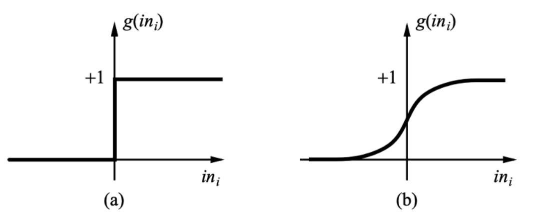
Lab 6

Neural Networks

Neuron model - Perceptron

$$w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n = \mathbf{w}^T\mathbf{x}$$

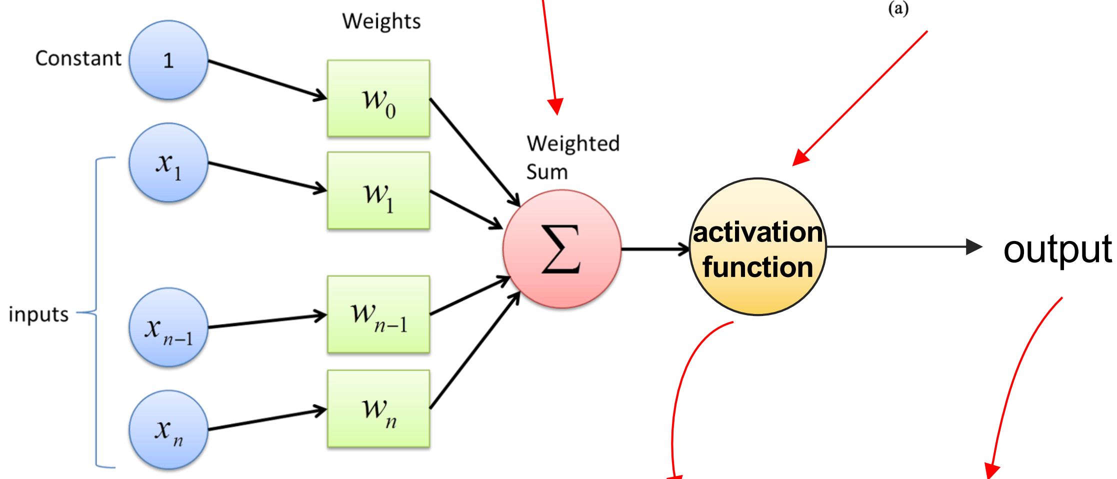


$x_0 = 1$

w_0 is the *bias* weight

Weights shows the strength of the particular node.

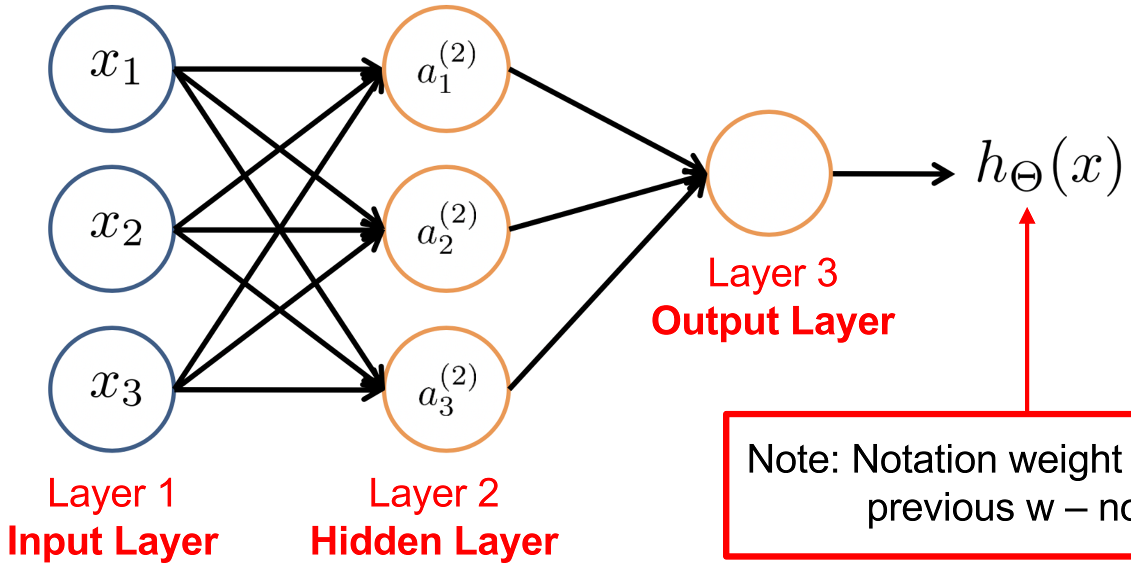
A **bias** value allows you to shift the activation function curve up or down.



logistic regression: $h_w(x) = g(w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n)$

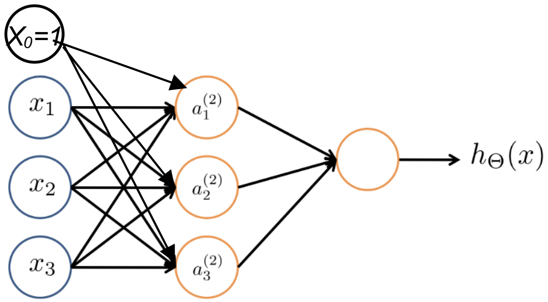
Artificial Neural Network

Multi-Layer Perceptrons architecture or Vanilla Neural Network



Note: Notation weight factor:
previous w – now Θ

Artificial Neural Network



$a_i^{(j)} =$ "activation" of unit i in layer j

$\Theta^{(j)} =$ matrix of weights controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)})$$

g is the **activation function** (e.g. Sigmoid)

vectors multiplication

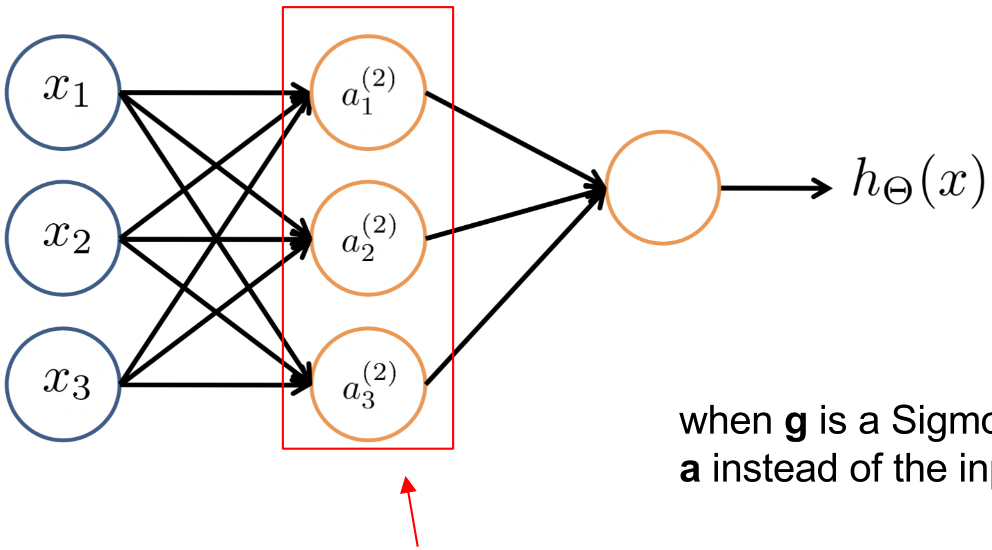
In the example:

$$\Theta^{(1)} \in \mathbb{R}^{3\times 4}$$

If network has s_j units in layer j, s_{j+1} units in layer $j+1$, then $\Theta^{(j)}$ will be of dimension $s_{j+1} \times (s_j + 1)$.

Artificial Neural Network

Forward Propagation



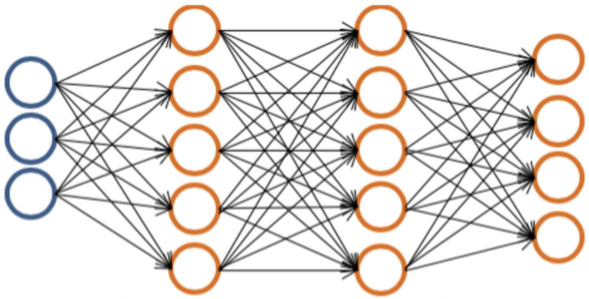
$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$
$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

when **g** is a Sigmoid, it's doing Logistic regression on **a** instead of the input features **x**…

The features vector **a** is *learned* from the input features!

Neural Networks

Multiple output units



$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$ → $y^{(i)}$ one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

pedestrian car motorcycle truck

Neural Networks - Cost Function

Logistic Regression:

m samples in the training set

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Neural Network: Instead of having 1 logistic regression output unit, we have **K** of them

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$(h_\Theta(x))_i = i^{th}$ output

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

(Think of $\mathrm{cost(i)} \approx (h_\Theta(x^{(i)}) - y^{(i)})^2$)

"how well is the network doing at predicting example *i*?"

Andrew Ng - Machine Learning: https://www.coursera.org/learn/machine-learning

Neural Networks - Backpropagation Algorithm

Minimizing the Cost Function

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$$\min_\Theta J(\Theta)$$

compute $J(\Theta)$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$$

$$\Theta_{\boxed{i}\boxed{j}}^{(\boxed{l})}$$

Layer of the NN

weight associated from neuron ***i*** of the previous later to neuron ***j***

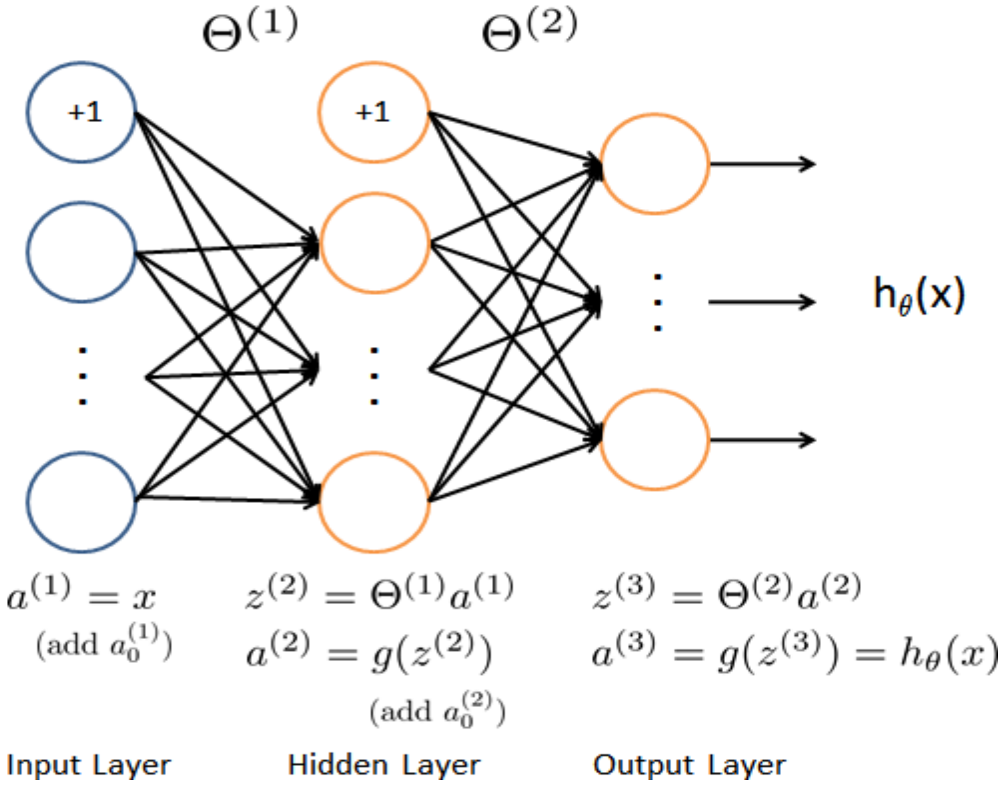
Lab 6. 2 – assignment Feedforward Propagation and Prediction

.
.
.

The 20 by 20 grid of pixels is “unrolled” into a 400-dimensional vector
5000 training examples.

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

[5000 x 400]

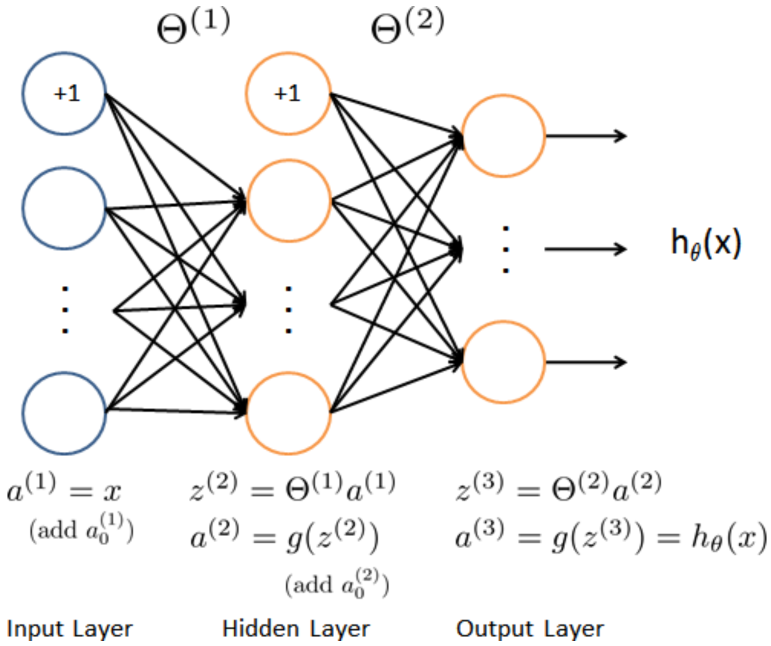


NN model

($\Theta^{(1)}$, $\Theta^{(2)}$) provided from the lab Theta1 and Theta2.

Lab 6. 2 – assignment Feedforward Propagation and Prediction

Implementation of Neural Network



- ($\Theta^{(1)}$, $\Theta^{(2)}$) provided from the lab
- Calculate
  1. Add $X_0 = 1$ to X matrix
  2. Compute $z^{(2)}$, $a^{(2)}$
  3. Add $a^{(2)}_0 = 1$ to $a^{(2)}$
  4. Compute $z^{(3)}$, $a^{(3)} = h_\theta(x)$
  5. Find max ($h_\theta(x)$)

Note: $h_\theta(x)$ = [5000 x 10]
Why?