Feed-forward Neural Networks

Lecture 8 - DAMLF | ML1

Frankfurt School

Motivation: Non-Linear Hypotheses

Random Forests

Very good out-of-the-box performance.

However, random forests presume a set of inherent, well-defined features.

Reasonable in many business & finance applications, which is why they are popular.

What if you have a problem in which the inherent features are **not** well-defined?

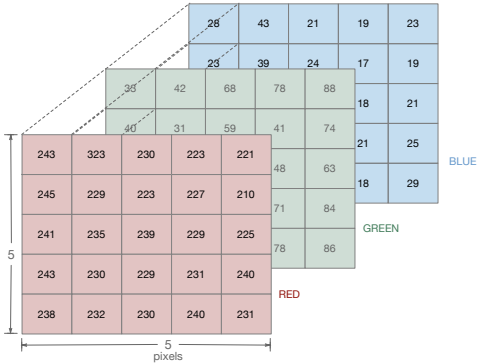
Motivation: Computer Vision

What you see

Motivation: Computer Vision

What you see

What the computer sees†



Motivation: Computer Vision

What you see

What the computer sees†

$$\underset{\text{RED}}{\begin{bmatrix} 243 & 232 & 230 & 223 & 221 \\ 245 & 229 & 223 & 227 & 210 \\ 241 & 235 & 239 & 229 & 225 \\ 243 & 230 & 229 & 231 & 240 \\ 238 & 232 & 230 & 240 & 231 \end{bmatrix}} \quad \underset{\text{GREEN}}{\begin{bmatrix} 33 & 42 & 68 & 78 & 88 \\ 40 & 31 & 59 & 41 & 74 \\ 39 & 38 & 53 & 48 & 63 \\ 51 & 36 & 38 & 71 & 84 \\ 43 & 70 & 29 & 78 & 86 \end{bmatrix}} \quad \underset{\text{BLUE}}{\begin{bmatrix} 28 & 43 & 21 & 19 & 23 \\ 23 & 39 & 24 & 17 & 19 \\ 31 & 28 & 19 & 18 & 21 \\ 20 & 27 & 16 & 21 & 25 \\ 27 & 32 & 17 & 18 & 29 \end{bmatrix}}$$

† a 5 x 5 pixel patch of the red ball has 25 x 3 = **75 pixels** (features).

Motivation: Computer Vision

What you see

What the computer sees

640 x 480 pixels = **921,600** Features

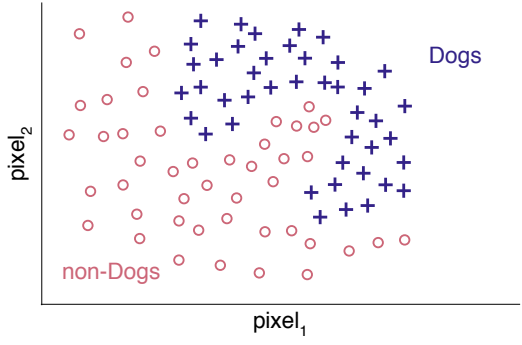
1200 x 1800 pixels = **6,480,000** Features % 10 x 15 cm image, 300 dpi

Dogs

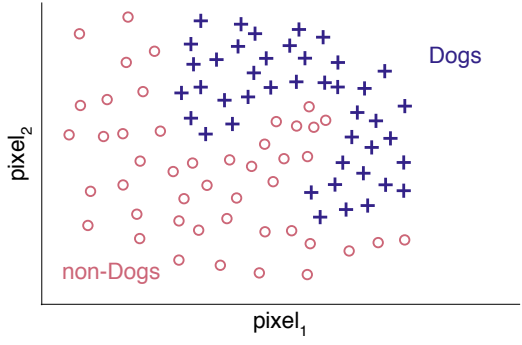
Dogs

non-Dogs

Classification



Classification



- *now simply add 1 million (or more) pixels*
- *and no idea about any inherent/general features*

Image and Language Processing

New Features

Natural language processing (NLP) is a straightforward example. 'Reading' all of the financial reports and extracting information automatically is an obvious application.

Image and Language Processing

New Features

Natural language processing (NLP) is a straightforward example. 'Reading' all of the financial reports and extracting information automatically is an obvious application.

Imagine processing may be less obvious.

Image and Language Processing

Remote Sensing Metrics/via CNBC

New Features

Natural language processing (NLP) is a straight-forward example. 'Reading' all of the financial reports and extracting information automatically is an obvious application.

Imagine processing may be less obvious.

- Cars in retail parking lots

Image and Language Processing

New Features

Natural language processing (NLP) is a straight-forward example. 'Reading' all of the financial reports and extracting information automatically is an obvious application.

Imagine processing may be less obvious.

- Cars in retail parking lots
- Shadows of oil storage tanks
  (Ras Tunura refinery, Saudi Arabia)

Image and Language Processing

New Features

Natural language processing (NLP) is a straight-forward example. 'Reading' all of the financial reports and extracting information automatically is an obvious application.

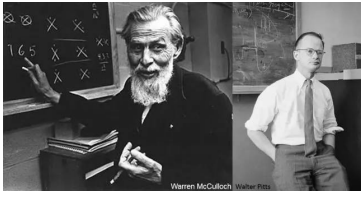
Imagine processing may be less obvious.

- Cars in retail parking lots
- Shadows of oil storage tanks
  (Ras Tunura refinery, Saudi Arabia)

Applications will increase with IoT and 5G

Neural Networks

Some History

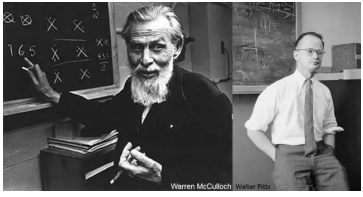


Warren McCulloch & Walter Pitts

A logical calculus of the ideas immanent in nervous activity, *Bul Math Biophys* 5:115–133, 1943

Neural networks refer to algorithms that ***mimic the brain***.

Some History



Warren McCulloch & Walter Pitts

A logical calculus of the ideas immanent in nervous activity, *Bul Math Biophys* 5:115–133, 1943

Neural networks refer to algorithms that ***mimic the brain***.

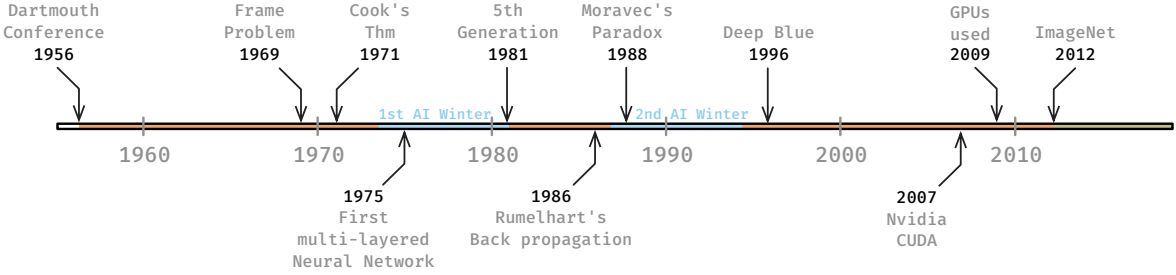
Two approaches:

- Biological process models
- Artificial neural networks (ANNs)

The One Algorithm Hypothesis

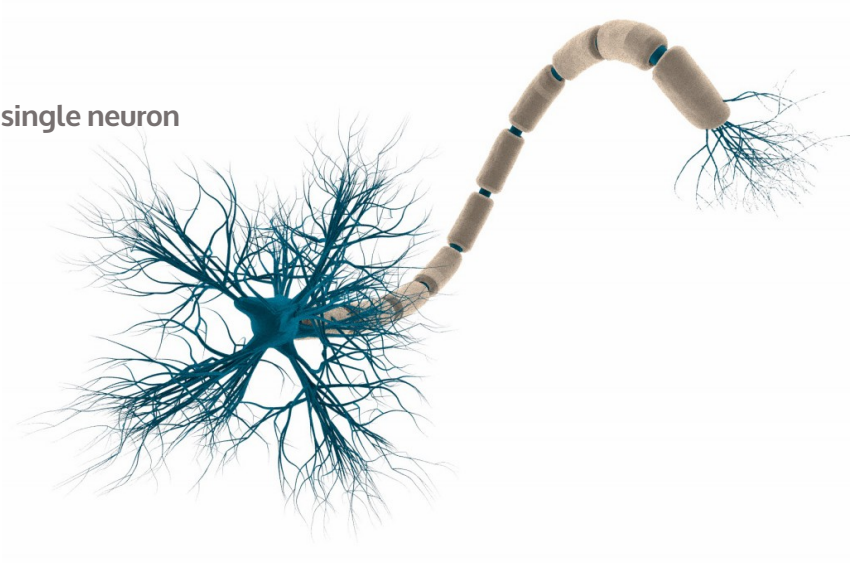
*Rather than 1000s of individual algorithms for different cognitive functions, perhaps the brain uses **just one**.*

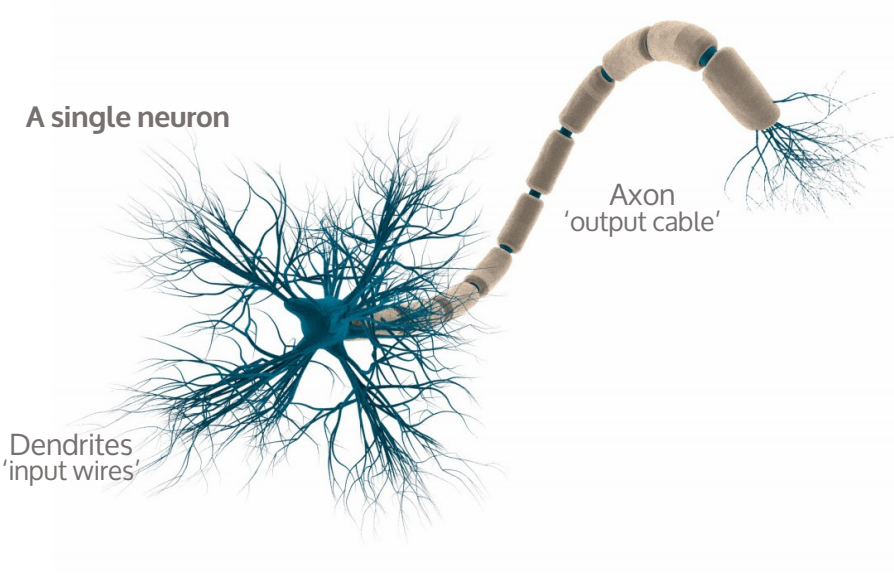
Potted History of AI

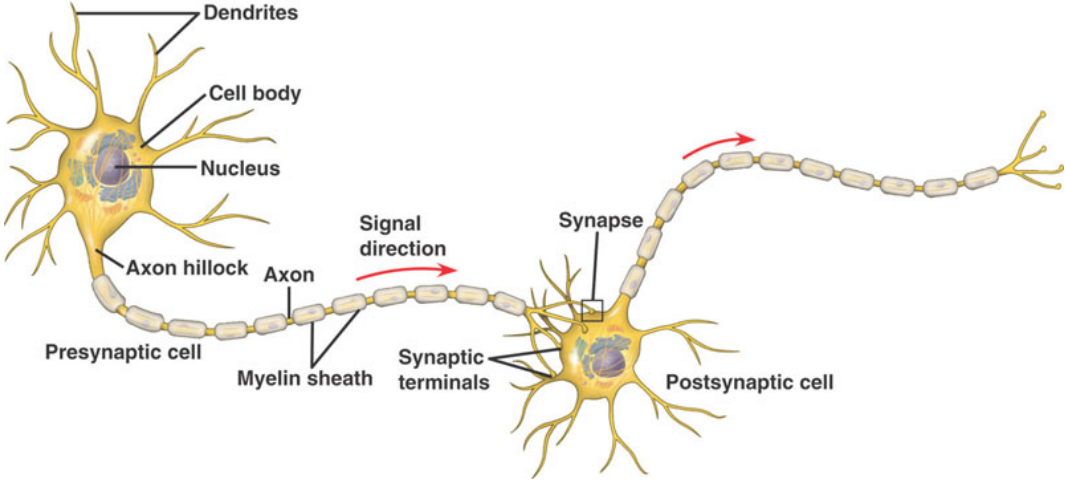


Neural Networks: Hypothesis Representation

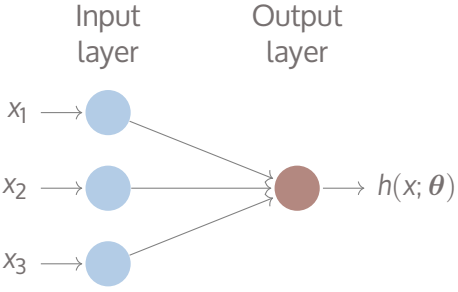
A single neuron



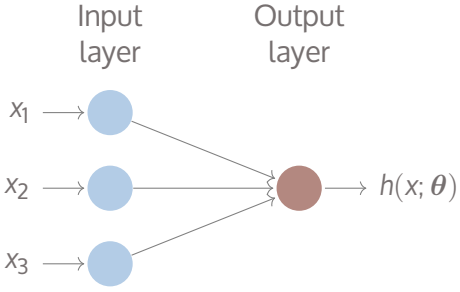

A single neuron
Axon
'output cable'
Dendrites
'input wires'


Dendrites
Cell body
Nucleus
Axon hillock
Axon
Signal direction
Synapse
Presynaptic cell
Myelin sheath
Synaptic terminals
Postsynaptic cell

Neuron model: logistic unit



Neuron model: logistic unit

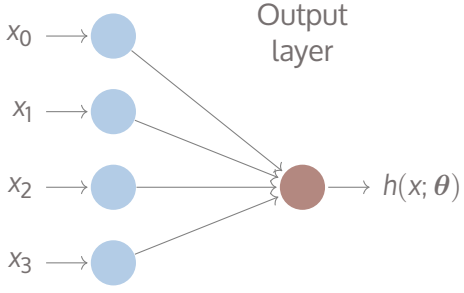


Recall:

$$h(x;\theta) = \frac{1}{1+e^{-\theta^T x}}$$

where: $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$

Neuron model: logistic unit



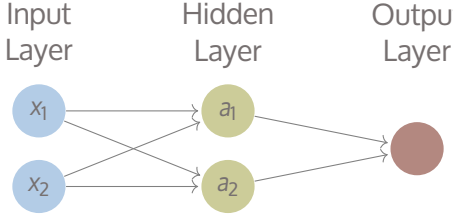
Note:

$x_0 = 1$

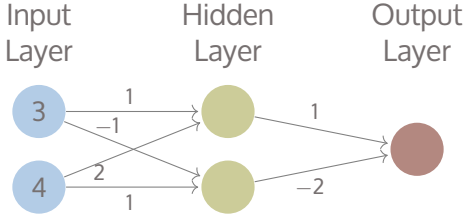
called the '**bias unit**'

where: $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$

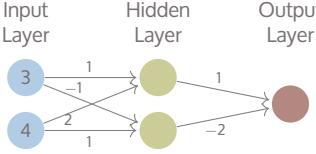
Adding Layers



Input Data, Output Prediction



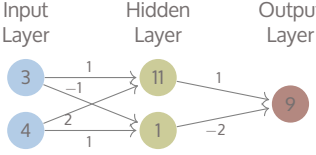
Forward Propagation Intuition



Forward Propagation

- Input Data
- Dot product *multiply then add*
- One-data-point at a time
- Last output is the model's prediction

Forward Propagation Intuition



Forward Propagation

In [1]:
```
import numpy as np
```

In [2]:
```
# Forward Propagation with
# Identity Activation Function
input_nodes = np.array([3,4])

Theta = {'node_a1': np.array([1, 2]),
         'node_a2': np.array([-1, 1]),
         'output': np.array([1, -2])}

node_a1_value = (input_nodes * Theta['node_a1']).sum()
node_a2_value = (input_nodes * Theta['node_a2']).sum()
```

In [3]:
```
hidden_layer_values = np.array([node_a1_value, node_a2_value])
print(hidden_layer_values)
```

```
[11  1]
```

In [4]:
```
output = (hidden_layer_values * Theta['output']).sum()
print(output)
```

```
9
```

Activation Function

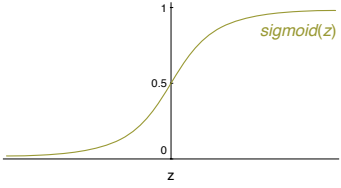
Non-Linearity

Activation functions:

- applied to **node inputs** to produce **node output**
- capture **non-linear relationships** in data

Examples:

Activation Function



Non-Linearity

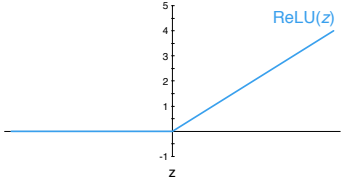
Activation functions:

- applied to **node inputs** to produce **node output**
- capture **non-linear relationships** in data

Examples:

Sigmoid Function

Activation Function



Non-Linearity

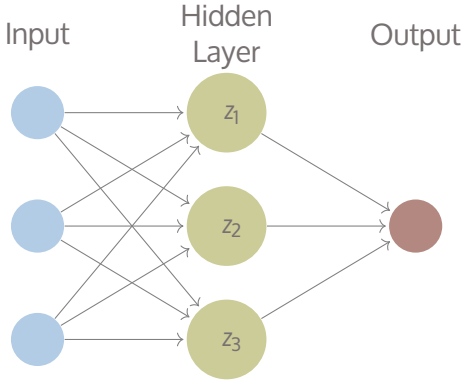
Activation functions:

- applied to **node inputs** to produce **node output**
- capture **non-linear relationships** in data

Examples:

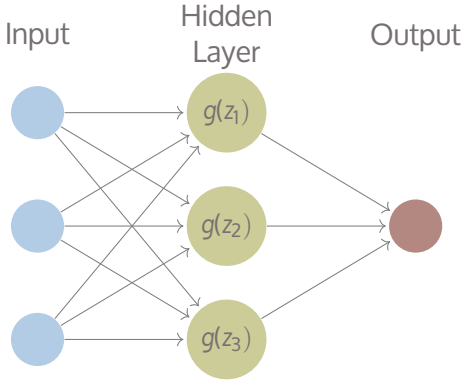
Rectified Linear Unit (ReLU)

Activation Function



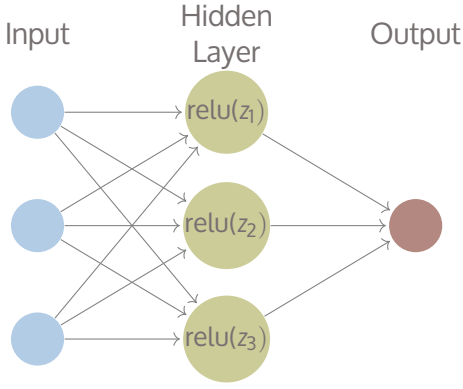
z_1 = node_a1_value
z_2 = node_a2_value
z_3 = node_a3_value

Activation Function



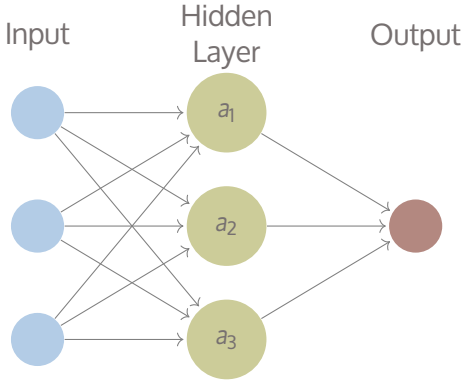
z_1 = node_a1_value
z_2 = node_a2_value
z_3 = node_a3_value
$g(z_i)$ = sigmoid activation

Activation Function



z_1 = node_a1_value
z_2 = node_a2_value
z_3 = node_a3_value
$relu(z_i) = \text{ReLU}$

Activation Function



z_1 = node_a1_value
z_2 = node_a2_value
z_3 = node_a3_value
a_i = activation_fun(z_i)

Terminology

Features: x
Inputs: x

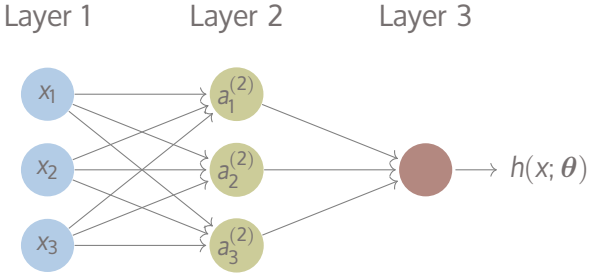
Parameters: θ
Weights: θ

Activation Function

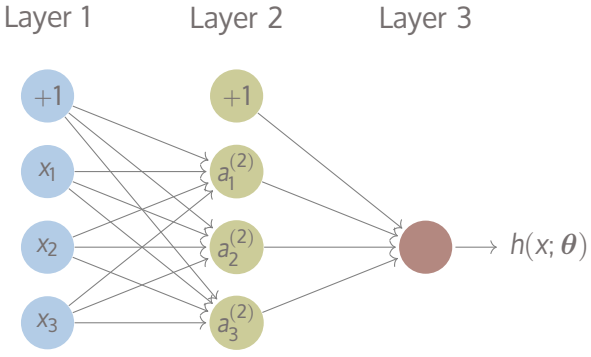
Some types of activation functions:

- Logistic regression hypothesis (Sigmoid)
- Hyperbolic Tangent (tanh) *# rescaled sigmoid to (-1, +1)*
- Rectified Linear Unit (ReLU)
- Normalized Exponential Function (Softmax)

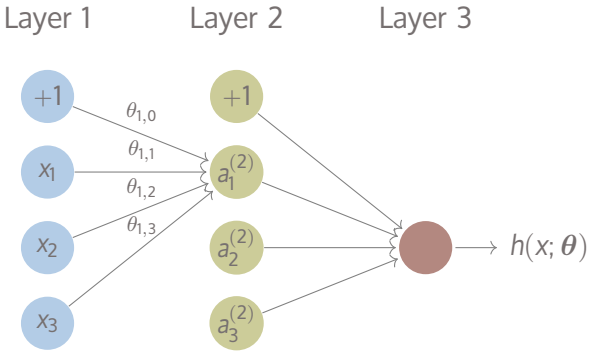
Neural Network Bias Nodes



Neural Network Bias Nodes



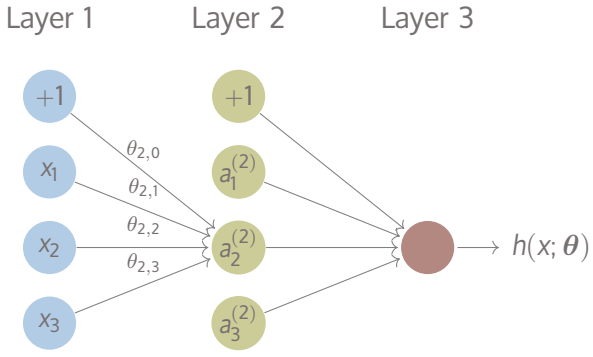
Neural Network Bias Nodes



Highlighted Step:

$$\theta_{1,0} + \theta_{1,1}x_1 + \theta_{1,2}x_2 + \theta_{1,3}x_3 = z_1^{(2)} \quad \Rightarrow \quad g(z_1^{(2)}) = a_1^{(2)}$$

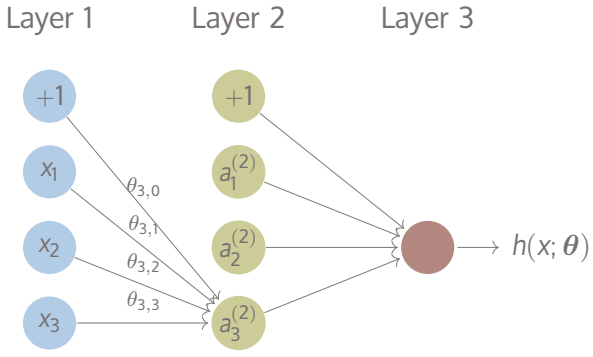
Neural Network Bias Nodes



Highlighted Step:

$$\theta_{2,0} + \theta_{2,1}x_1 + \theta_{2,2}x_2 + \theta_{2,3}x_3 = z_2^{(2)} \quad \Rightarrow \quad g(z_2^{(2)}) = a_2^{(2)}$$

Neural Network Bias Nodes



Highlighted Step:

$$\theta_{3,0} + \theta_{3,1}x_1 + \theta_{3,2}x_2 + \theta_{3,3}x_3 = z_3^{(2)} \quad \Rightarrow \quad g(z_3^{(2)}) = a_3^{(2)}$$

Comparison to Logistic Regression

Recall the form of the hypothesis, $h(x; \boldsymbol{\theta})$, for **logistic regression**.

For example,

$$\begin{aligned} h(x; \boldsymbol{\theta}) &= g\left(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2\right) \\ &= g\left(\theta^T x\right) \end{aligned}$$

with **parameter** vector θ and **feature** vector x:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Comparison to Logistic Regression

Recall the form of the hypothesis, $h(x;\boldsymbol{\theta})$, for **logistic regression**.

For example,

$$\begin{aligned} h(x;\boldsymbol{\theta}) &= g\left(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2\right) \\ &= g\left(\theta^T x\right) \end{aligned}$$

with **parameter** vector θ and **feature** vector x:

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Comparison to Logistic Regression

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Observe that $\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix}$ is a 1×3 **matrix**.

Comparison to Logistic Regression

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Observe that $\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix}$ is a 1×3 **matrix**.
Let's call it $\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix}$.

Comparison to Logistic Regression

We can also relabel the elements of Θ using the convention $\Theta_{i,j}$ to refer to the element in the ith row and jth column of Θ.

$$\Theta = \begin{bmatrix} \Theta_{10} & \Theta_{11} & \Theta_{12} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Comparison to Logistic Regression

We can also relabel the elements of Θ using the convention $\Theta_{i,j}$ to refer to the element in the ith row and jth column of Θ.

$$\Theta = \begin{bmatrix} \Theta_{10} & \Theta_{11} & \Theta_{12} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

So, our familiar logistic regression hypothesis can be rewritten (baroquely) as

$$\begin{aligned} h(x;\boldsymbol{\theta}) &= g\left(\Theta_{10}x_0 + \Theta_{11}x_1 + \Theta_{12}x_2\right) \\ &= g\left(\Theta x\right) \end{aligned}$$

Comparison to Logistic Regression

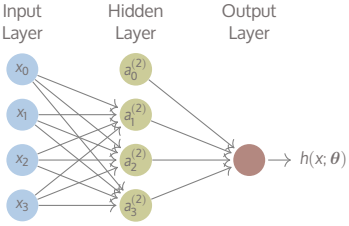
We can also relabel the elements of Θ using the convention $\Theta_{i,j}$ to refer to the element in the ith row and jth column of Θ.

$$\Theta^{(1)} = \begin{bmatrix} \Theta_{10}^{(1)} x_0 & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

So, our familiar logistic regression hypothesis can be rewritten (baroquely) as

$$\begin{aligned} h(x; \boldsymbol{\theta}) &= g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2\right) \\ &= g\left(\Theta^{(1)} x\right) \end{aligned}$$

A Neural Network

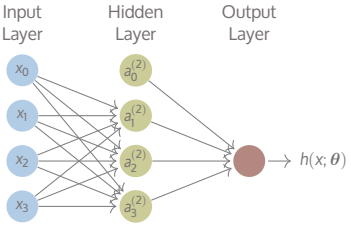


Notation:

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

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

A Neural Network



Notation:

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

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

Example:

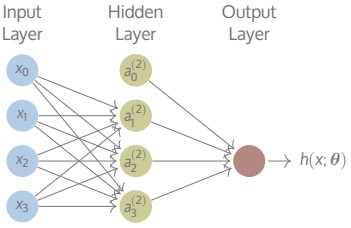
3 input units and 3 hidden units entails that $\Theta^{(1)}$ is a 3 x 4 matrix

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right)$$

A Neural Network



Notation:

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

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

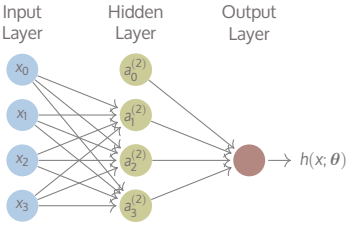
Example:

3 input units and 3 hidden units entails that $\Theta^{(1)}$ is a 3 x 4 matrix

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

In general, if layer j has s_j units and layer $j+1$ has s_{j+1} units, then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j + 1)$.

A Neural Network



Notation:

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

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

Example:

3 hidden units and 1 output unit entails that $\Theta^{(2)}$ is a 1 x 4 matrix

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right)$$

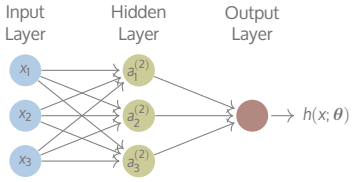
$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right)$$

$$a_1^{(3)} = g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}\right) = h(x;\theta)$$

The **vectorized** implementation of these calculations is called **forward propagation**.

Forward propagation computes the value of the output layer, $h(x; \theta)$, in terms of the values of inputs, $x_0 \ldots x_n$, propagated through the layers of a neural network.

Vectorized Implementation

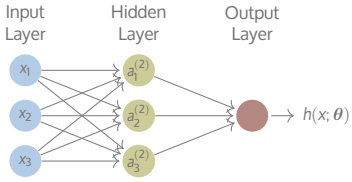


$$
\begin{aligned}
a_1^{(2)} &= g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right) \\
a_2^{(2)} &= g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right) \\
a_3^{(2)} &= g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right)
\end{aligned}
$$

Notation:

$$
\begin{aligned}
a_1^{(2)} &= g\left(z_1^{(2)}\right) \\
a_2^{(2)} &= g\left(z_2^{(2)}\right) \\
a_3^{(2)} &= g\left(z_3^{(2)}\right)
\end{aligned}
$$

Vectorized Implementation



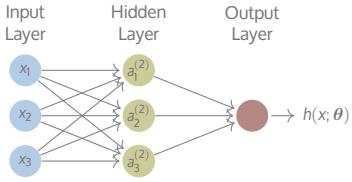
Notation:

$$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3$$

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

$$a_1^{(2)} = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(z_3^{(2)}\right)$$

Vectorized Implementation



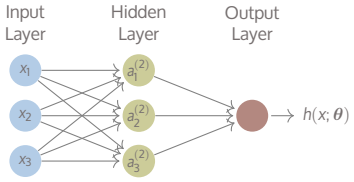
Notation:

$$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3$$
$$z_2^{(2)} = \Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3$$

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

$$a_1^{(2)} = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(z_3^{(2)}\right)$$

Vectorized Implementation



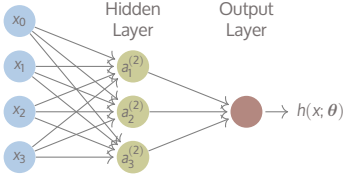
$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right)$$
$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right)$$
$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1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

Notation:

$$z_1^{(2)} = \Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3$$
$$z_2^{(2)} = \Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3$$
$$z_3^{(2)} = \Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3$$

$$a_1^{(2)} = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(z_3^{(2)}\right)$$

Vectorized Implementation



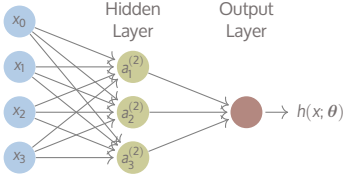
Notation:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

$$a_1^{(2)} = g\left(z_1^{(2)}\right)$$
$$a_2^{(2)} = g\left(z_2^{(2)}\right)$$
$$a_3^{(2)} = g\left(z_3^{(2)}\right)$$

Vectorized Implementation



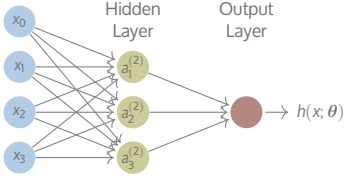
Notation:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

$$z^{(2)} = \Theta^{(1)} x$$
$$a^{(2)} = g\left(z^{(2)}\right)$$

Vectorized Implementation



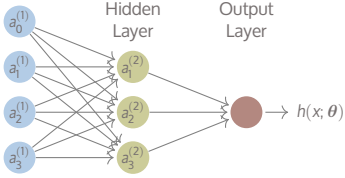
Notation:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

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

$$z^{(2)} = \Theta^{(1)} x$$
$$a^{(2)} = g\left(z^{(2)}\right)$$

Vectorized Implementation



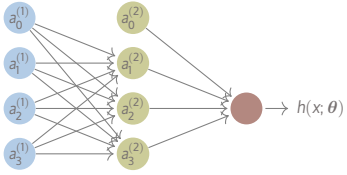
$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right)$$
$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right)$$
$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)}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

Notation:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad a^{(1)} = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g\left(z^{(2)}\right)$$

Vectorized Implementation



Notation:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad a^{(1)} = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{bmatrix}$$

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right)$$

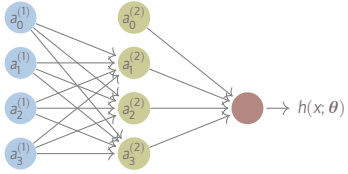
$$h(x;\theta) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right)$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

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

$$a_0^{(2)} = 1 = a_0^{(1)}$$

Forward Propagation



Equations to plug in:

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \qquad a^{(1)} = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{bmatrix}$$

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} a_0^{(1)} + \Theta_{11}^{(1)} a_1^{(1)} + \Theta_{12}^{(1)} a_2^{(1)} + \Theta_{13}^{(1)} a_3^{(1)}\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} a_0^{(1)} + \Theta_{21}^{(1)} a_1^{(1)} + \Theta_{22}^{(1)} a_2^{(1)} + \Theta_{23}^{(1)} a_3^{(1)}\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} a_0^{(1)} + \Theta_{31}^{(1)} a_1^{(1)} + \Theta_{32}^{(1)} a_2^{(1)} + \Theta_{33}^{(1)} a_3^{(1)}\right)$$

$$h(x;\theta) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right)$$

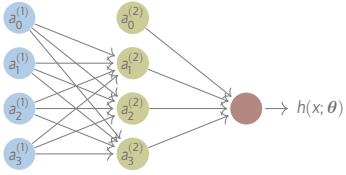
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

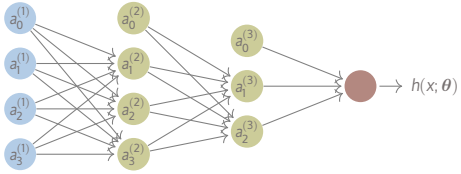
$$a^{(2)} = g\left(z^{(2)}\right)$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$a^{(3)} = h(x;\theta) = g\left(z^{(3)}\right)$$

Adding Hidden Layers





References

Nielsen, M. (2019).
Neural Networks and Deep Learning.
Determination Press.