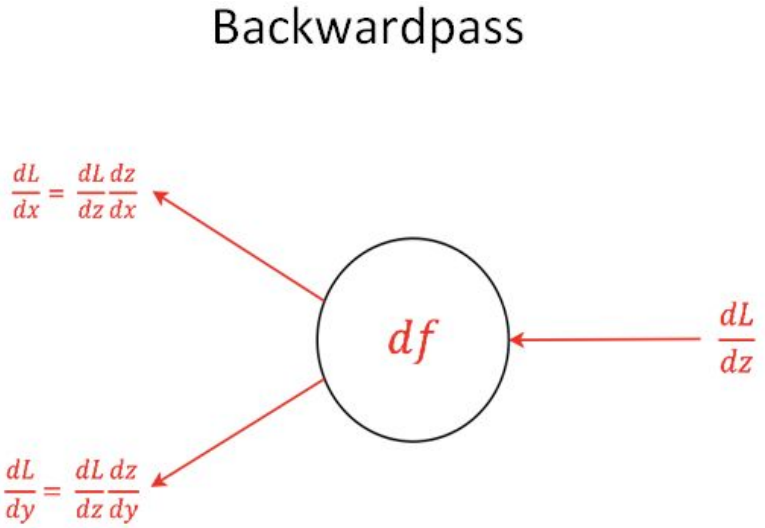
# Neural Networks: Forward Propagation





# Neural Networks: Backpropagation





# Neural Networks