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

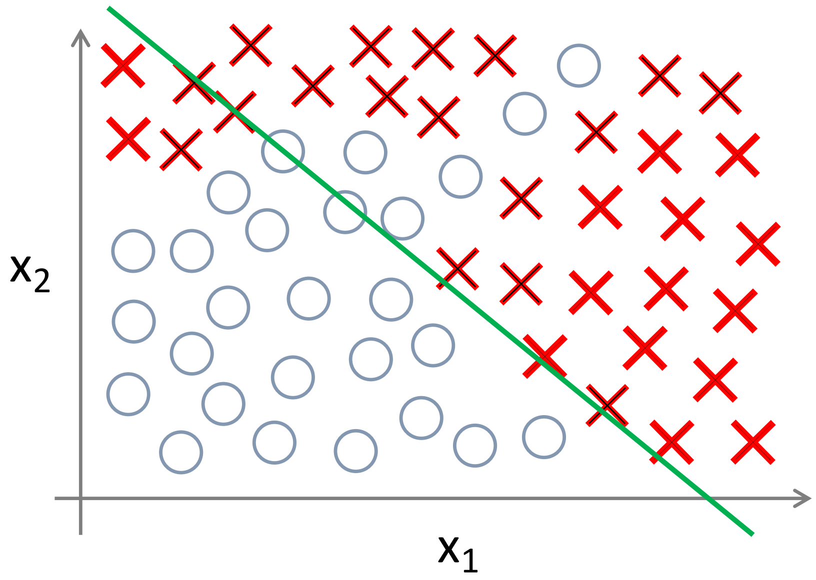
- Linear Regression
- Logistic Regression
- Support Vector Machines
- Trees (Decision and Regression)
- Random Forests
- Boosting
- **Artificial Neural Networks**

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 !

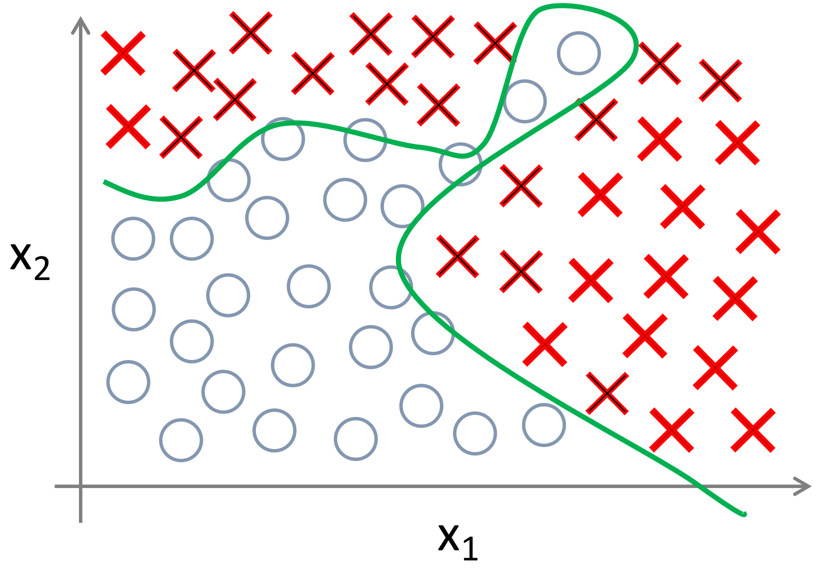


$$g(w_0 + w_1 x_1 + w_2 x_2)$$

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

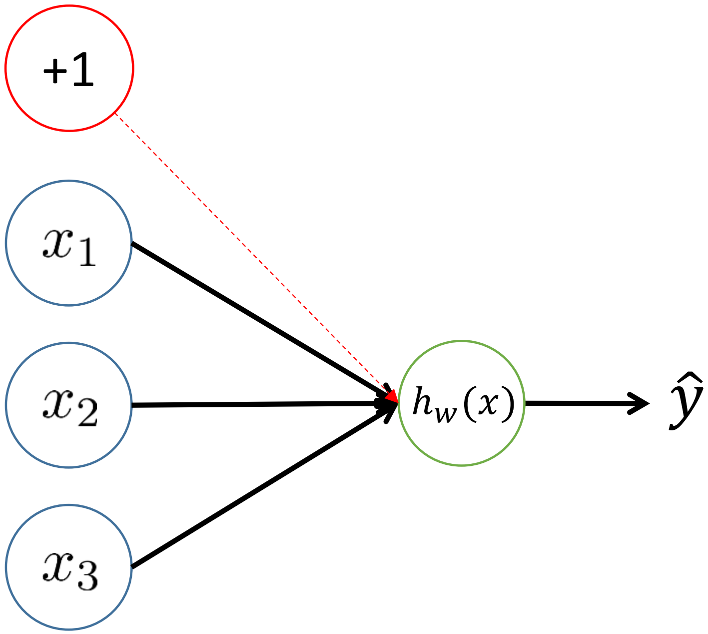
A non linear model in a high dimensional space may be a solution, but…

… 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

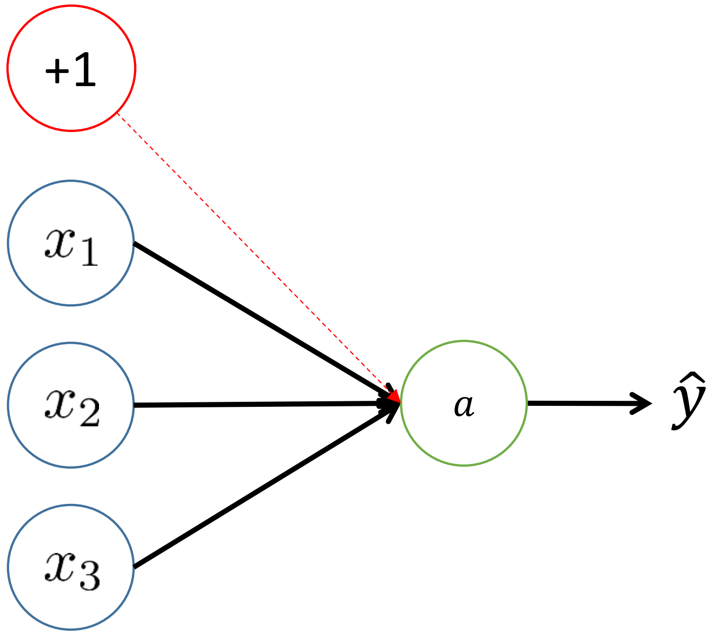
ANN Model Representation



$$x = \begin{bmatrix} \overset{=1}{x_0} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix}$$

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

ANN Model Representation



=1

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

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

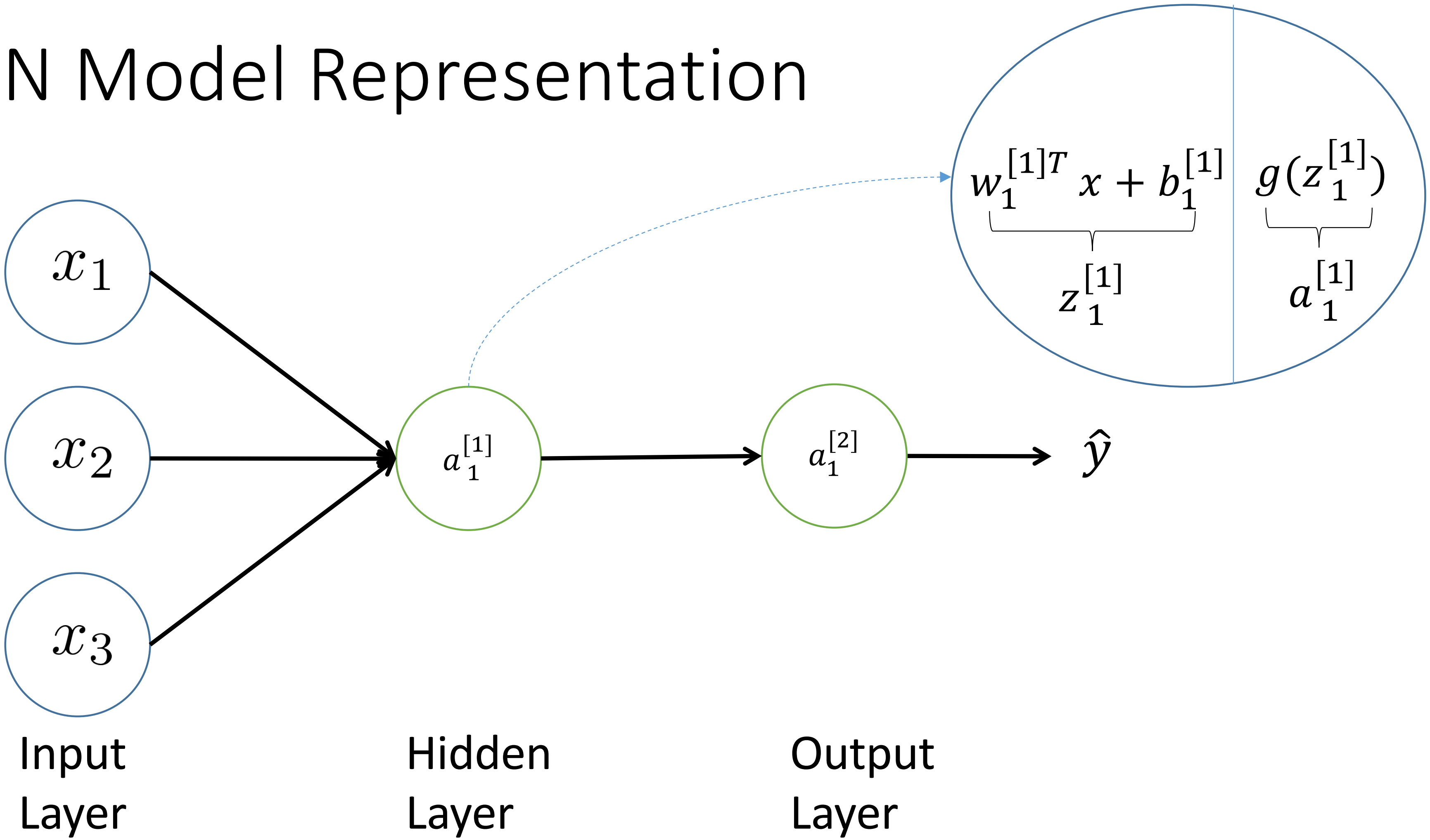
Change notation

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

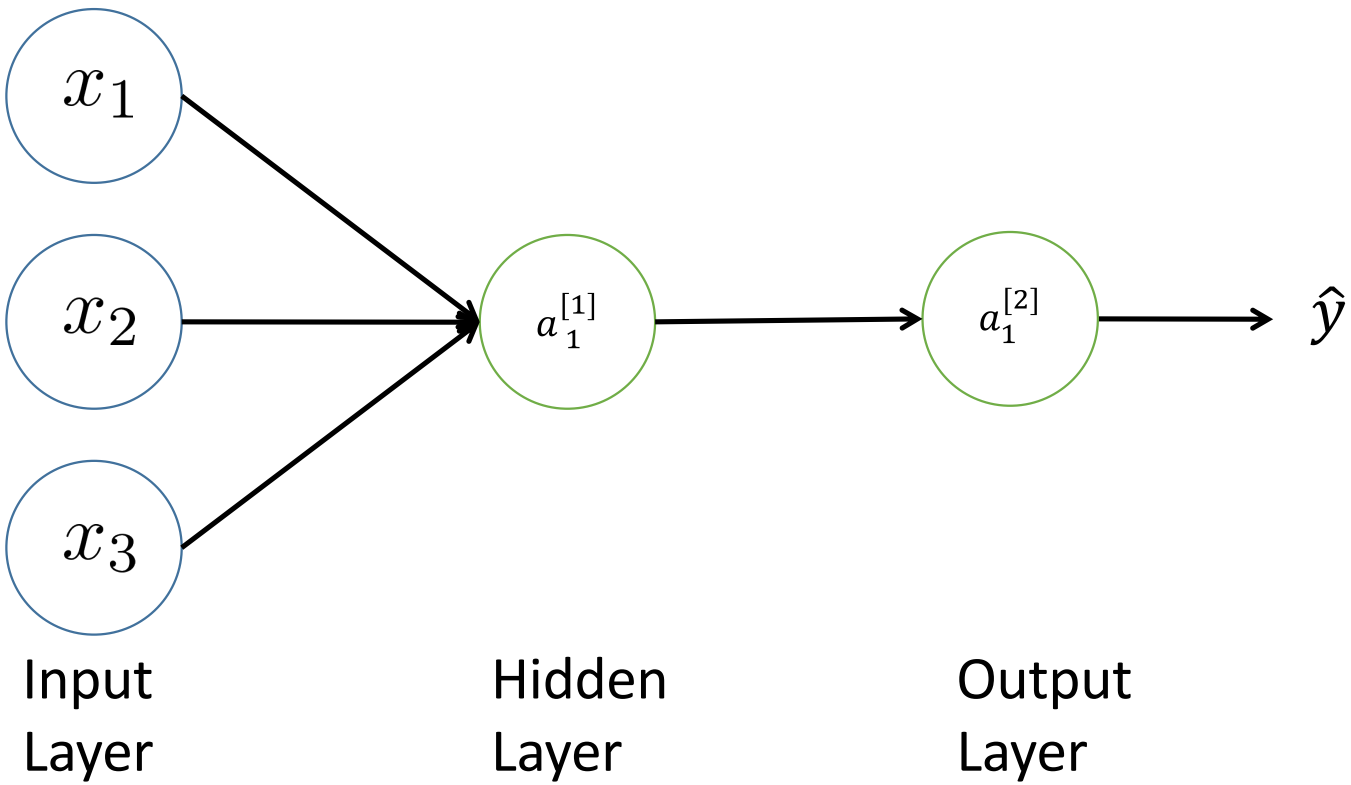
$$\boxed{= w_0}$$

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

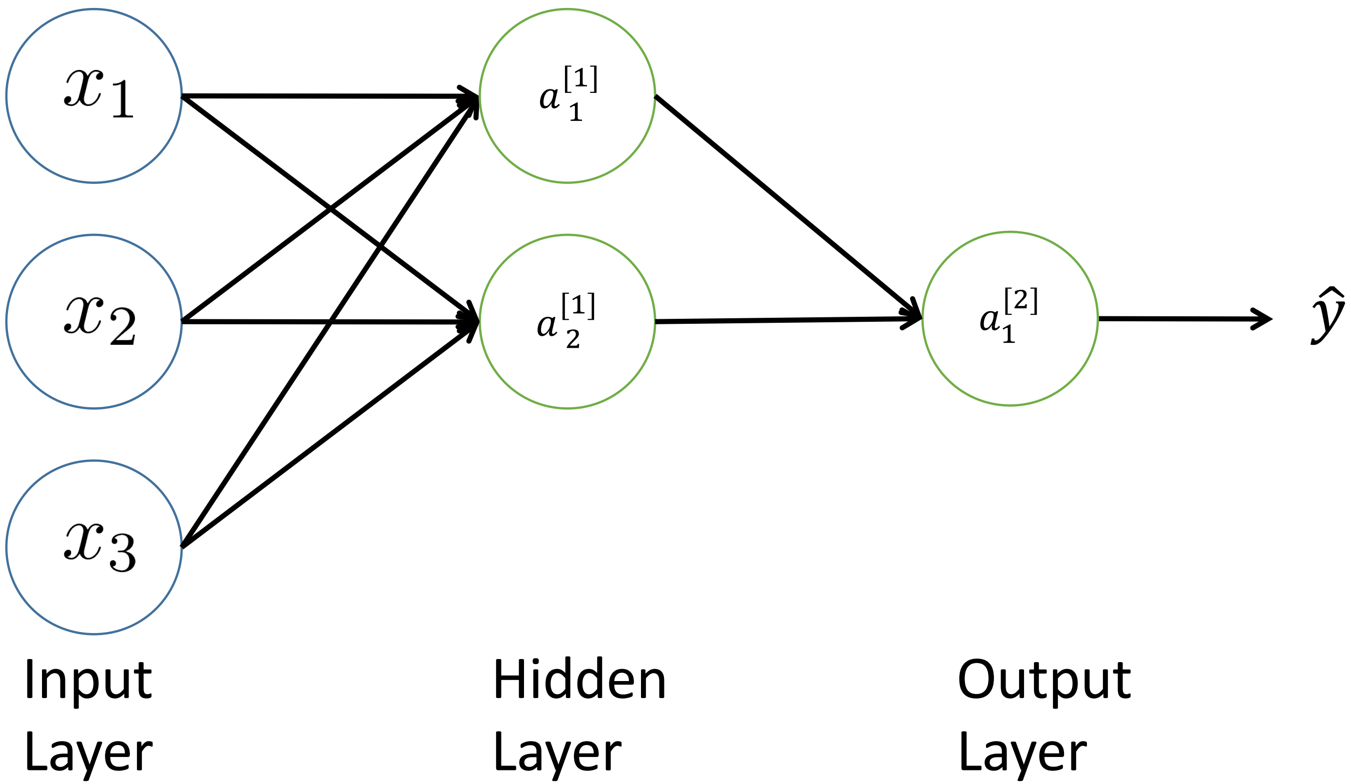
ANN Model Representation



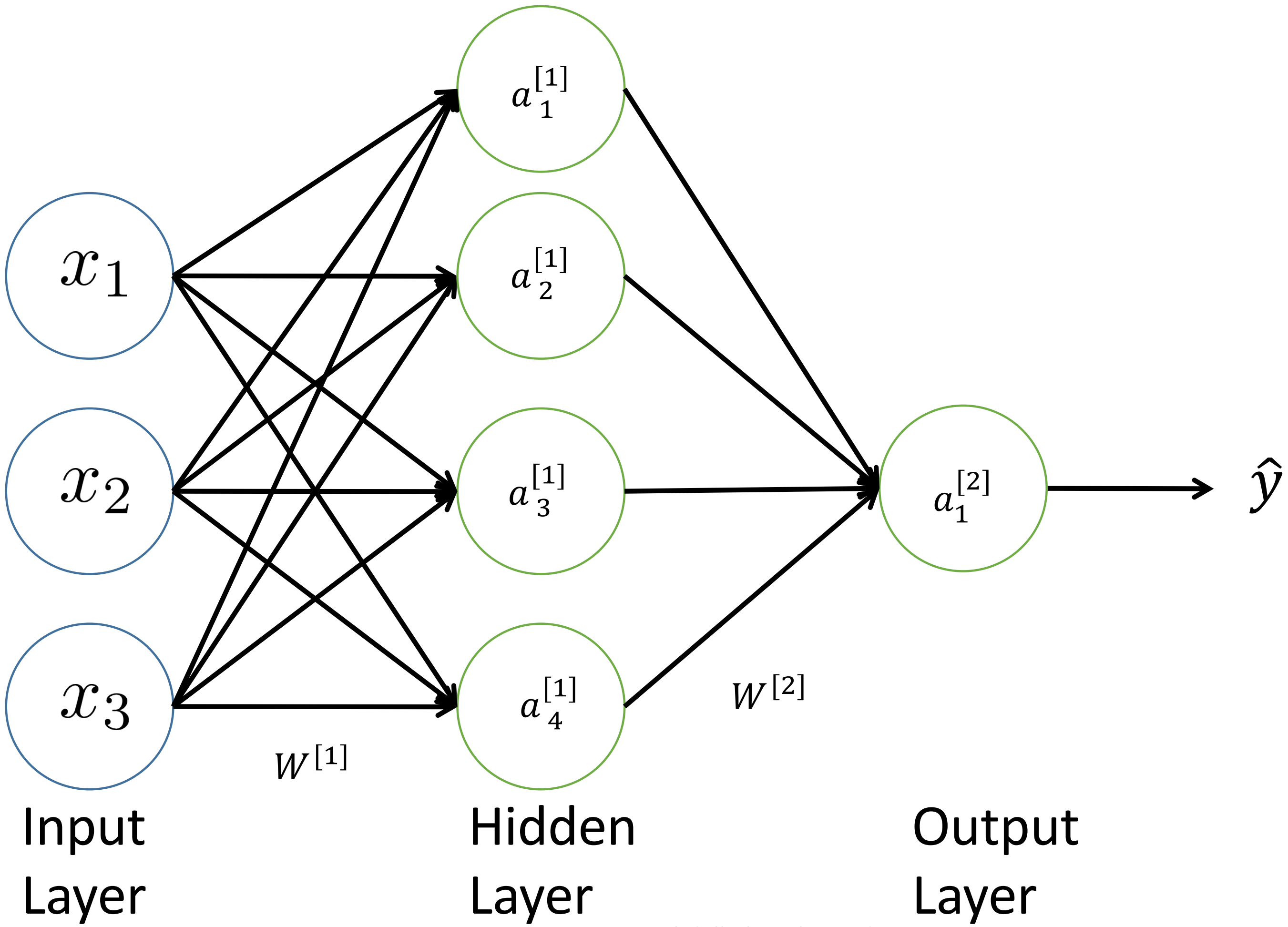
ANN Model Representation



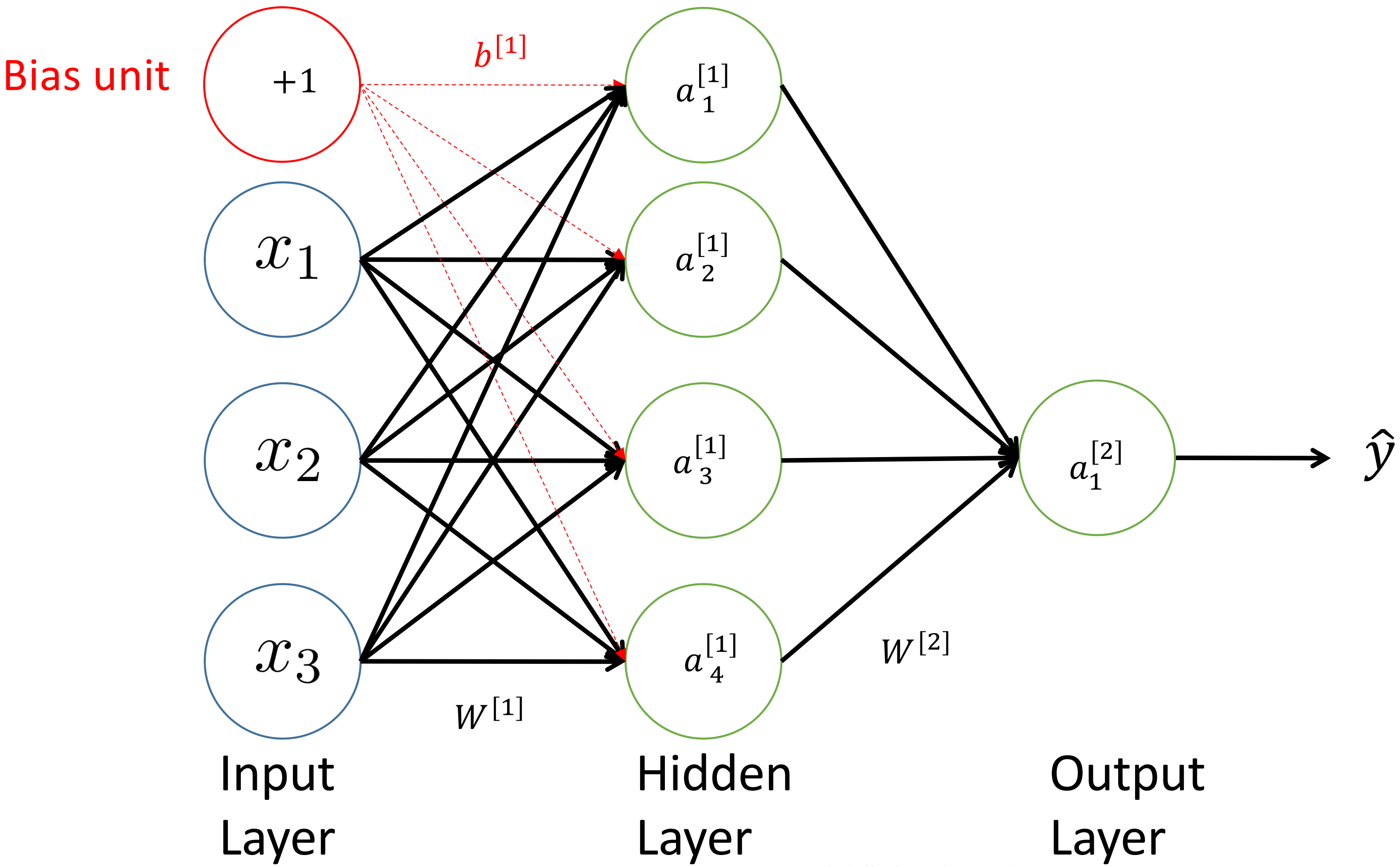
ANN Model Representation



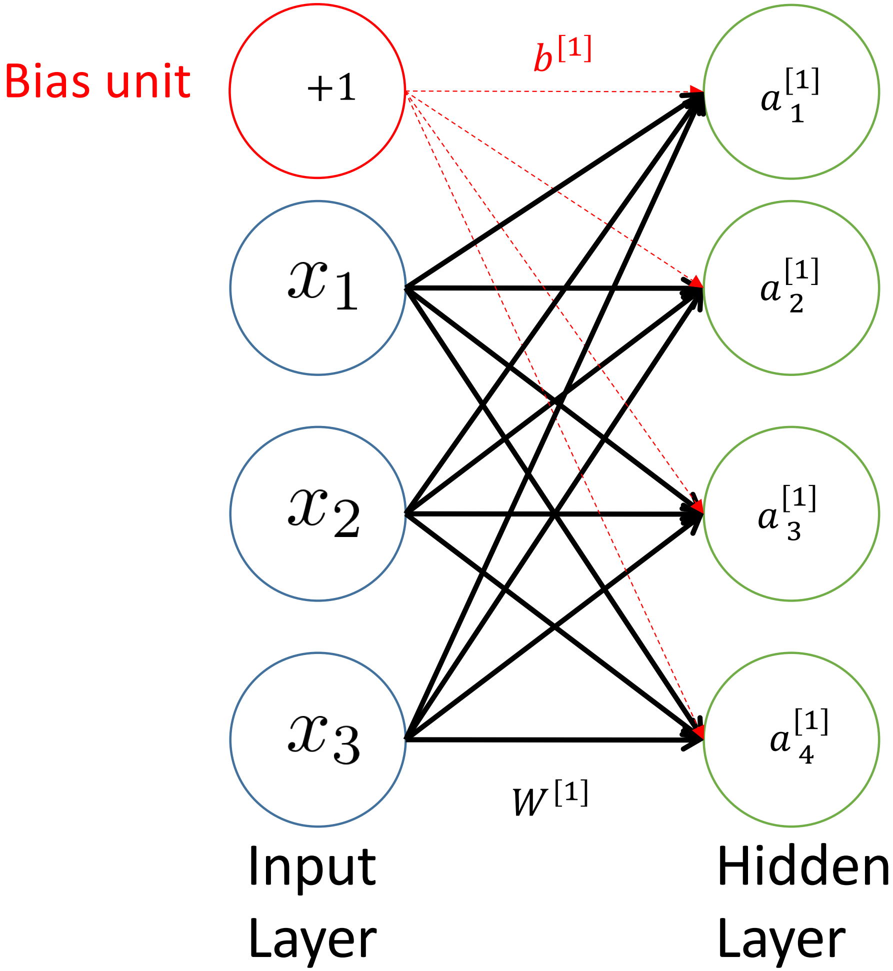
ANN Model Representation



ANN Model Representation

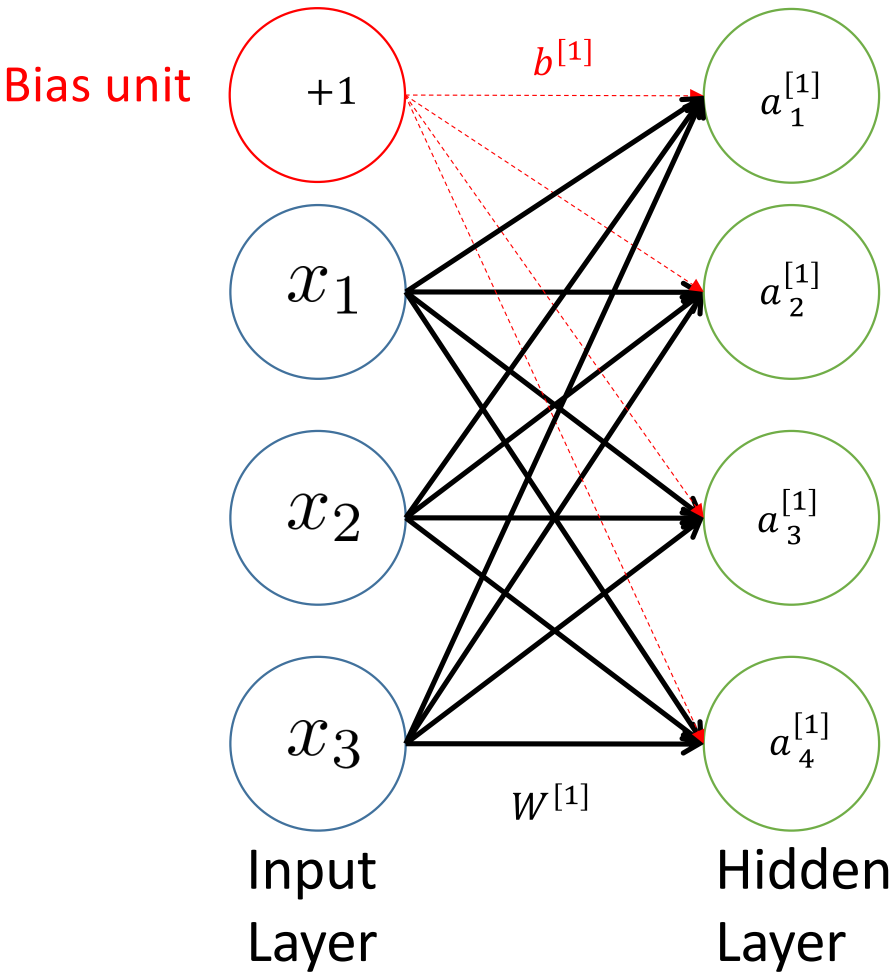


ANN Model Representation



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

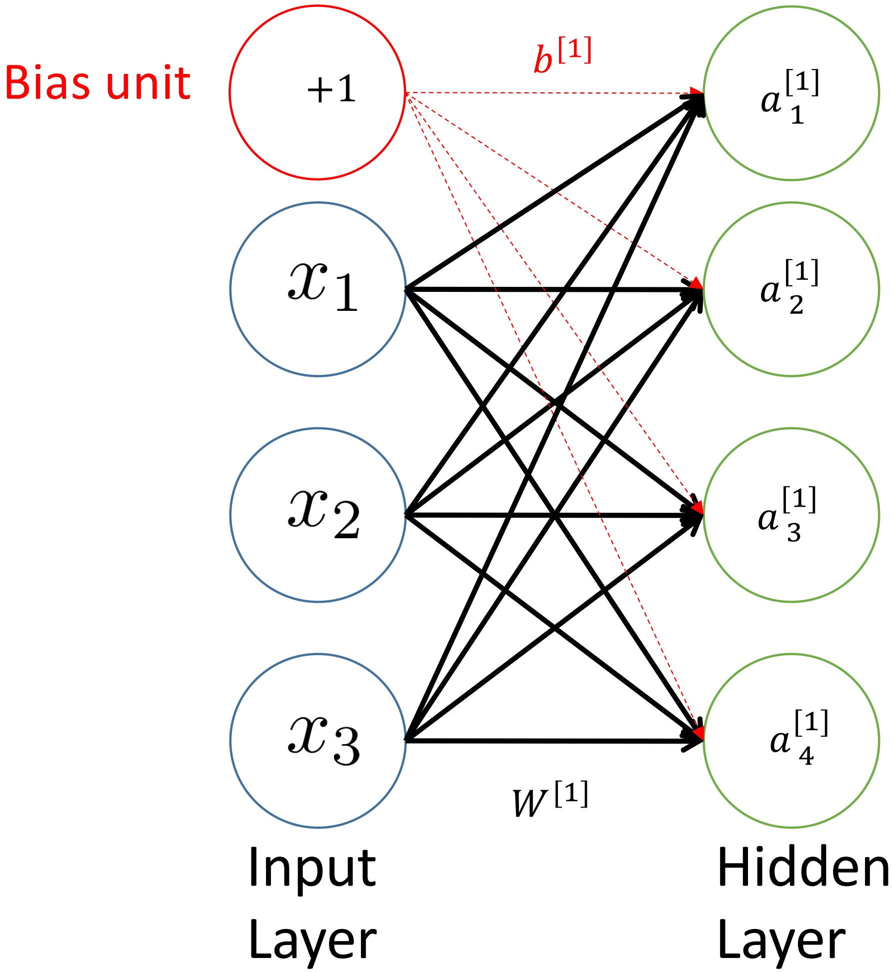
ANN Model Representation



$$
\begin{array}{cccccc}
z^{[1]} & & W^{[1]} & & b^{[1]} & a^{[1]} \\
\boxed{\begin{matrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{matrix}} & \begin{matrix} = \\ = \\ = \\ = \end{matrix} & \boxed{\begin{matrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{matrix}} & \begin{matrix} x + \\ x + \\ x + \\ x + \end{matrix} & \boxed{\begin{matrix} b_1^{[1]}, \\ b_2^{[1]}, \\ b_3^{[1]}, \\ b_4^{[1]}, \end{matrix}} & \boxed{\begin{matrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{matrix}} \begin{matrix} = g(z_1^{[1]}) \\ = g(z_2^{[1]}) \\ = g(z_3^{[1]}) \\ = g(z_4^{[1]}) \end{matrix}
\end{array}
$$

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

ANN Model Representation

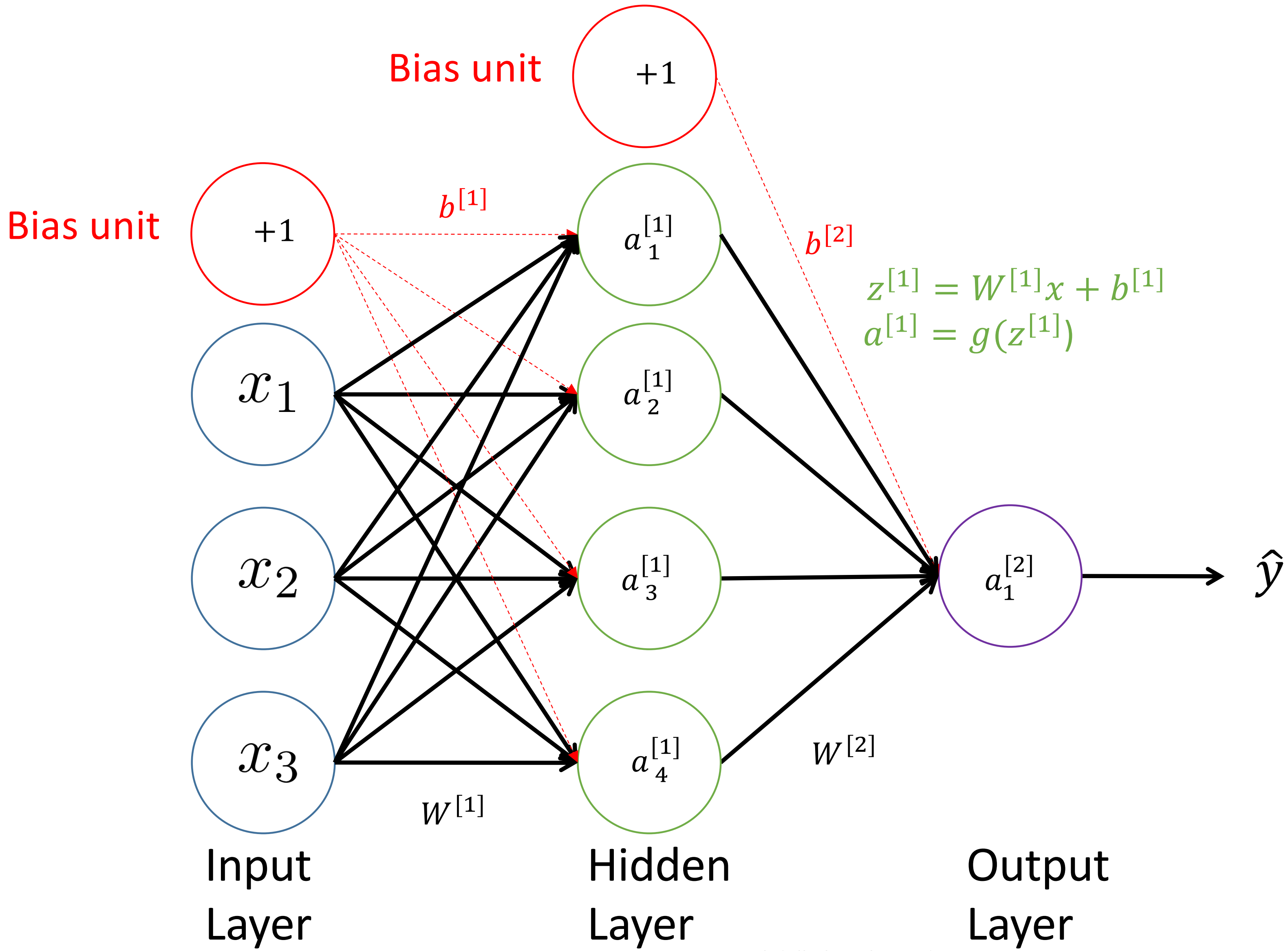


$$\begin{array}{cccc} z^{[1]} & W^{[1]} & b^{[1]} & a^{[1]} \end{array}$$

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

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

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



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

ANN Model Representation: Vectorization

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

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

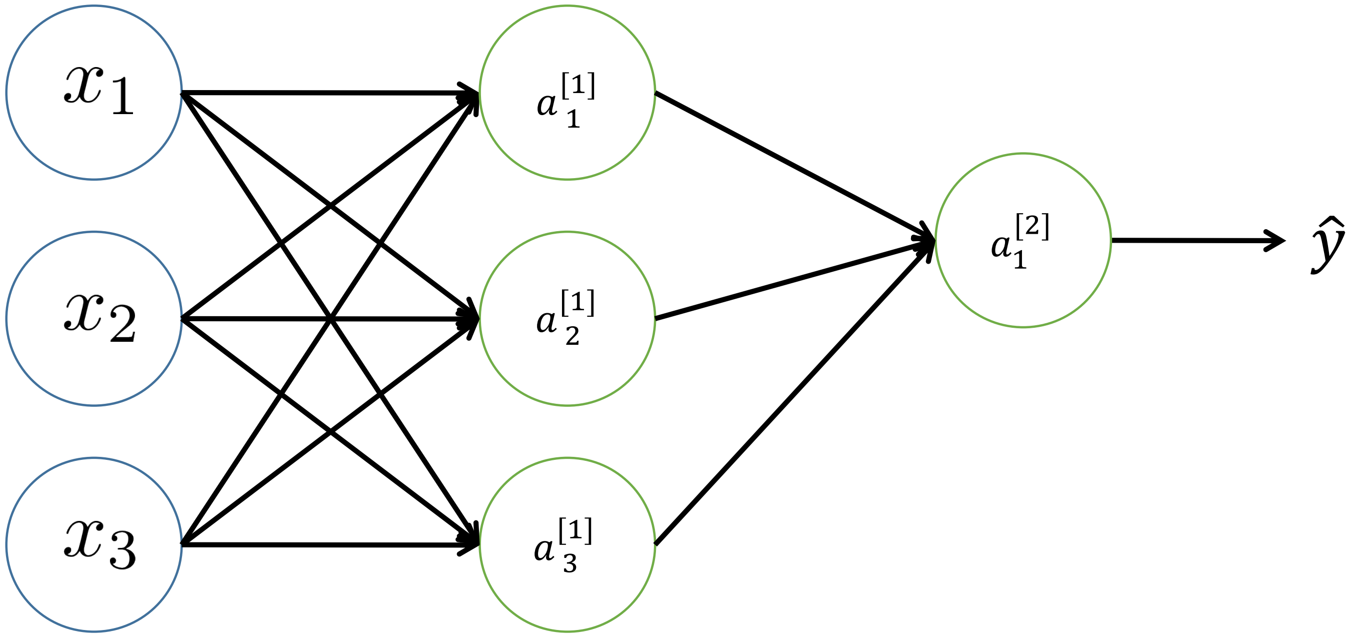
For one example

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

For All Examples (matrix notation)

ANN Representation: Learning its own features

Input Features

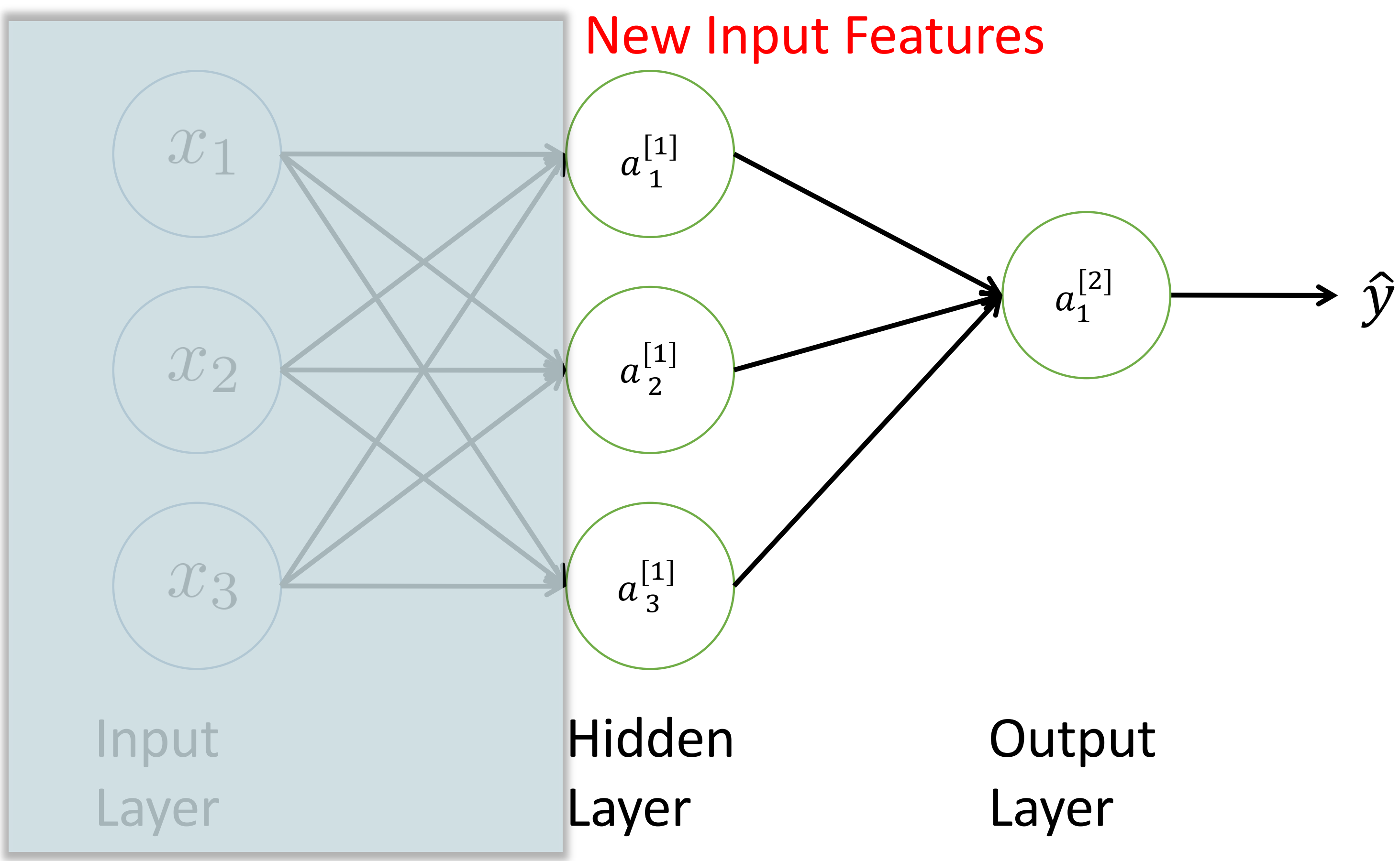


Input Layer

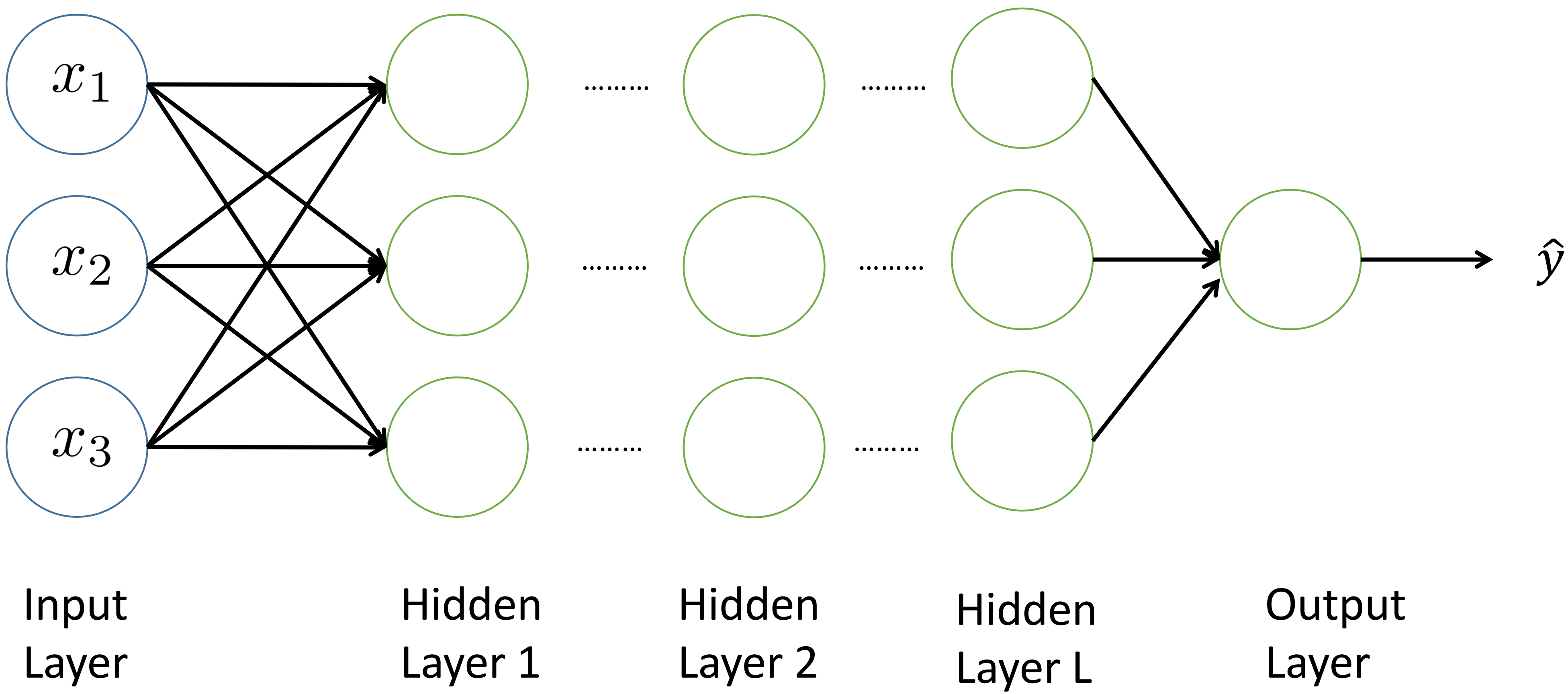
Hidden Layer

Output Layer

ANN Representation: Learning its own features



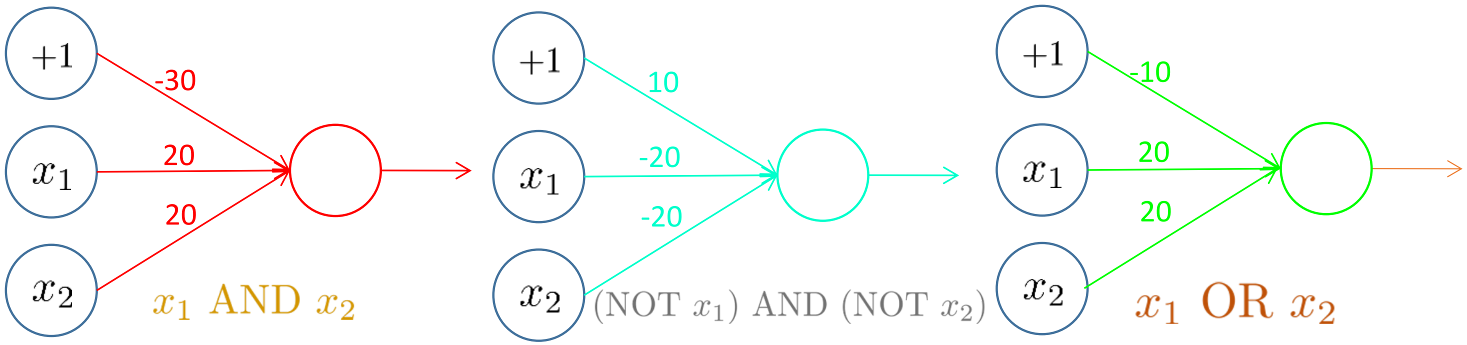
ANN Representation: Deep NN

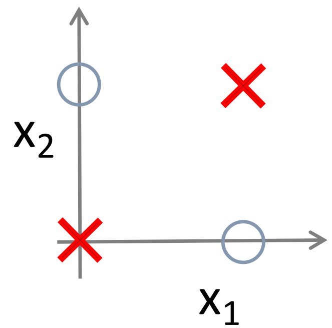


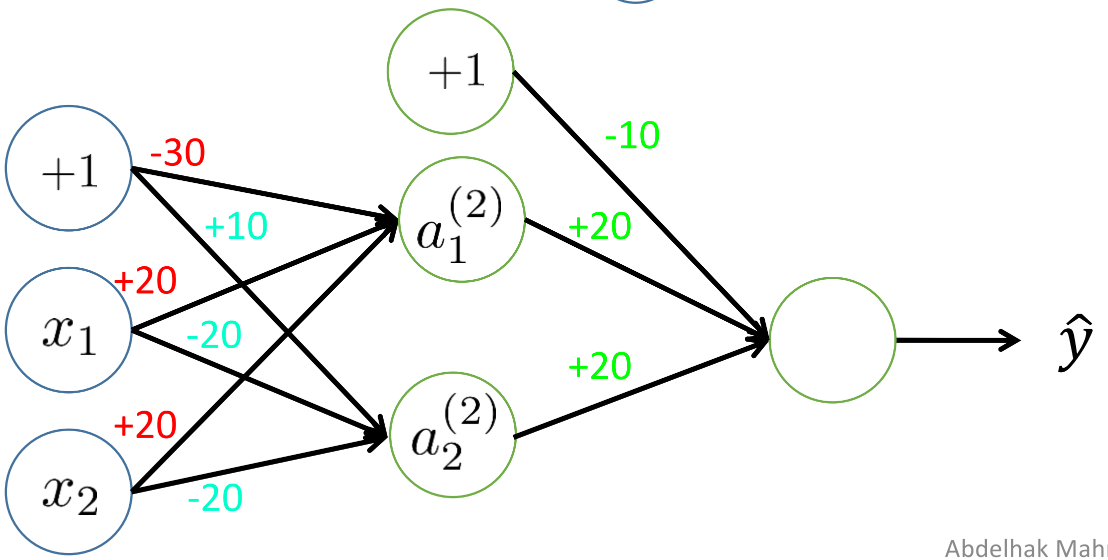
Why ANN?

- 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

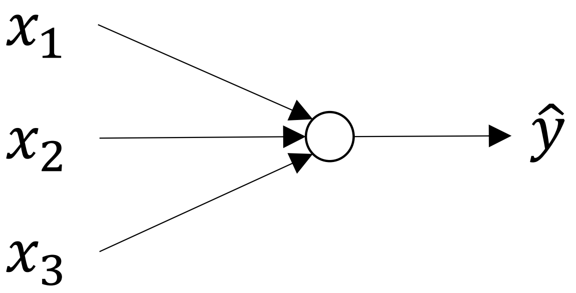






x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$\hat{y}$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

ANN Learning



Remember Logistic Regression

$$x \longrightarrow \boxed{z = w^T x + b} \longrightarrow \boxed{a = g(z)} \longrightarrow \boxed{\mathcal{L}(a, y)}$$

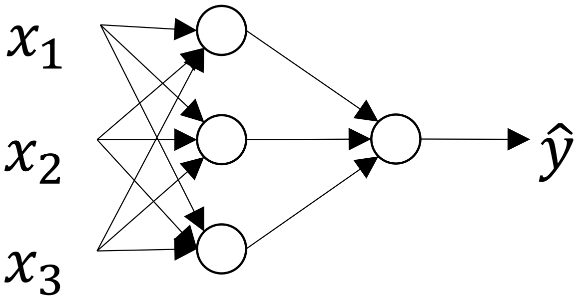
dw dz da

db

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



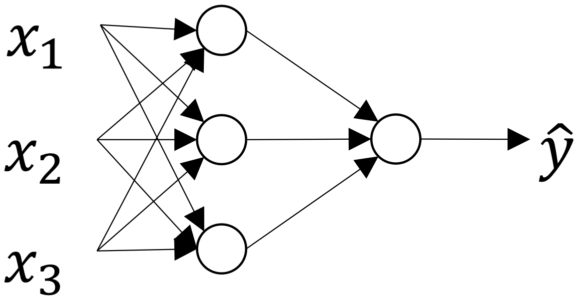
$$x,\ W^{[1]},\ b^{[1]} \rightarrow \boxed{z^{[1]} = W^{[1]}x + b^{[1]}} \rightarrow \boxed{a^{[1]} = g(z^{[1]})} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}} \rightarrow \boxed{a^{[2]} = g(z^{[2]})} \rightarrow \boxed{\mathcal{L}(a^{[2]}, y)}$$

$$W^{[2]},\ b^{[2]} \rightarrow \boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}}$$

Cost Function

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

ANN Learning: Backward Propagation



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

$$\begin{matrix} dW^{[1]} \\ db^{[1]} \end{matrix} \leftarrow dz^{[1]} \leftarrow da^{[1]} \leftarrow dz^{[2]} \leftarrow da^{[2]}$$

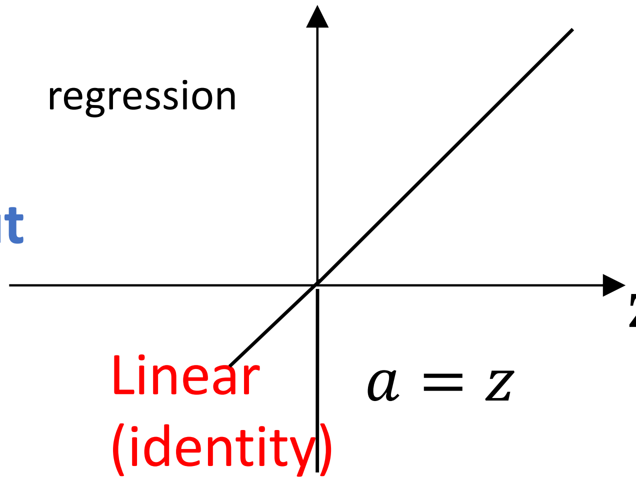
$dW^{[2]}$

$db^{[2]}$

ANN Learning: Activation Functions

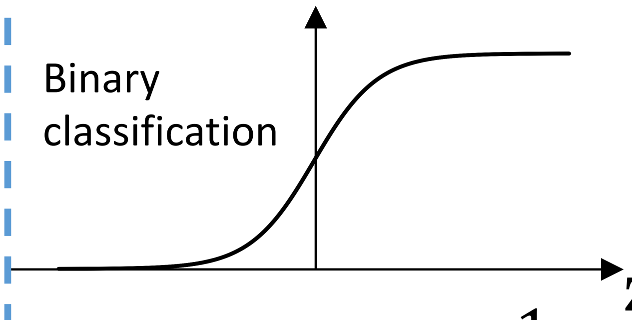
Output Layer

regression



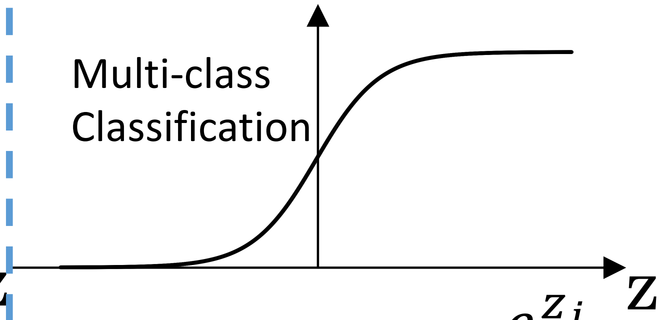
Linear (identity) $a = z$

Binary classification



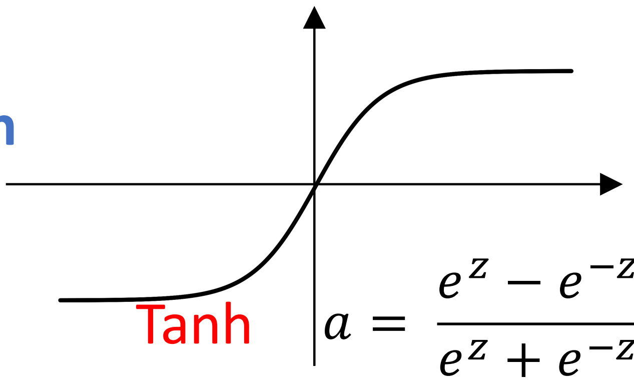
Sigmoid $a = \dfrac{1}{1 + e^{-z}}$

Multi-class Classification

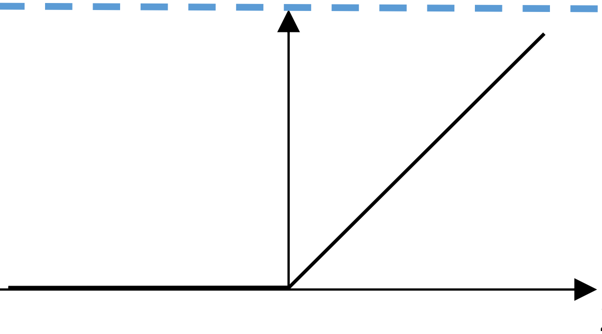


Softmax $a_j = \dfrac{e^{z_j}}{\sum_{j=1}^{K} e^{z_j}}$

Hidden Layers

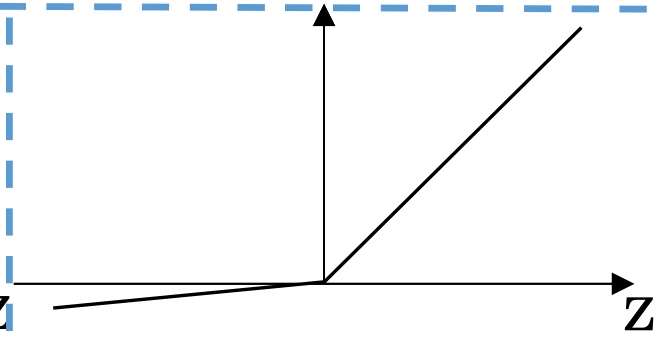


Tanh $a = \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$



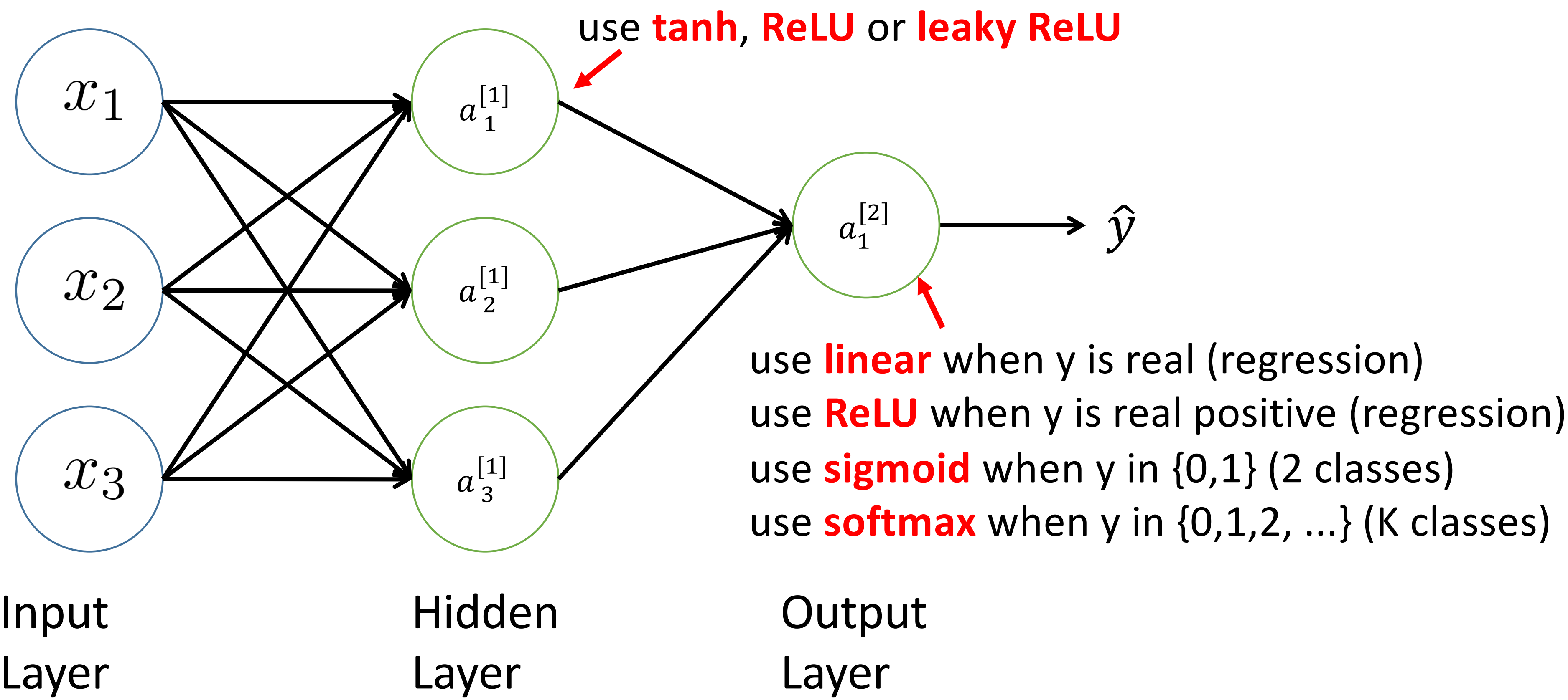
ReLU $a = \max(0, z)$

Rectified Linear Unit



Leaky ReLU $a = \max(0.01z, z)$

ANN Learning : Activation Functions



f ∘ g	(f′ ∘ g) × g′
f(g(x))	f′(g(x))g′(x)
$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

ANN Learning : Backward Propagation

$W^{[1]}$ $b^{[1]}$ $W^{[2]}$ $b^{[2]}$

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

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

$dW^{[2]}$ $db^{[2]}$

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)

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]^T}$$

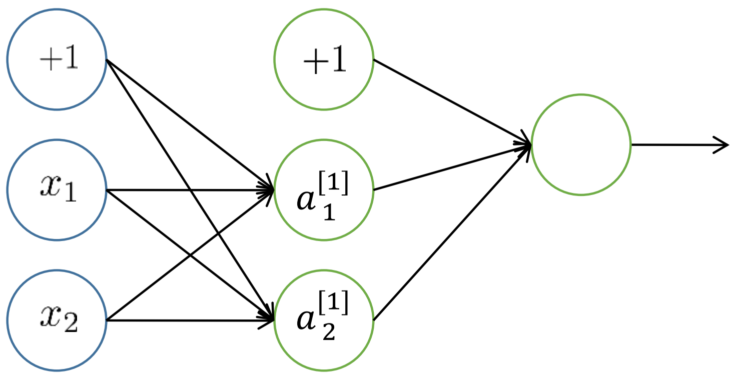
$$db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

$$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$$

$$dW^{[1]} = \frac{1}{m}dZ^{[1]}X^T$$

$$db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

ANN Learning: Random Initialization

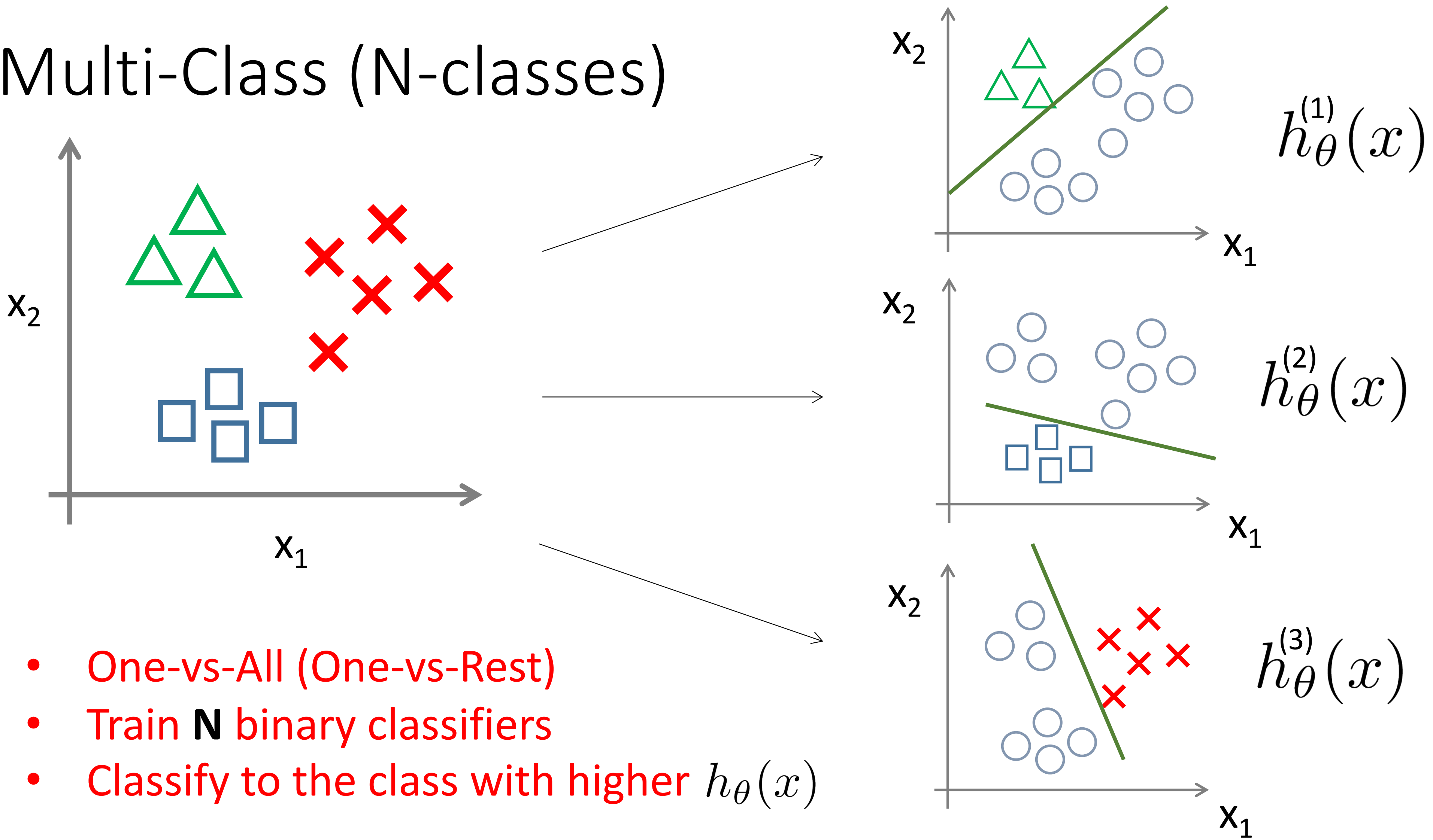


If $W^{[1]}$ and $b^{[1]}$ are initialized with zeros, after each update, parameters corresponding to inputs going into each of the two hidden units are identical, which results in $a_1^{[1]} = a_2^{[1]}$

ANN Implementation

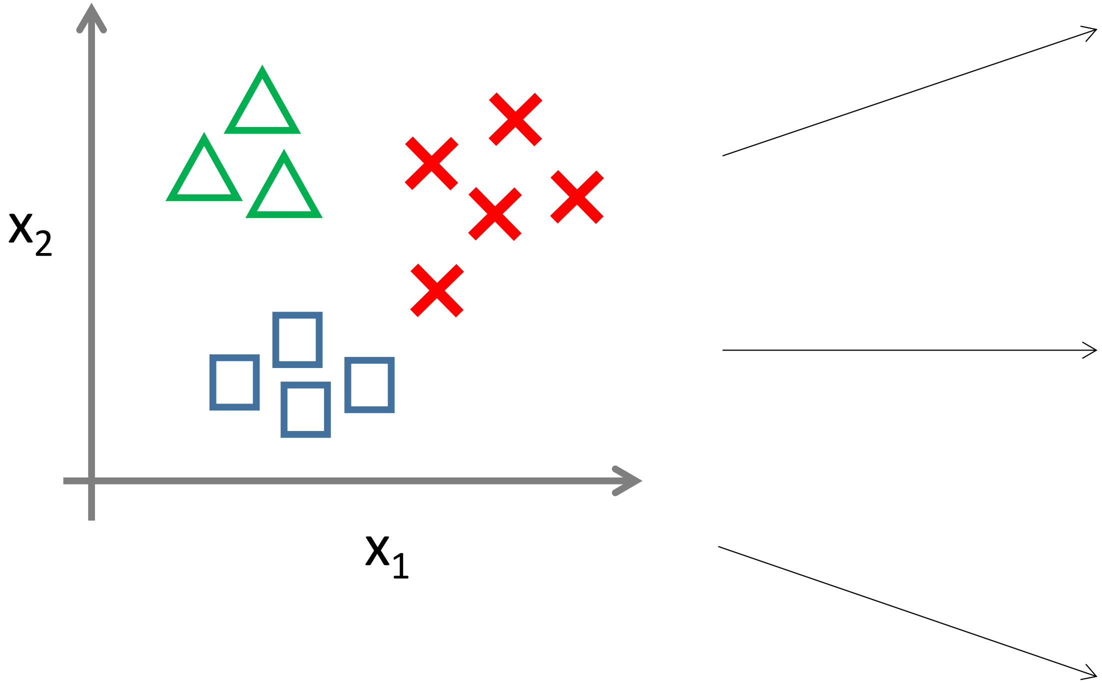
1. Pick a network architecture
   - No. of input units: Dimension of features
   - No. output units: Number of classes
   - No. hidden layers
   - No. hidden units: (have same no. in every layer, the more the better)
2. Randomly initialize weights
3. Repeat (chose a number of iterations)
   1. Implement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

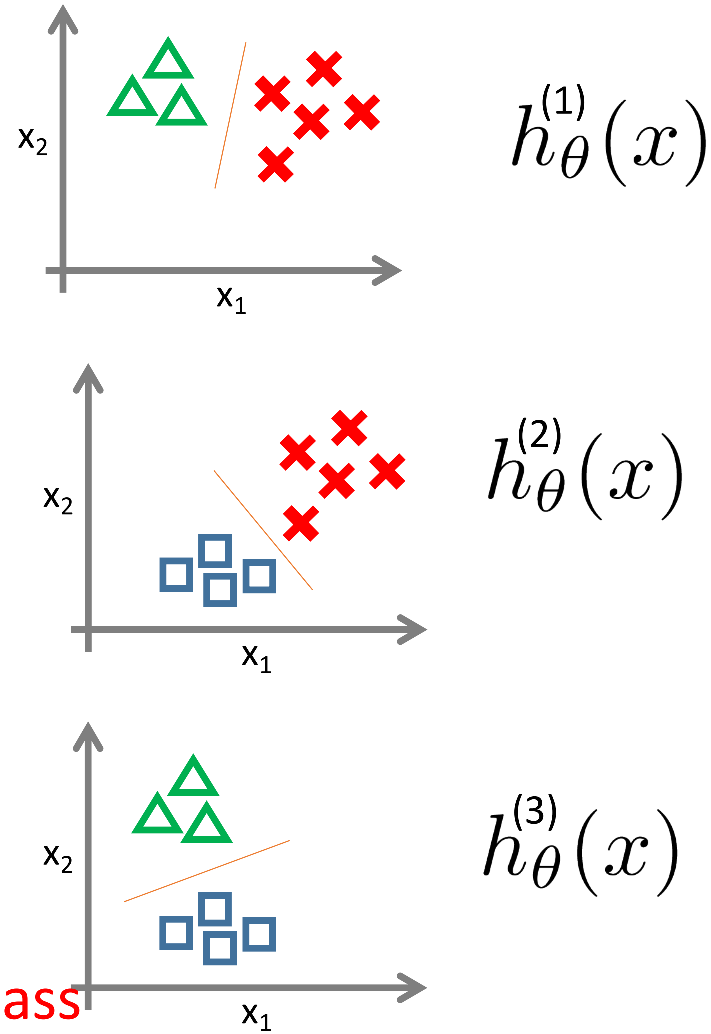
Multi-Class (N-classes)



- 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