



# Machine Learning

Abdelhak Mahmoudi
abdelhak.mahmoudi@um5.ac.ma

2020

# Content

1. The Big Picture
2. Supervised Learning
   - Linear Regression, Logistic Regression, Support Vector Machines, Trees, Random Forests, Boosting, Artificial Neural Networks
3. Unsupervised Learning
   - Principal Component Analysis, K-means, Mean Shift

# Supervised Learning

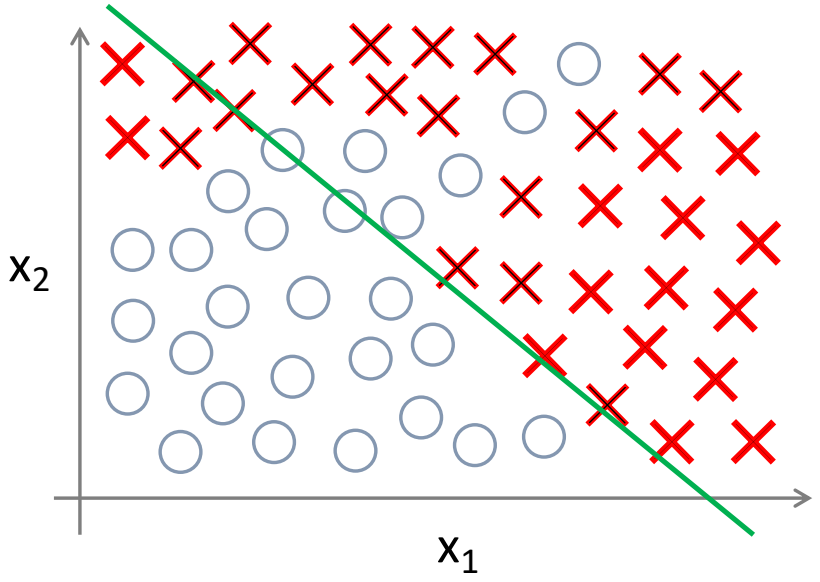
- Linear Regression
- Logistic Regression
- Support Vector Machines
- Trees (Decision  and Regression)
- Random Forests
- Boosting
- **Artificial Neural Networks**

# History

- Logical Neuron (1943 - Warren McCulloch and Walter Pitts)
  - Logical operations
  - No activation function
- Linear Threshold Unit
  - real numbers, weight,
  - step activation function
- Perceptron (1957 - Rosenblatt)
  - single layer of LTU
  - Trained with Hebb's rule
  - Weakness of perceptron (1969 - Marvin Minsky and Seymour Papert)
- Multi Layer Perceptron (MLP) (1986 – D. Rumelhart, G. Hinton, R. Williams)
  - Artificial Neural Network
  - Sigmoid activation function
  - Backpropagation
- Deep Neural Networks
  - More data, more power
  - More layers
  - Different activation functions
  - Different Architectures

# Non Linear Classification

A linear model will certainly underfit !

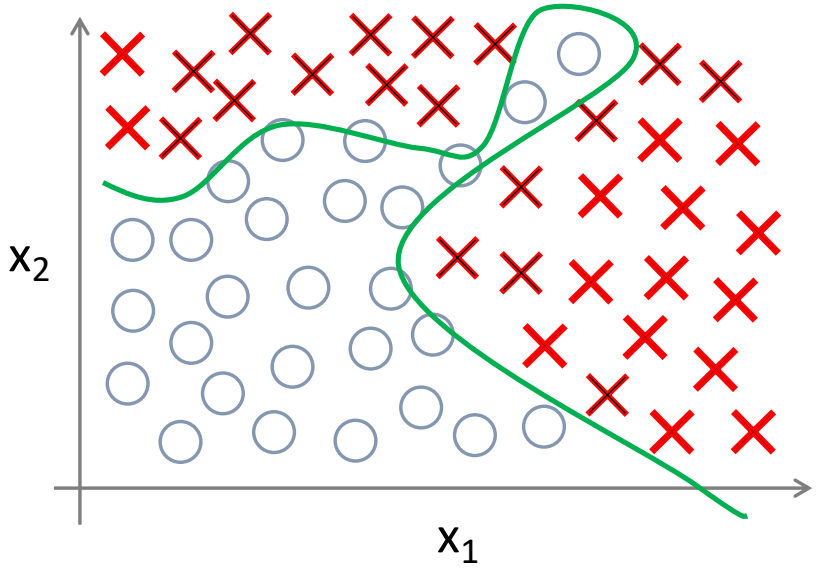



$$g(w_0 + w_1 x_1 + w_2 x_2)$$

# Non Linear Classification

A non linear model in a high dimensional space may be a solution, but…

… what if we have many many features?!




$$g(w_0 + w_1x_1 + w_2x_2 + \cdots + w_{12}x_1^2 + w_{13}x_2^2 \ldots + w_{25}x_1^d + w_{26}x_2^d)$$

# Non Linear Classification: Computer Vision

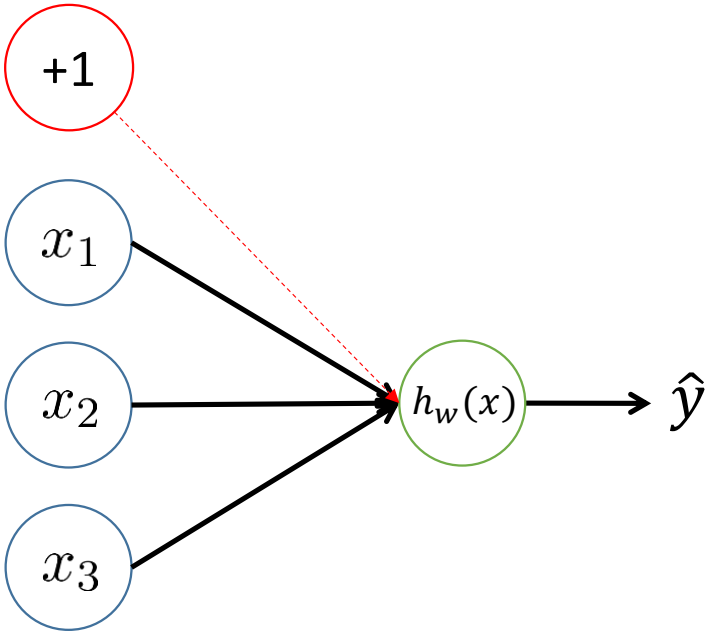
A non linear model in a high dimensional space may be a solution, but…

… what if we have many many features?!

… like in computer vision

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 194 | 210 | 201 | 212 | 199 | 213 | 215 | 195 | 178 | 158 | 182 | 209 |
| 180 | 189 | 190 | 221 | 209 | 205 | 191 | 167 | 147 | 115 | 129 | 163 |
| 114 | 126 | 140 | 188 | 176 | 165 | 152 | 140 | 170 | 106 | 78 | 88 |
| 87 | 103 | 115 | 154 | 143 | 142 | 149 | 153 | 173 | 101 | 57 | 57 |
| 102 | 112 | 106 | 131 | 122 | 138 | 152 | 147 | 128 | 84 | 58 | 66 |
| 94 | 95 | 79 | 104 | 105 | 124 | 129 | 113 | 107 | 87 | 69 | 67 |
| 68 | 71 | 69 | 98 | 89 | 92 | 98 | 95 | 89 | 88 | 76 | 67 |
| 41 | 56 | 68 | 99 | 63 | 45 | 60 | 82 | 58 | 76 | 75 | 65 |
| 20 | 43 | 69 | 75 | 56 | 41 | 51 | 73 | 55 | 70 | 63 | 44 |
| 50 | 50 | 57 | 69 | 75 | 75 | 73 | 74 | 53 | 68 | 59 | 37 |
| 72 | 59 | 53 | 66 | 84 | 92 | 84 | 74 | 57 | 72 | 63 | 42 |
| 67 | 61 | 58 | 65 | 75 | 78 | 76 | 73 | 59 | 75 | 69 | 50 |

# ANN Model Representation

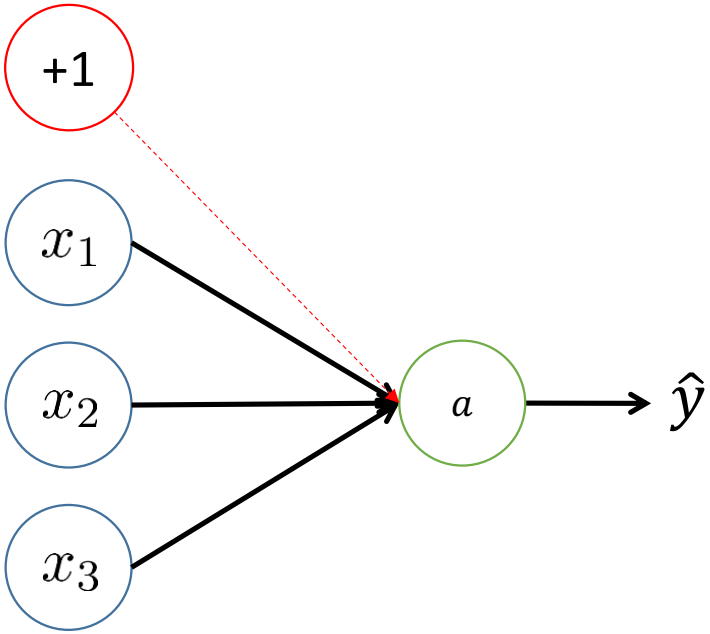



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

=1

$$\hat{y} = h_w(x) = g(w^T x)$$

# ANN Model Representation




=1

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

$$\hat{y} = h_w(x) = g(w^T x)$$

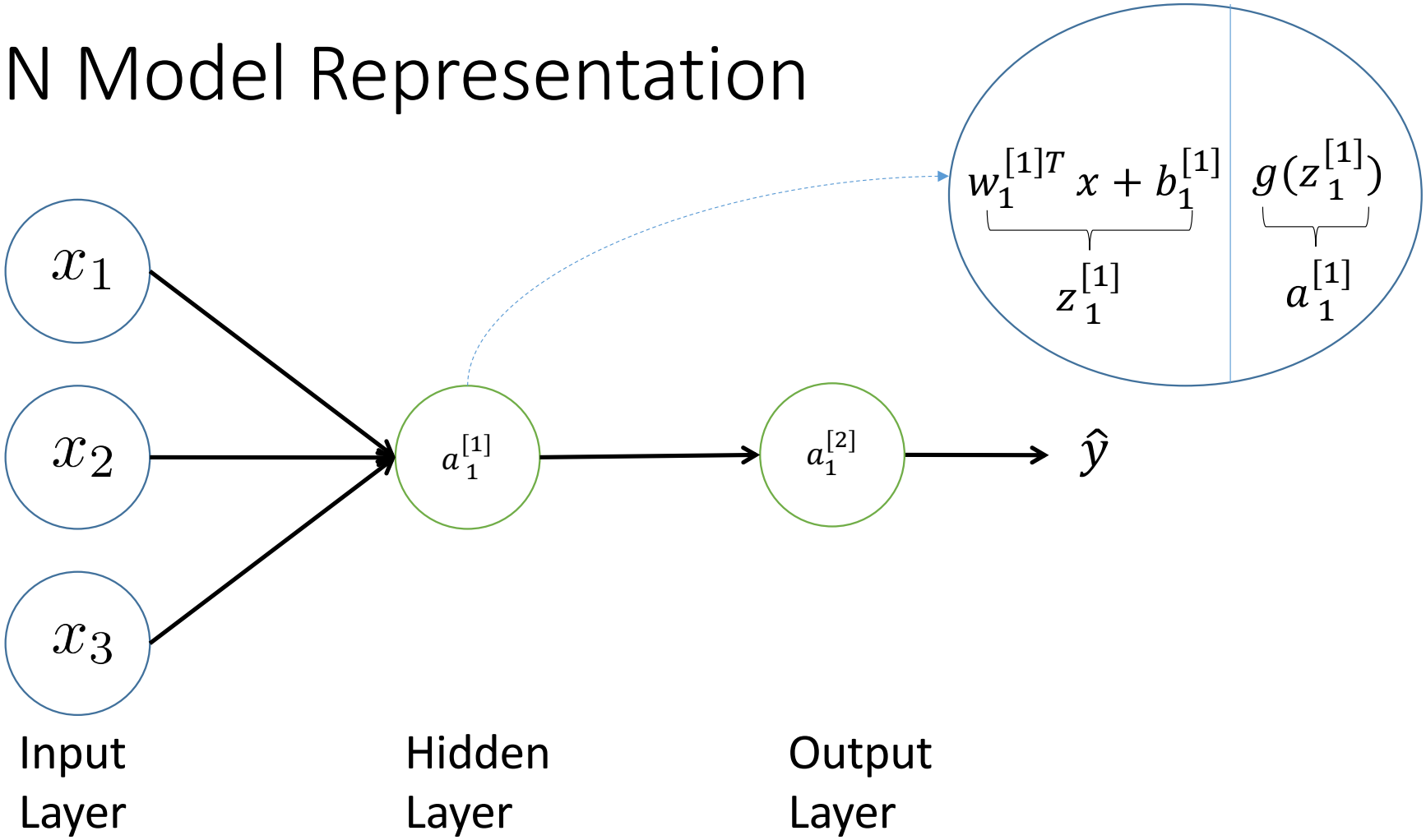
Change notation

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

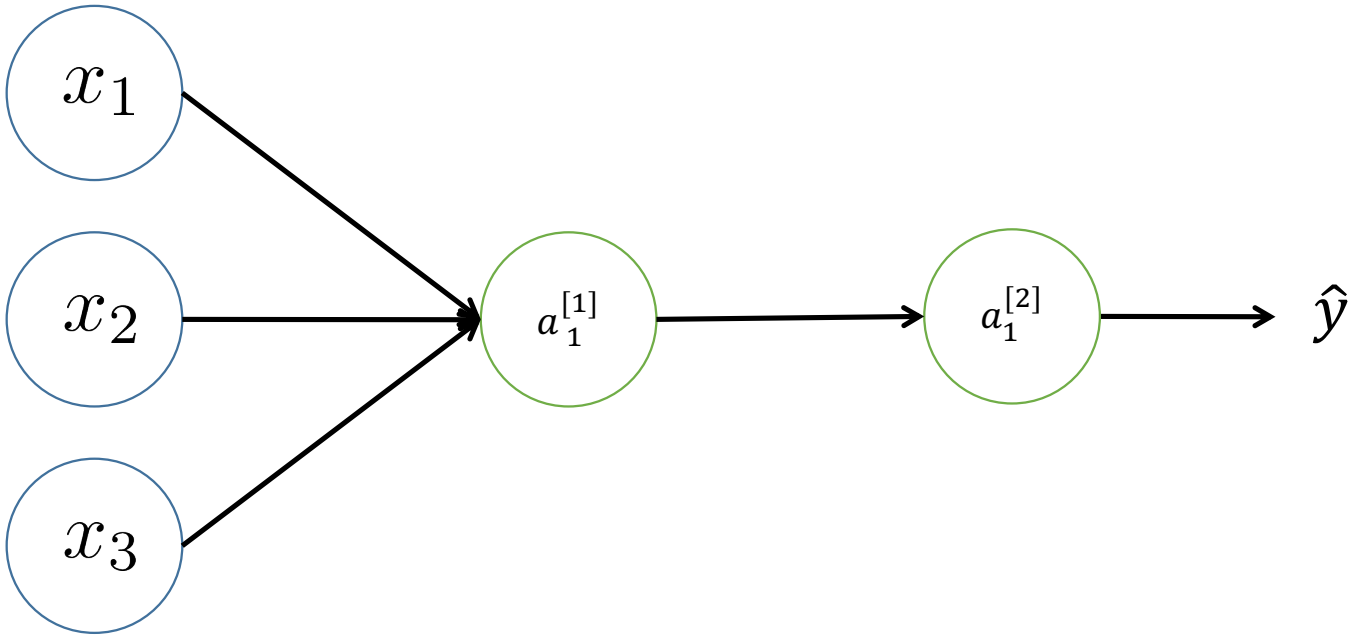
$= w_0$

$$z = w^T x + b,$$
$$\hat{y} = a = g(z)$$

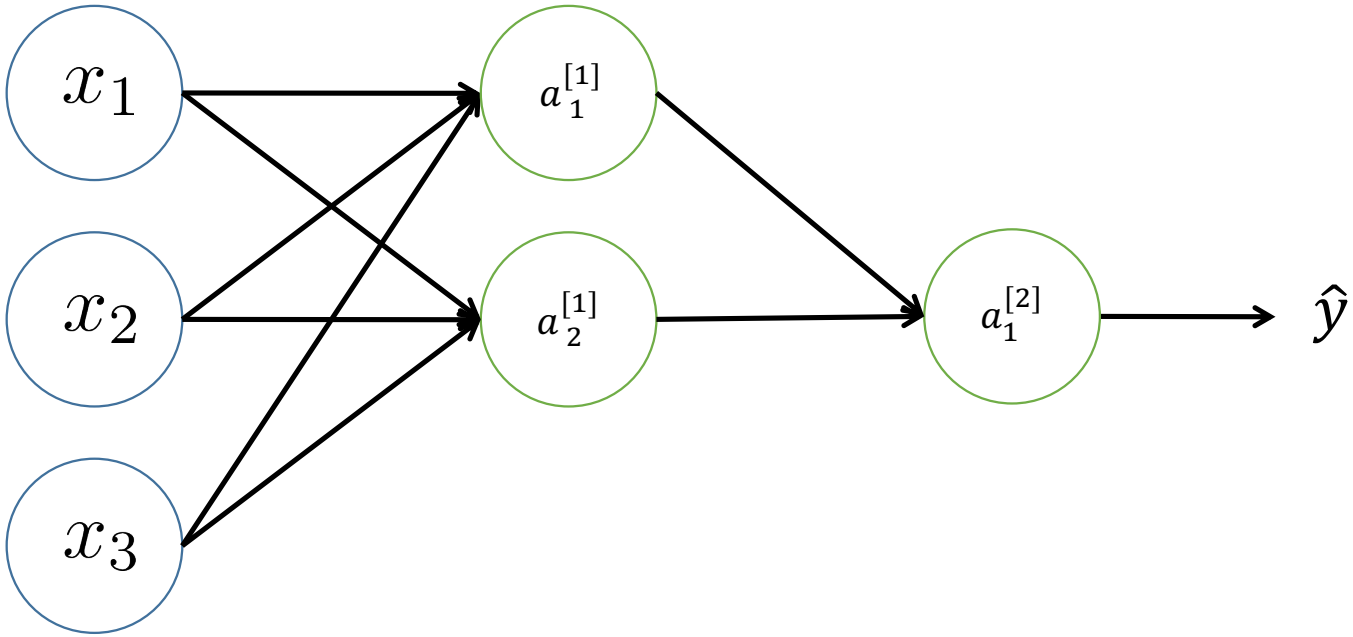
# ANN Model Representation



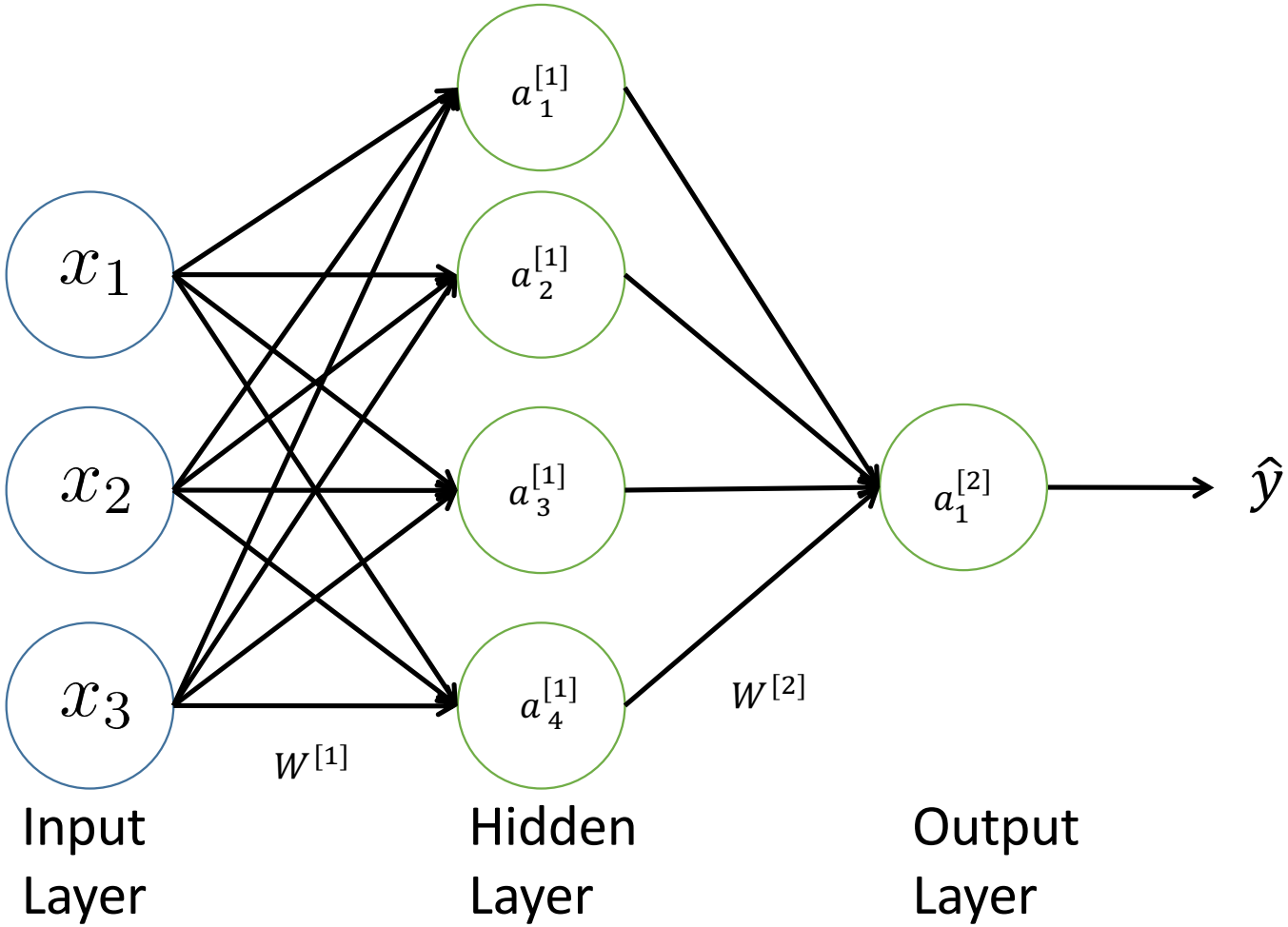
# ANN Model Representation



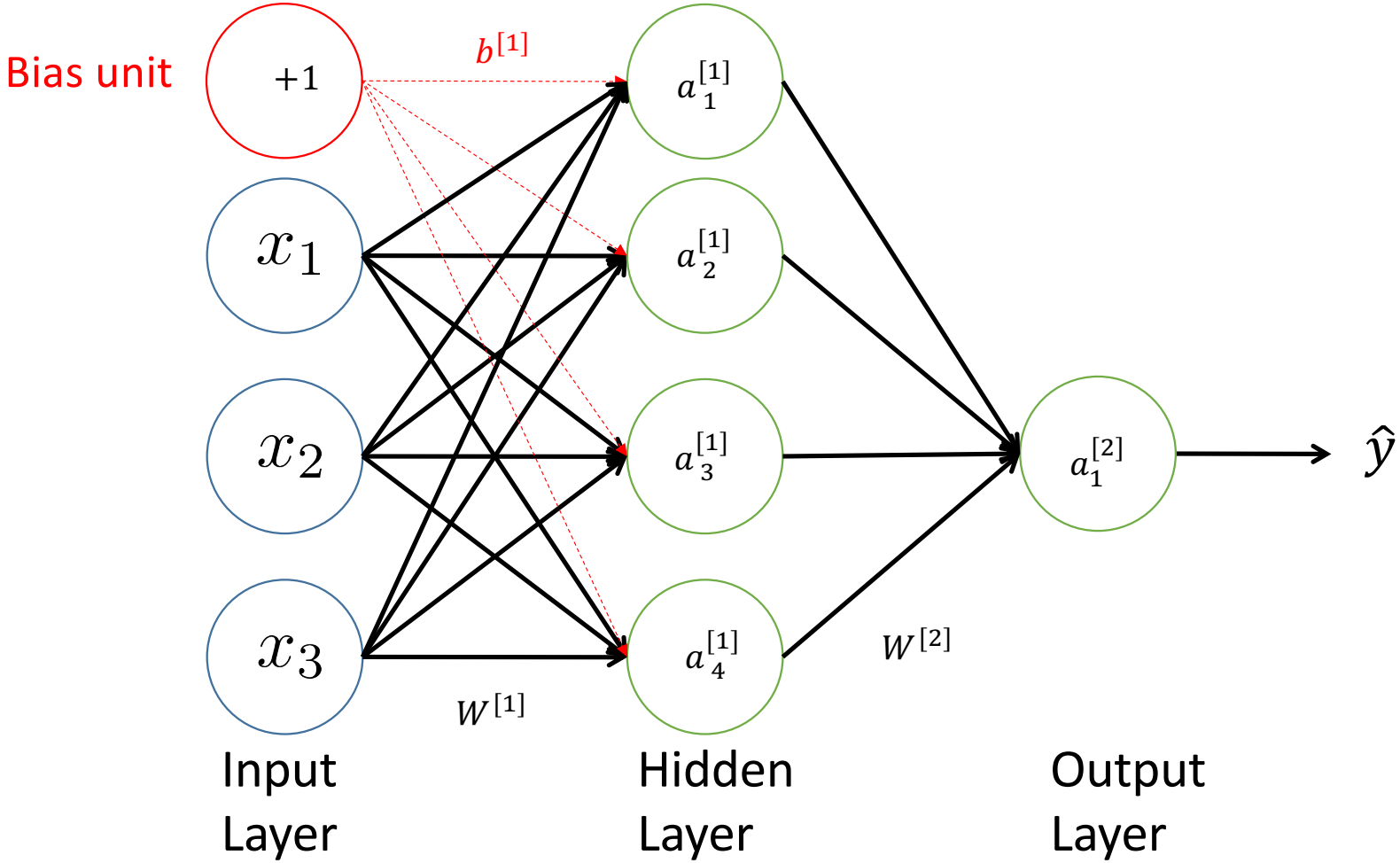
# ANN Model Representation



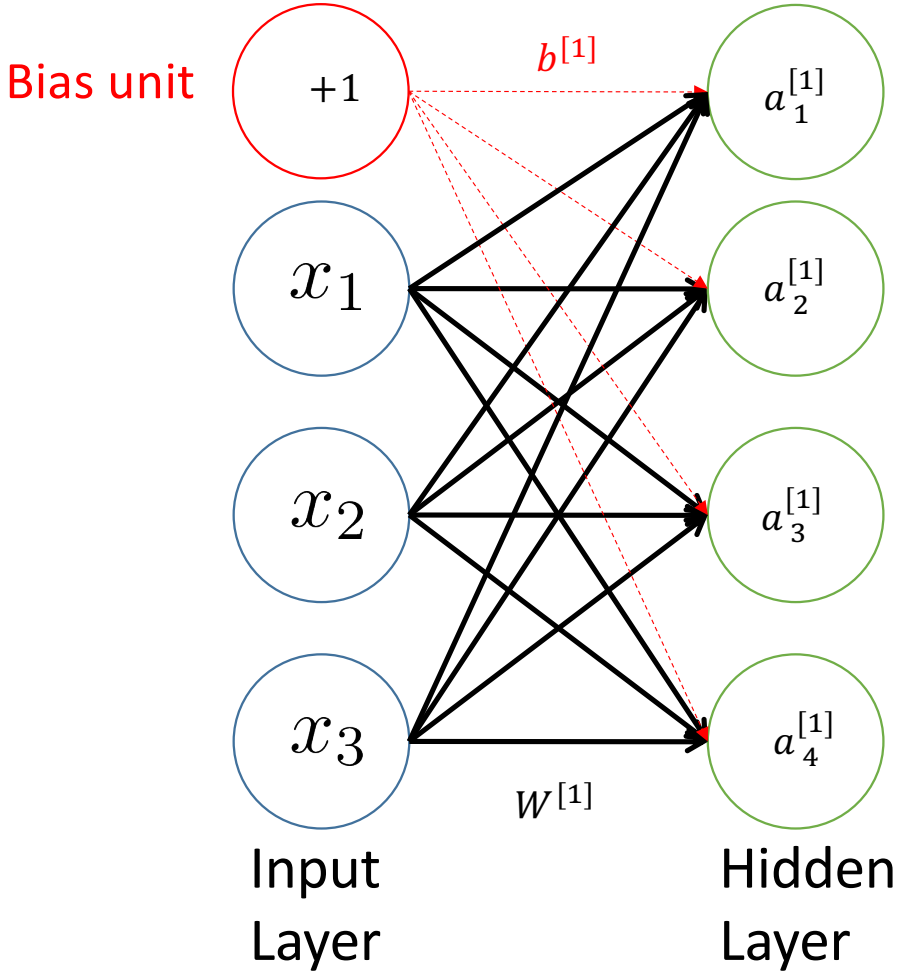
# ANN Model Representation



# ANN Model Representation

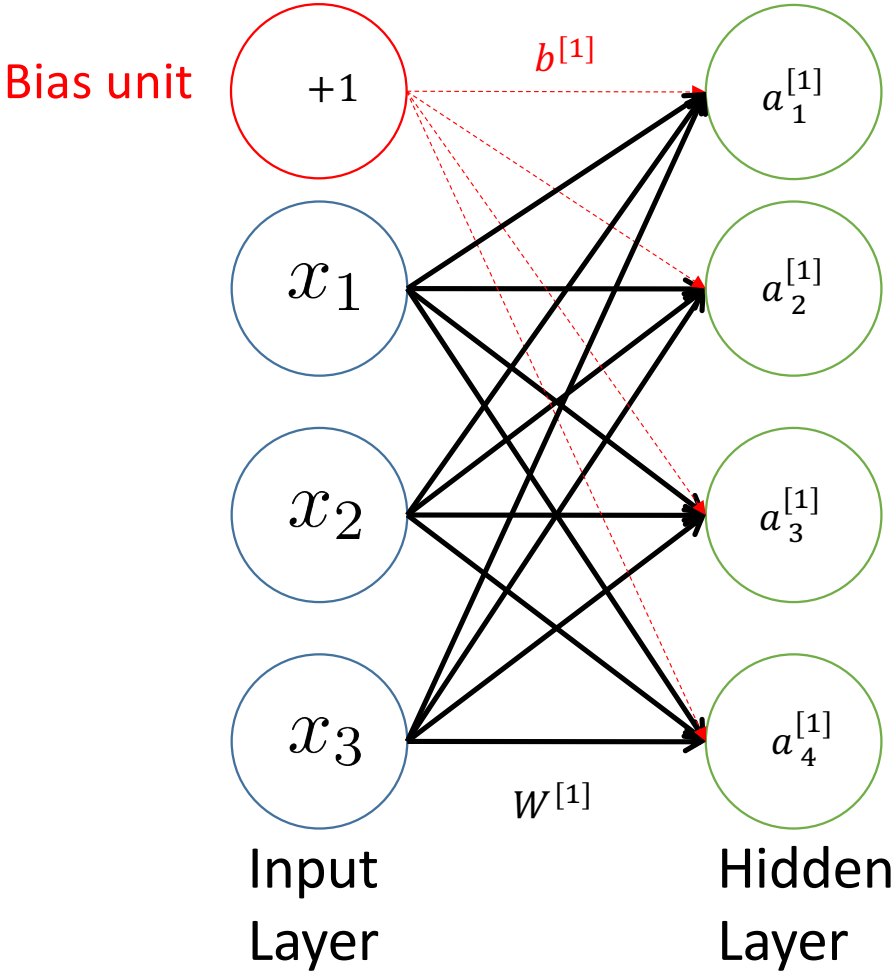


# ANN Model Representation




$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = g(z_1^{[1]})$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]},\ a_2^{[1]} = g(z_2^{[1]})$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]},\ a_3^{[1]} = g(z_3^{[1]})$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]},\ a_4^{[1]} = g(z_4^{[1]})$$

# ANN Model Representation

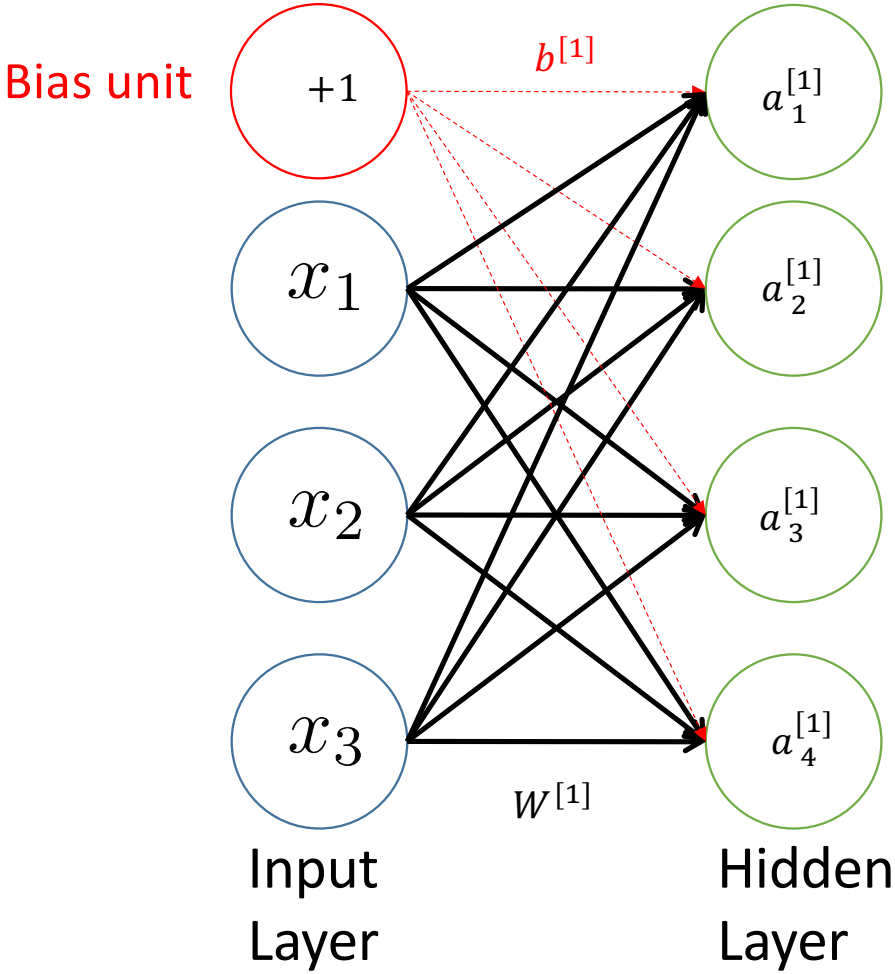



$$\begin{matrix} z^{[1]} & & W^{[1]} & & b^{[1]} & a^{[1]} \end{matrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} x + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}, \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = g(z_1^{[1]}) \\ = g(z_2^{[1]}) \\ = g(z_3^{[1]}) \\ = g(z_4^{[1]})$$

$$W^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix}$$

# ANN Model Representation

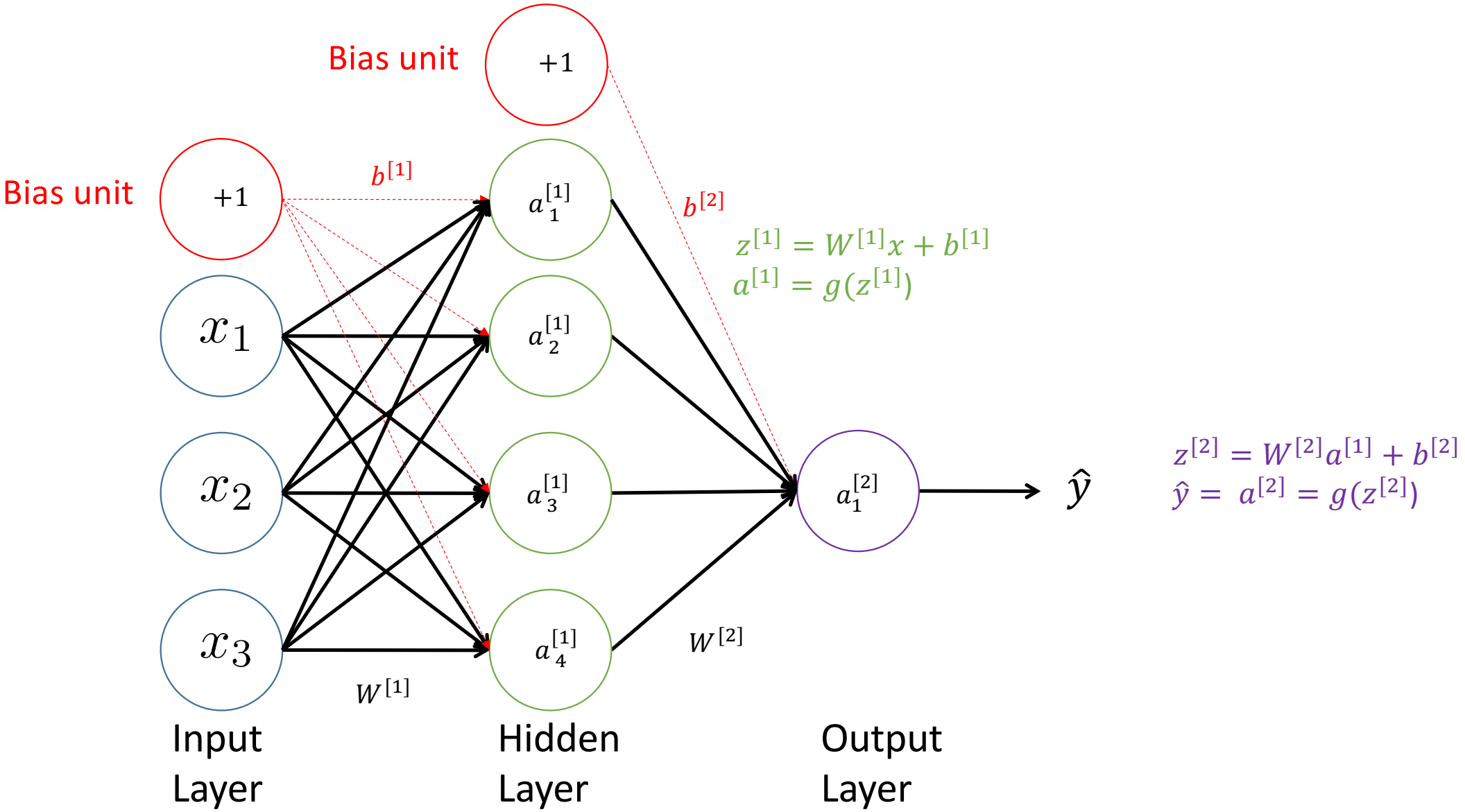



$$\begin{matrix} z^{[1]} & & W^{[1]} & & b^{[1]} & a^{[1]} \end{matrix}$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} x + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \begin{matrix} g(z_1^{[1]}) \\ g(z_2^{[1]}) \\ g(z_3^{[1]}) \\ g(z_4^{[1]}) \end{matrix}$$

$$W^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} & w_{33}^{[1]} \\ w_{41}^{[1]} & w_{42}^{[1]} & w_{43}^{[1]} \end{bmatrix}$$

$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \mathrm{g}(z^{[1]})$$


Bias unit
+1
Bias unit
+1
$b^{[1]}$
$b^{[2]}$
$a_1^{[1]}$
$a_2^{[1]}$
$a_3^{[1]}$
$a_4^{[1]}$
$x_1$
$x_2$
$x_3$
$a_1^{[2]}$
$\hat{y}$
$z^{[1]} = W^{[1]}x + b^{[1]}$
$a^{[1]} = g(z^{[1]})$
$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$
$\hat{y} = a^{[2]} = g(z^{[2]})$
$W^{[1]}$
$W^{[2]}$
Input Layer
Hidden Layer
Output Layer

# ANN Model Representation: Vectorization

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

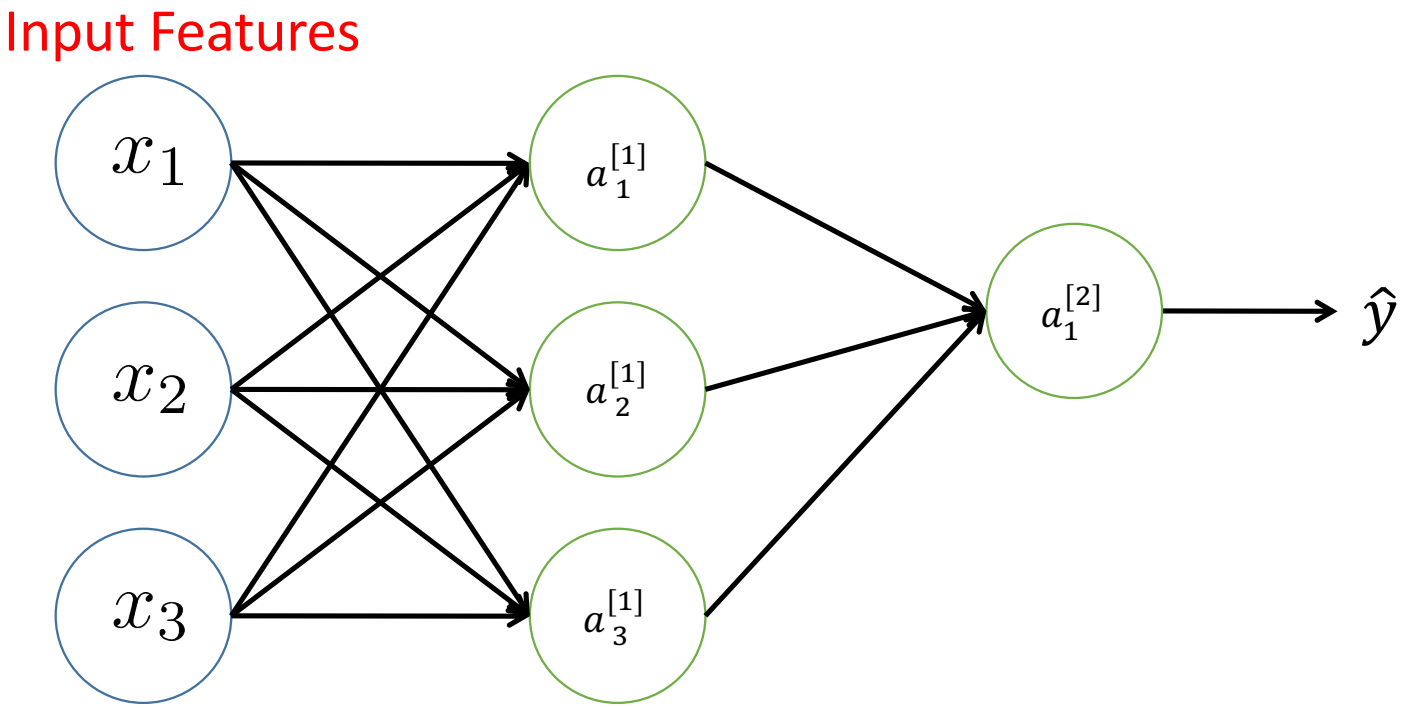
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$\hat{y} = a^{[2]} = g(z^{[2]})$$

For one example

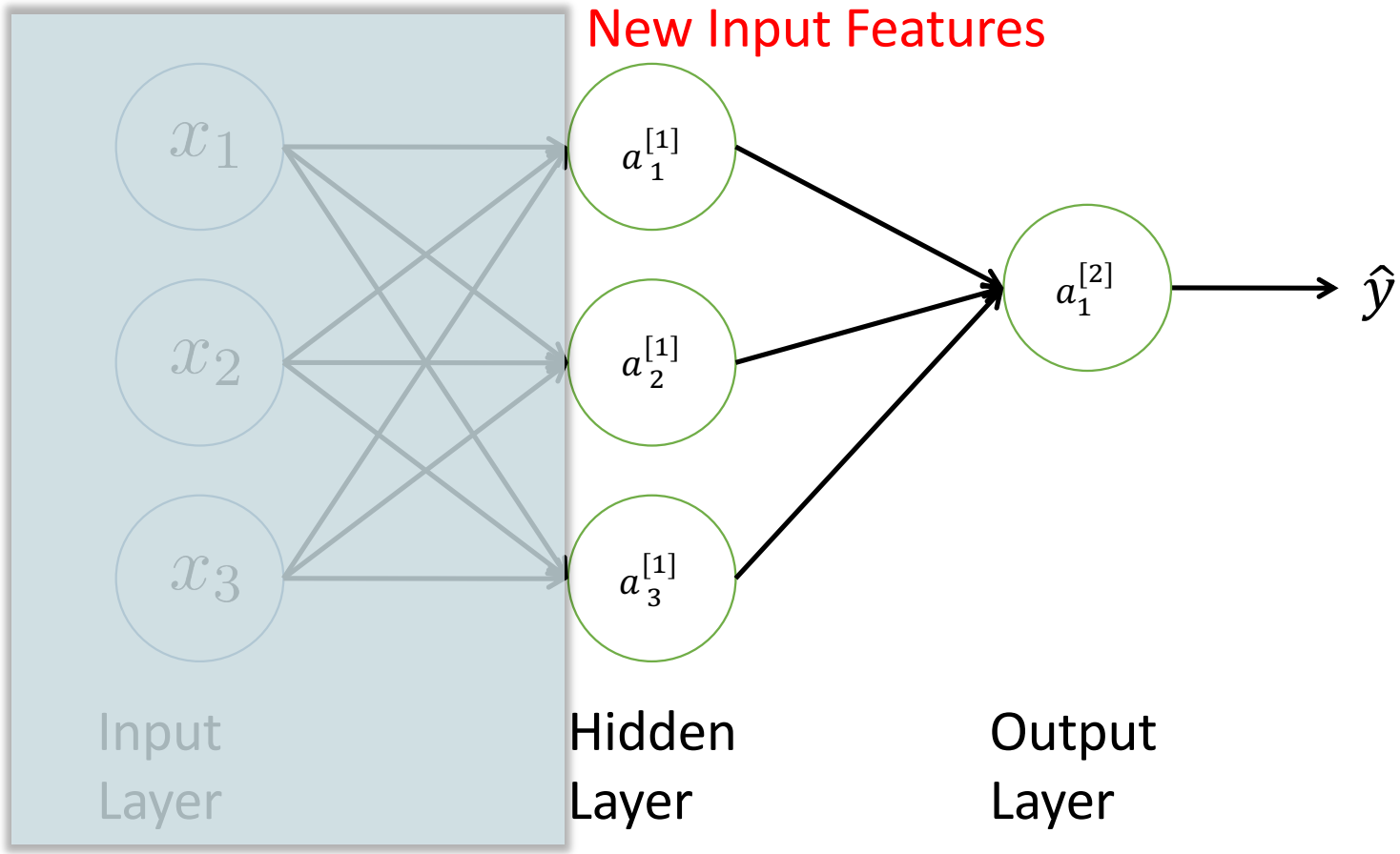
$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g(Z^{[2]})$$

For All Examples (matrix notation)

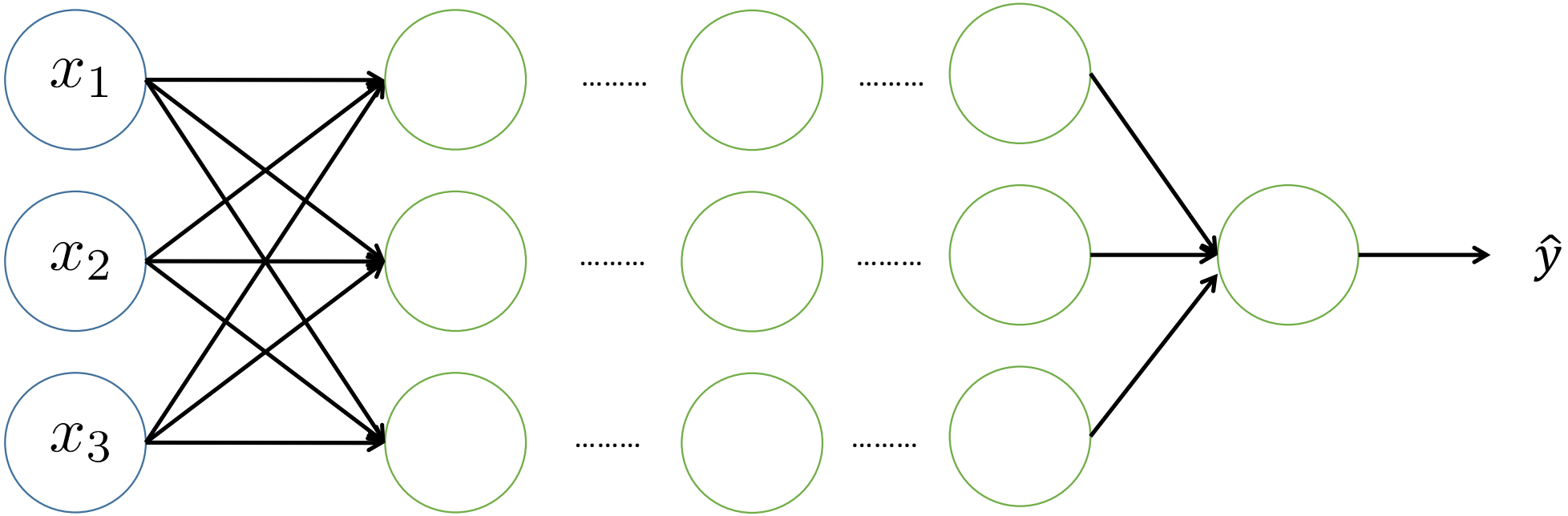
# ANN Representation: Learning its own features



# ANN Representation: Learning its own features



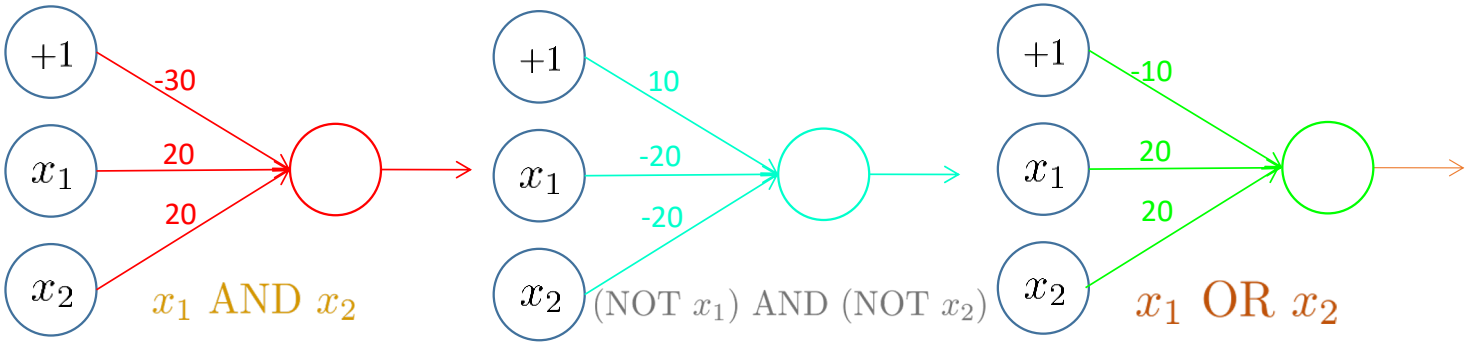
# ANN Representation: Deep NN

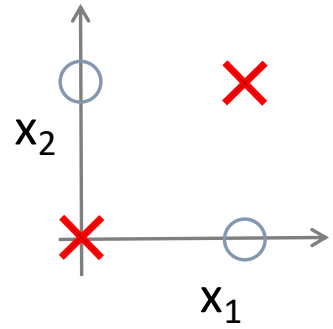


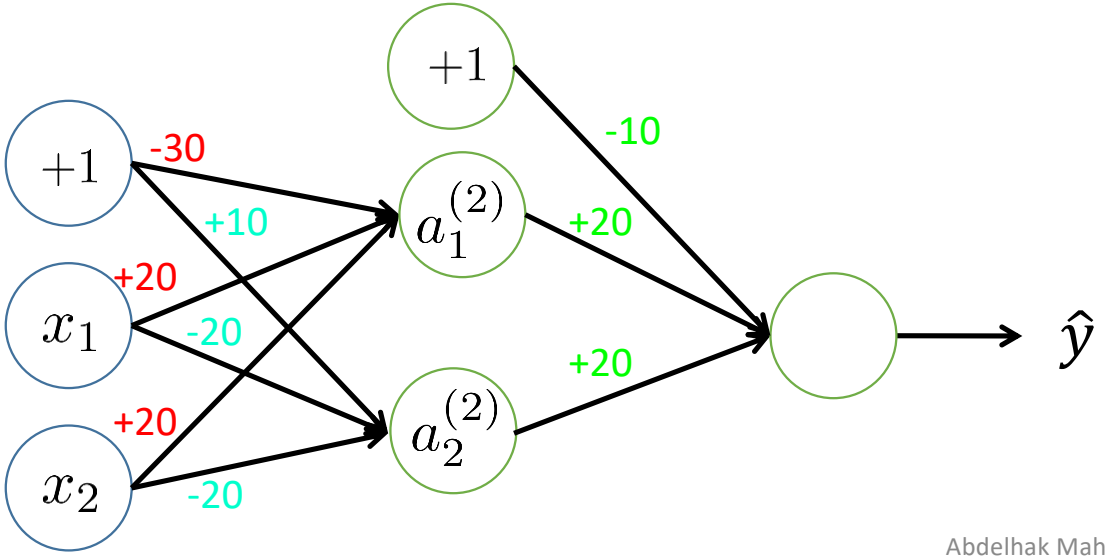
# Why ANN?

- Features Generation
  - Learn Features by it self
- Data non linearly separable
  - Learn complex non linear functions
- Deals with Unstructured data
  - Convolutional Neural Networks (Vision)
  - Recurrent Neural Networks (Sequence)
  - Generative Adversarial Networks (Generate data)

# Simple Example

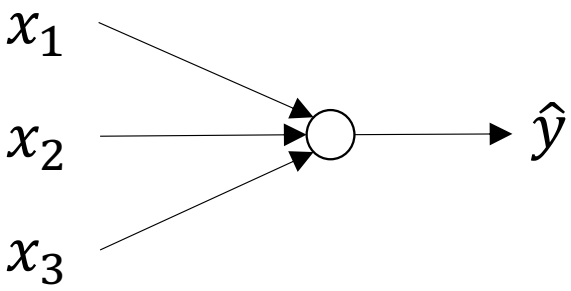


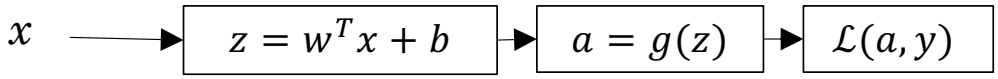




| $x_1$ | $x_2$ | $a_1^{(2)}$ | $a_2^{(2)}$ | $\hat{y}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

# ANN Learning




Remember Logistic Regression

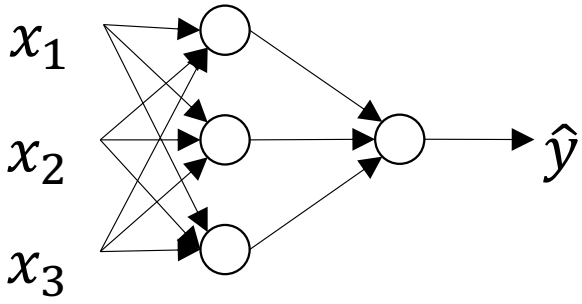


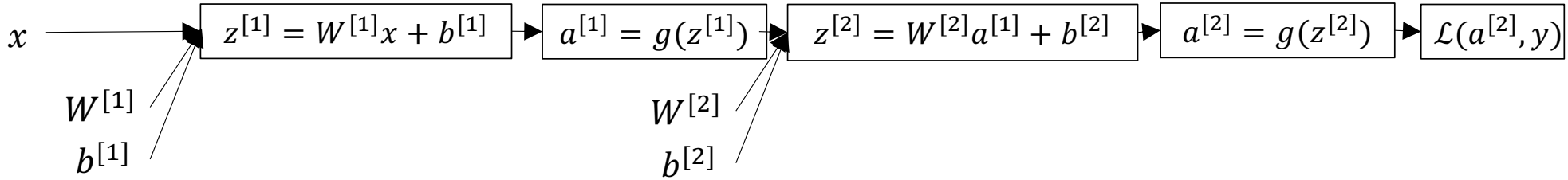

$dw$
$db$ $dz$ $da$

Notation: $dt = \frac{\partial \mathcal{L}}{\partial t}$

Cost Function $\mathcal{L}(a, y) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log (1 - a^{(i)})]$

# ANN Learning: Forward Propagation

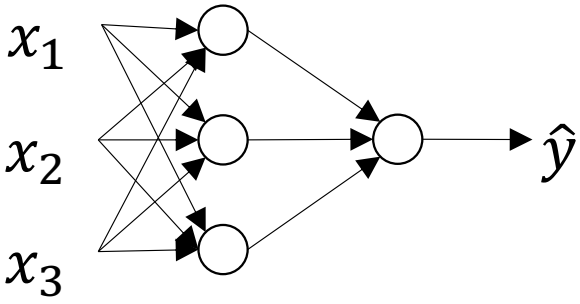



Cost Function

$$\mathcal{L}(a^{[2]}, y) = -\frac{1}{m}\sum_{i=0}^{m}\left(y^{(i)}\log\left(a^{[2](i)}\right) + (1-y^{(i)})\log\left(1-a^{[2](i)}\right)\right)$$

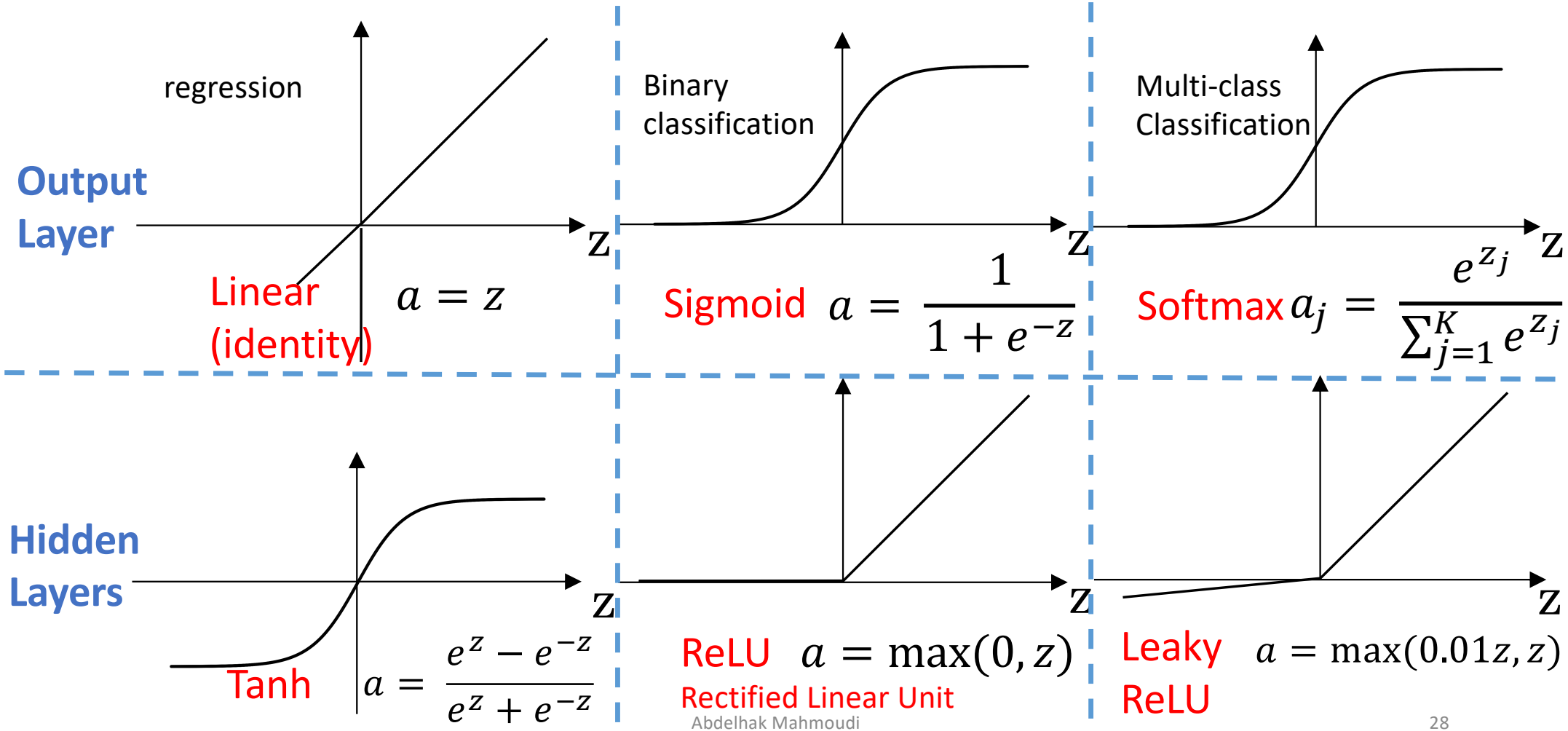
# ANN Learning: Backward Propagation



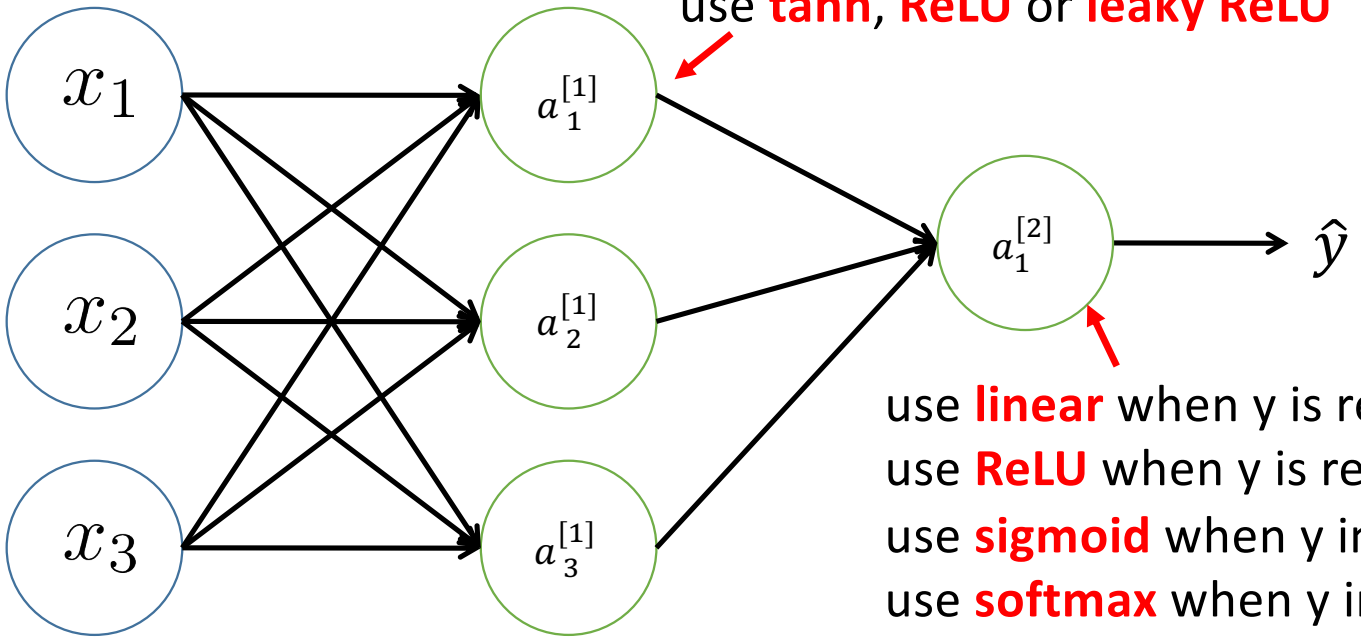

$$x \rightarrow z^{[1]} = W^{[1]}x + b^{[1]} \rightarrow a^{[1]} = g(z^{[1]}) \rightarrow z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \rightarrow a^{[2]} = g(z^{[2]}) \rightarrow \mathcal{L}(a^{[2]}, y)$$

$$dW^{[1]} \quad db^{[1]} \leftarrow dz^{[1]} \leftarrow da^{[1]} \quad dW^{[2]} \quad db^{[2]} \leftarrow dz^{[2]} \leftarrow da^{[2]}$$

# ANN Learning: Activation Functions



# ANN Learning : Activation Functions

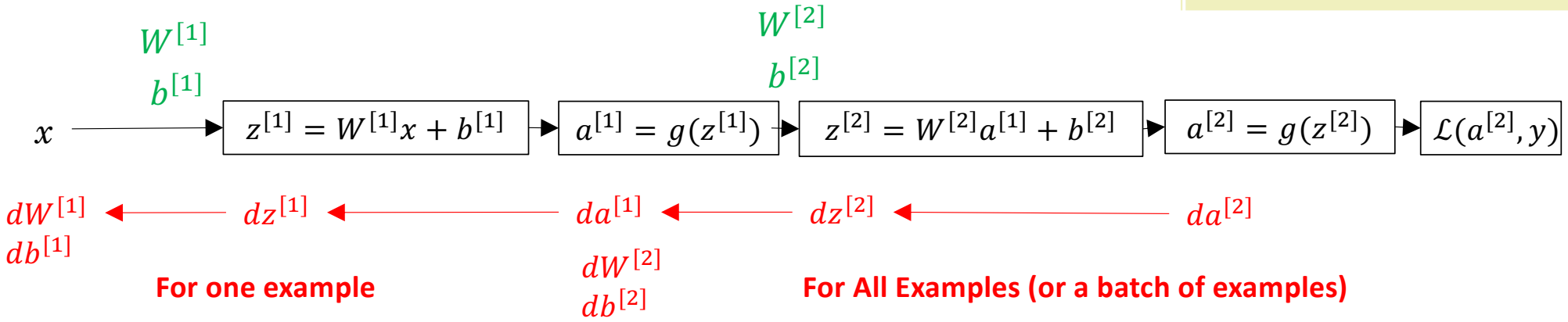



use **linear** when y is real (regression)
use **ReLU** when y is real positive (regression)
use **sigmoid** when y in {0,1} (2 classes)
use **softmax** when y in {0,1,2, ...} (K classes)

Input Layer | Hidden Layer | Output Layer

# ANN Learning : Backward Propagation

| $f \circ g$ | $(f' \circ g) \times g'$ |
|---|---|
| $f(g(x))$ | $f'(g(x))g'(x)$ |
| $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$ | |



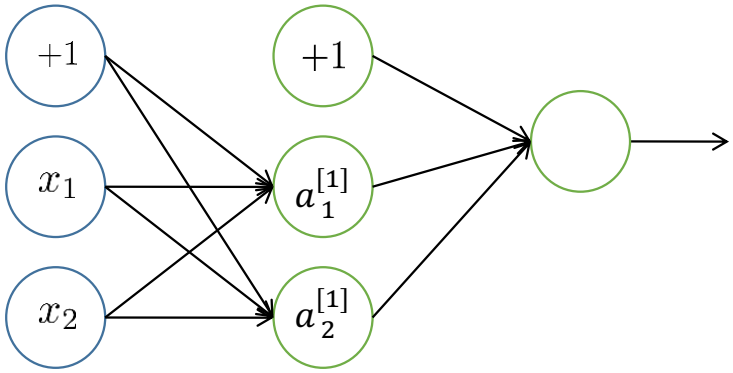

**For one example**

$$dz^{[2]} = a^{[2]} - y$$
$$dW^{[2]} = dz^{[2]}a^{[1]^T}$$
$$db^{[2]} = dz^{[2]}$$
$$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$$
$$dW^{[1]} = dz^{[1]}x^T$$
$$db^{[1]} = dz^{[1]}$$

**For All Examples (or a batch of examples)**

$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dZ^{[2]}A^{[1]^T}$$
$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$
$$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]}X^T$$
$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

# ANN Learning: Random Initialization

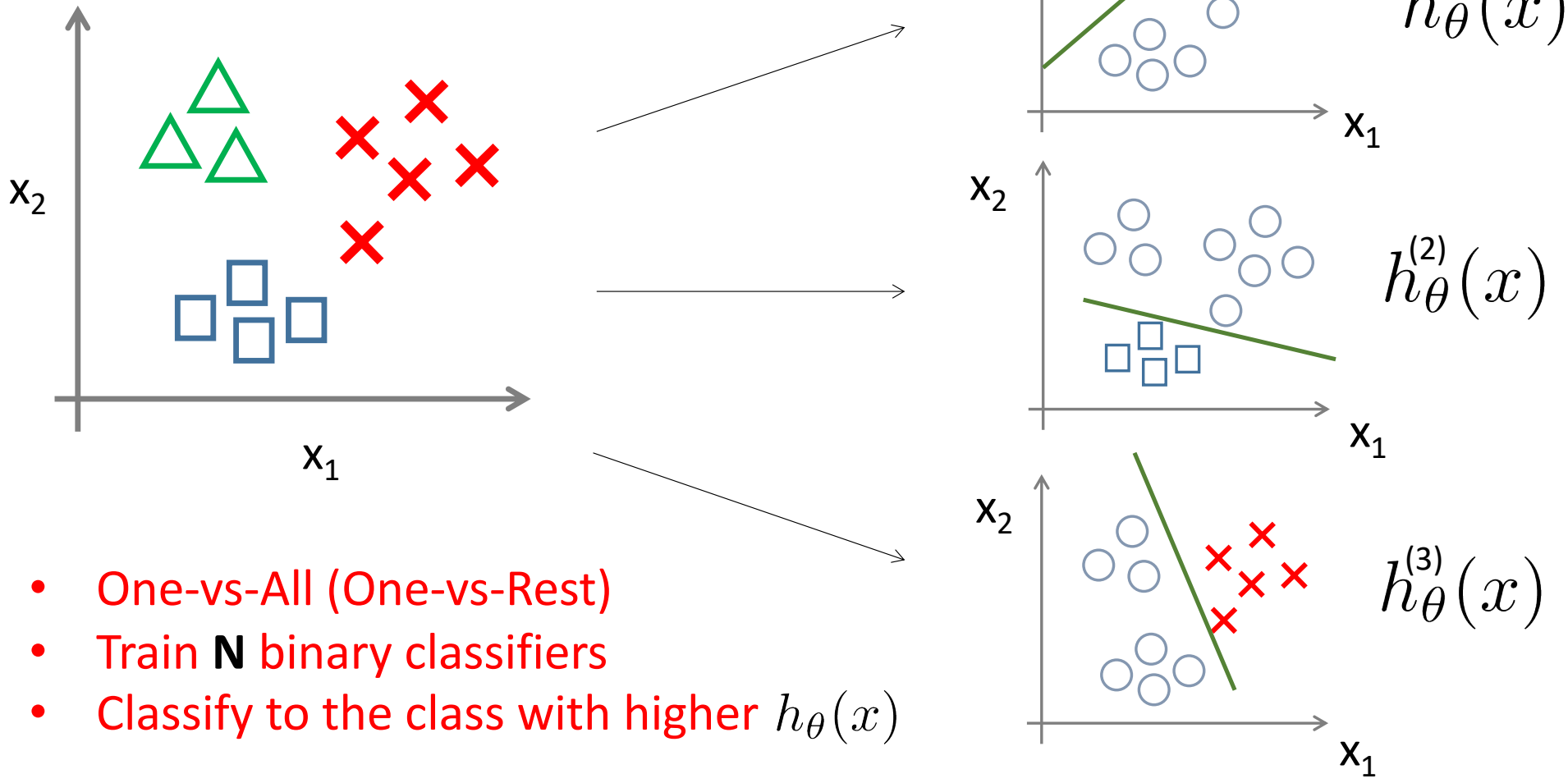



If $W^{[1]}$ and $b^{[1]}$ are initialized with zeros, after each update, parameters corresponding to inputs going into each of the two hidden units are identical, which results in $a_1^{[1]} = a_2^{[1]}$

# ANN Implementation

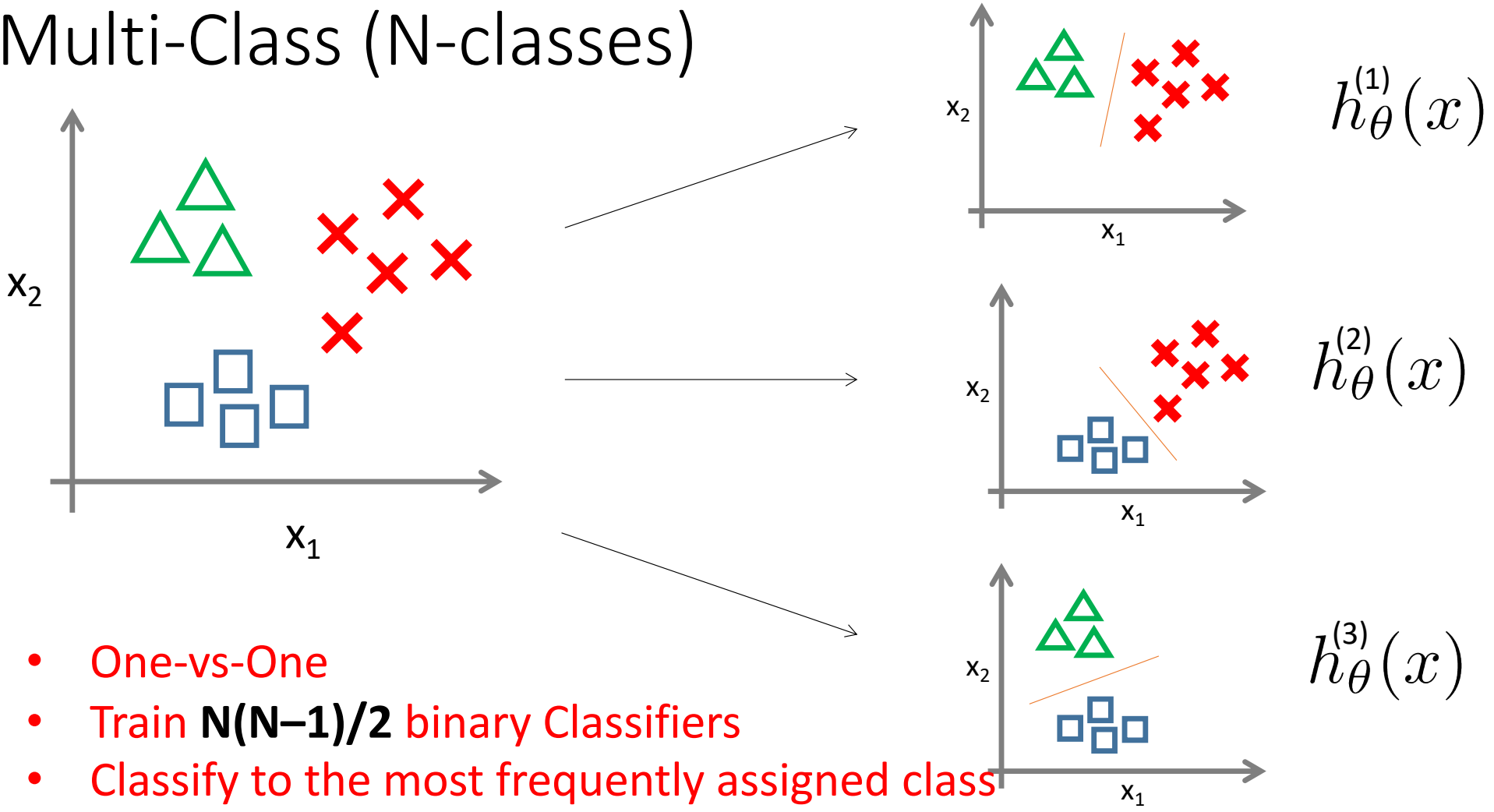
1. Pick a network architecture
   - No. of input units: Dimension of features
   - No. output units: Number of classes
   - No. hidden layers
   - No. hidden units: (have same no. in every layer, the more the better)
2. Randomly initialize weights
3. Repeat (chose a number of iterations)
   1. Implement forward propagation
   2. Compute cost function
   3. Implement backward propagation to compute partial derivatives
   4. Update the W and b: $W^{[l]} =: W^{[l]} - \alpha\, dW^{[l]}$ $b^{[l]} =: b^{[l]} - \alpha\, db^{[l]}$

# Multi-Class (N-classes)




- One-vs-All (One-vs-Rest)
- Train **N** binary classifiers
- Classify to the class with higher $h_\theta(x)$

# Multi-Class (N-classes)




- One-vs-One
- Train **N(N–1)/2** binary Classifiers
- Classify to the most frequently assigned class