Computational Intelligence

Samaneh Hosseini

Isfahan University of Technology

Outline

- Vectorization on Logistic Regression
- Vectorization on Neural Networks

Vectorization

What is vectorization?

$$z = w^T x + w_0$$

non_vectorized:

```
z = 0
for i in range(n_x)
        z += w[i] * x[i]
z += w_0
```

$w \in \mathbb{R}^{n_x}$ $\quad$ $x \in \mathbb{R}^{n_x}$

$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $\quad$ $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$

vectorized

z = np.dot(w, x) + w_0

GPU
CPU $\}$ → SIMD

single instruction multiple data

```python
a = np.random.rand(10000000)
b = np.random.rand(10000000)

# Numpy outer product implementation
n_tic = time.process_time()
outer_product = np.dot(a, b)
n_toc = time.process_time()
print("numpy_dot_product = "+str(outer_product))
print("Time = "+str(1000*(n_toc - n_tic))+"ms")

# # pure Python outer product implementation
tic = time.process_time()
outer_product = 0
for i in range(10000000):
    outer_product += a[i] * b[i]
toc = time.process_time()
print("python_outer_product = "+ str(outer_product))
print("Time = "+str(1000*(toc - tic ))+"ms\n")
```

```
numpy_dot_product = 2500326.8572343425
Time = 125.0ms
python_outer_product = 2500326.8572343607
Time = 6875.0ms
```

Vectorizing Logistic Regression

Vectorizing Logistic Regression

$$z^{(1)} = w^T x^{(1)} + b \qquad z^{(2)} = w^T x^{(2)} + b \qquad z^{(3)} = w^T x^{(3)} + b$$

$$a^{(1)} = \sigma(z^{(1)}) \qquad a^{(2)} = \sigma(z^{(2)}) \qquad a^{(3)} = \sigma(z^{(3)})$$

training data $\underline{n}$

Vectorizing Logistic Regression's Gradient Computation

$dz^{(1)} = a^{(1)} - y^{(1)}$ $\quad dz^{(2)} = a^{(2)} - y^{(2)}$ $\quad \ldots$

$\to dZ = [dz^{(1)}\ dz^{(2)} \ldots dz^{(n)}]$

$A = [a^{(1)} \ldots a^{(n)}]$ $\quad Y = [y^{(1)} \ldots y^{(n)}]$

$dZ = A - Y = [a^{(1)} - y^{(1)}\quad a^{(2)} - y^{(2)} \ldots a^{(n)} - y^{(n)}]$

$dw = 0$

$dw += x^{(1)} dz^{(1)}$

$dw += x^{(2)} dz^{(2)}$

$\vdots$

$dw /= n$

$db = 0$

$db += dz^{(1)}$

$db += dz^{(2)}$

$\vdots$

$db += dz^{(n)}$

$db /= n$

$db = \frac{1}{n} \sum_{i=1}^{n} dz^{(i)}$

$\boxed{db = \frac{1}{n} np.sum(dz)}$

$\boxed{dw = \frac{1}{n} X dz^T}$

$= \frac{1}{n} \begin{bmatrix} | & & | \\ x^{(1)} & \ldots & x^{(n)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(n)} \end{bmatrix}$

$n_x \times n \qquad n \times 1$

$= \frac{1}{n} [x^{(1)} dz^{(1)} + x^{(2)} dz^{(2)} \ldots x^{(n)} dz^{(n)}]$

$n_x \times 1$

Implementing Logistic Regression

vectorized

non-vectorized

$J = 0, dw_1 = 0, dw_2 = 0, db = 0$

for i = 1 to n:

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$

$dw \mathrel{+}= x^{(i)} dz^{(i)}$

$db \mathrel{+}= dz^{(i)}$

$J = J/n, dw_1 = dw_1/n, dw_2 = dw_2/n$

$db = db/n$

for iter in range(1000):

$Z = w^T X + b$

$= np.dot(w.T, X) + b$

$A = tf.nn.sigmoid(Z)$

$dZ = A - Y$

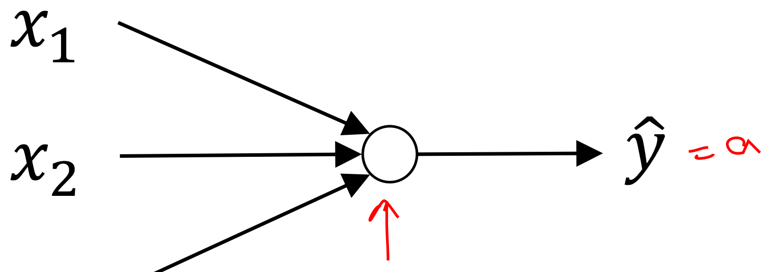
$dw = \frac{1}{n} X dZ^T$

$db = \frac{1}{n} np.sum(dZ)$

$w = w - \eta dw$

$b = b - \eta db$

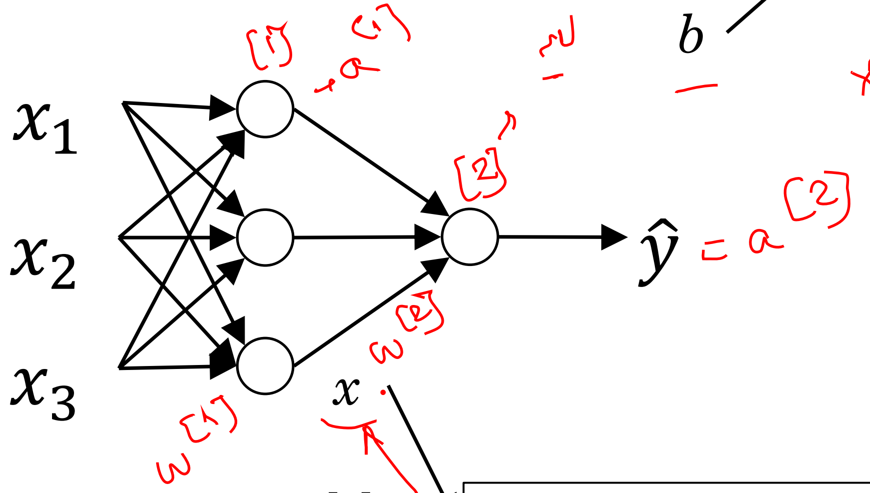
Expanding Formulations on Neural Networks



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

dz $\qquad$ da

$x^{(i)} \longrightarrow$ training data



$da^{[1]}$ $\qquad$ $dz^{[2]}$ $\qquad$ $da^{[2]}$

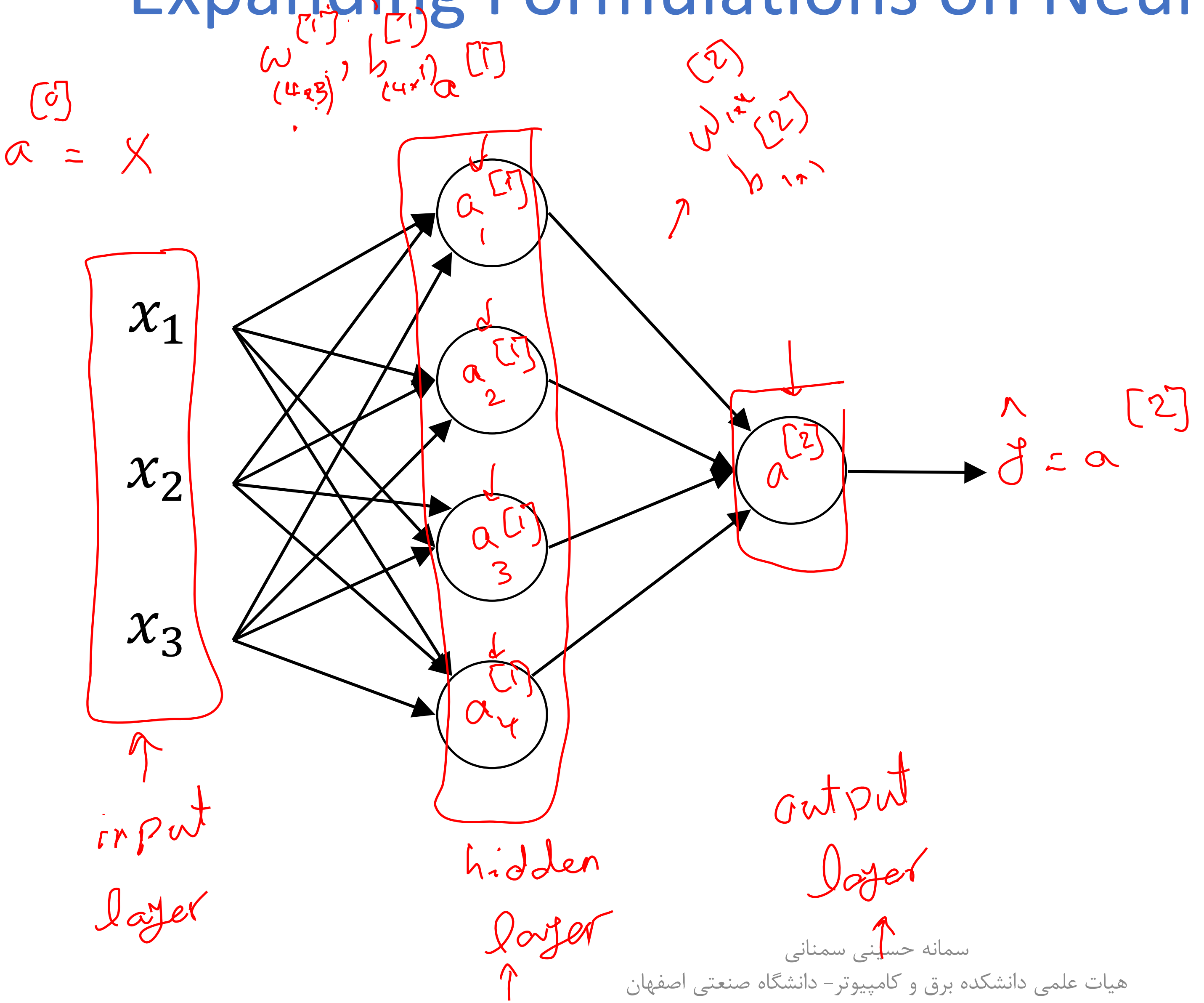
x, $W^{[1]}$, $b^{[1]}$ $\rightarrow$ $\boxed{z^{[1]} = W^{[1]}x + b^{[1]}}$ $\rightarrow$ $\boxed{a^{[1]} = \sigma(z^{[1]})}$ $\rightarrow$ $\boxed{z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}}$ $\rightarrow$ $\boxed{a^{[2]} = \sigma(z^{[2]})}$ $\rightarrow$ $\boxed{\mathcal{L}(a^{[2]}, y)}$

$dw^{[2]}$ $\quad$ $w^{[2]}$

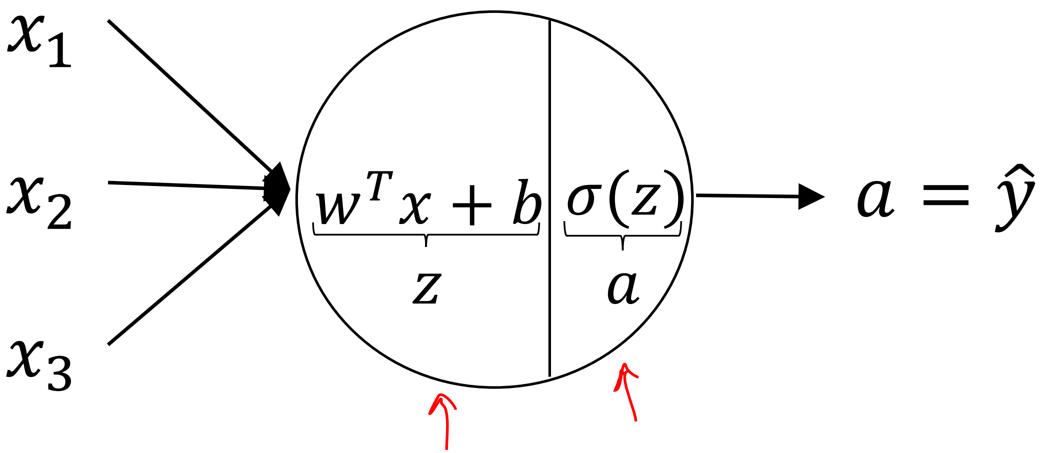
$db^{[2]}$ $\quad$ $b^{[2]}$

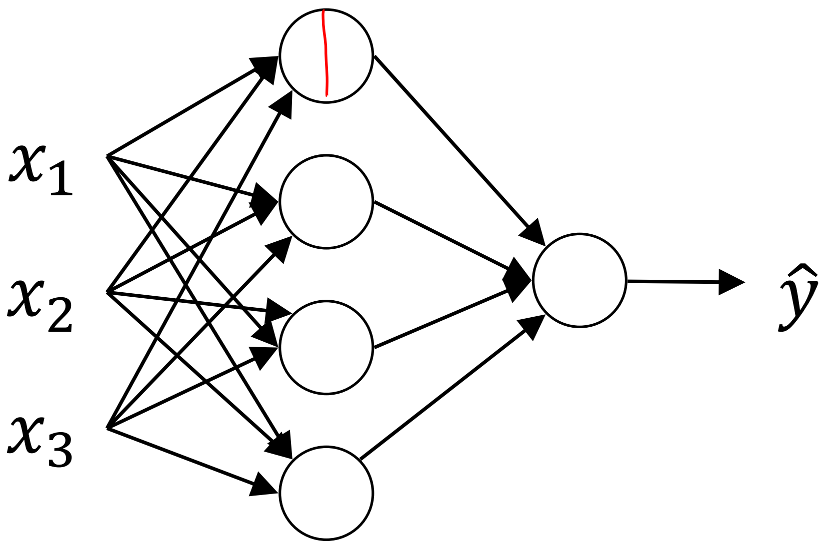
Expanding Formulations on Neural Networks

2 layer NN



Computing a Neural Network’s Output

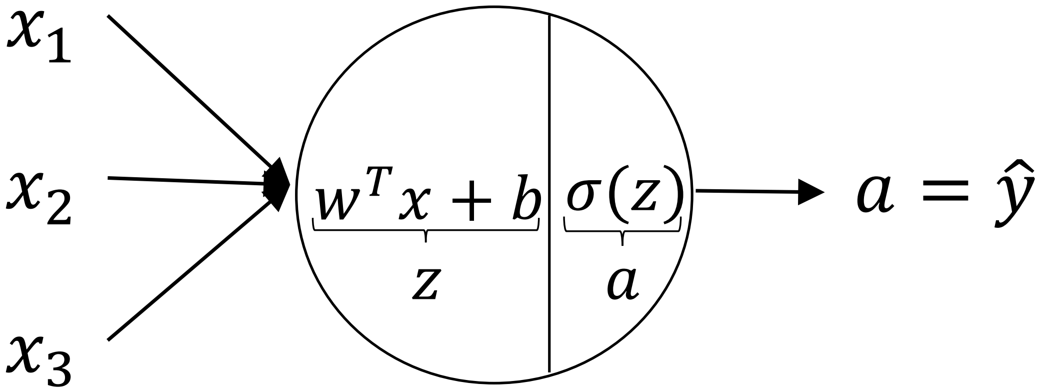




$$z = w^T x + b$$

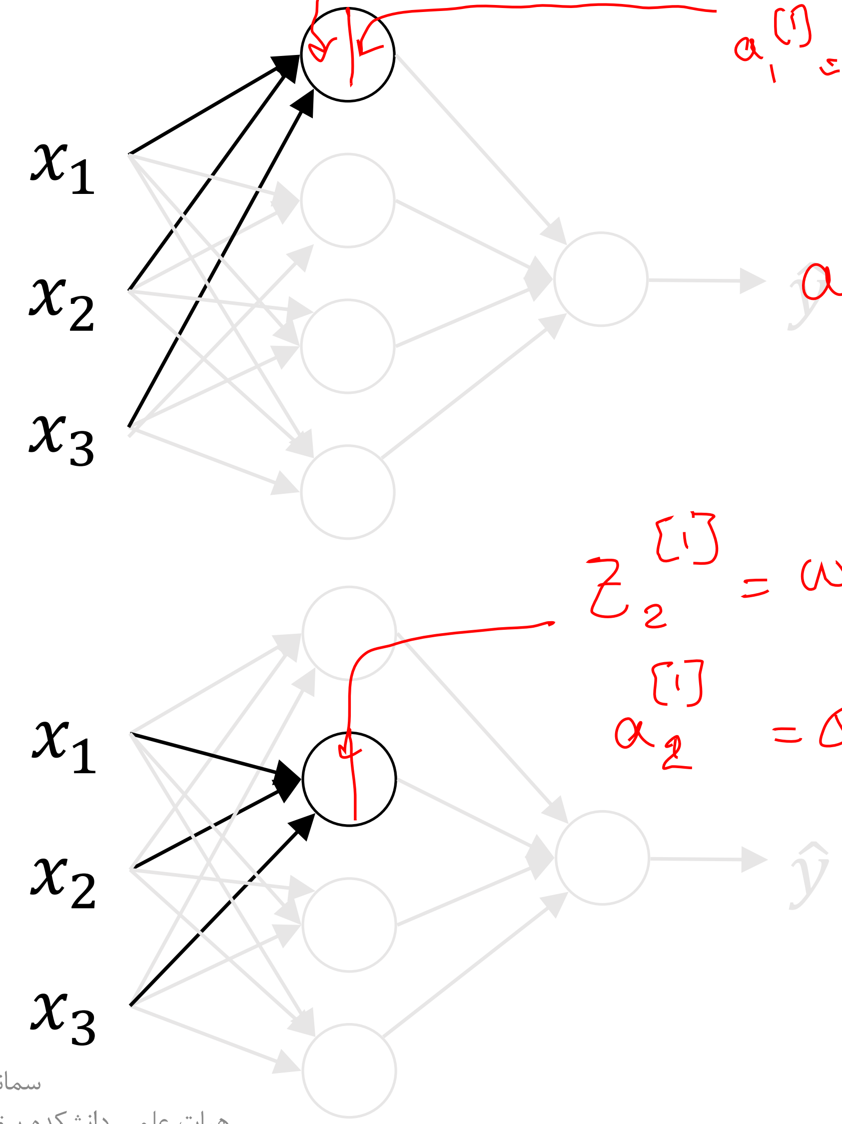
$$a = \sigma(z)$$

Computing a Neural Network's Output



$z = w^T x + b$

$a = \sigma(z)$



$z_1^{[1]} = w_1^{[1]T} X + b_1^{[1]}$

$a_1^{[1]} = \sigma(z_1^{[1]})$

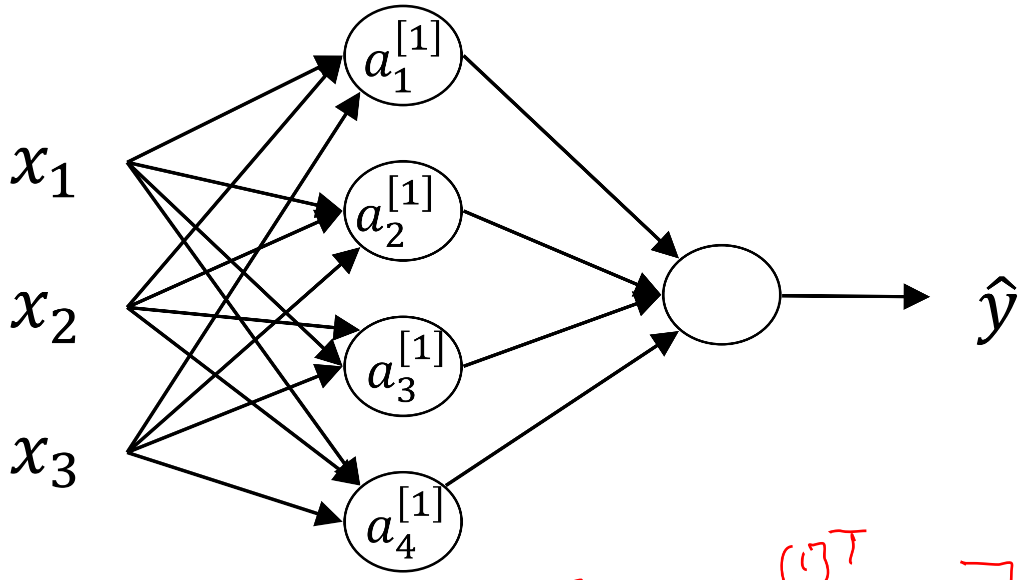
$a_i^{[0]}$ ← layer

i → node in layer

$z_2^{[1]} = w_2^{[1]T} X + b_2^{[1]}$

$a_2^{[1]} = \sigma(z_2^{[1]})$

Computing a Neural Network's Output



$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

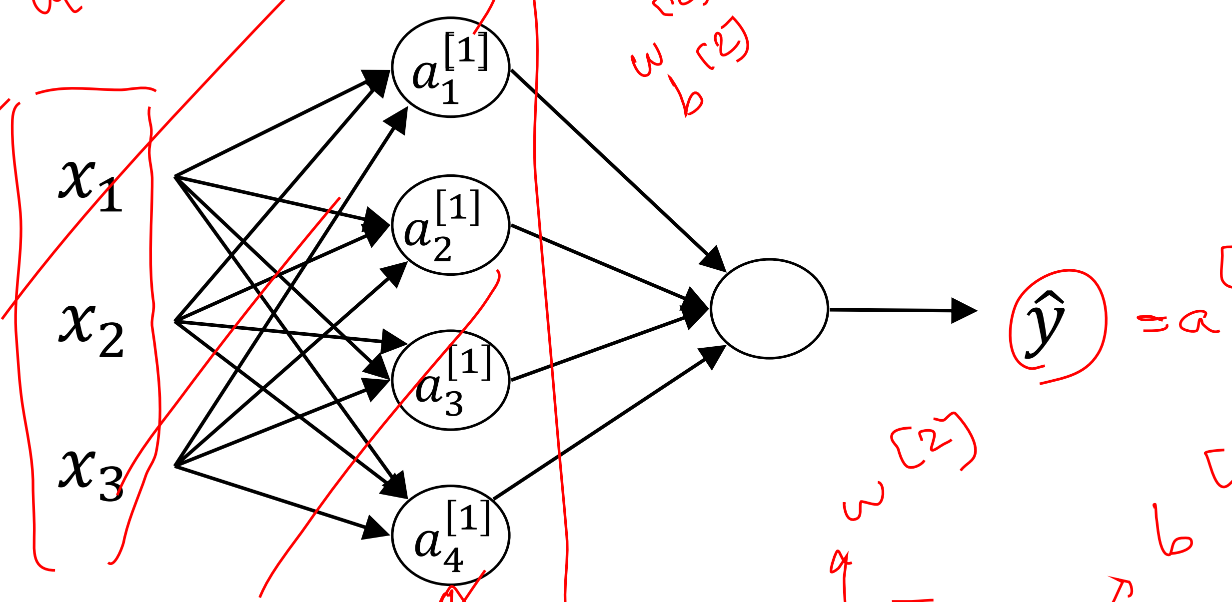
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

$$Z^{[1]} = \underbrace{\begin{bmatrix} — w_1^{[1]T} — \\ — w_2^{[1]T} — \\ — w_3^{[1]T} — \\ — w_4^{[1]T} — \end{bmatrix}}_{4\times 3} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \\ w_4^{[1]T} x + b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(Z^{[1]})$$

Computing a Neural Network's Output



$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

Given input x:

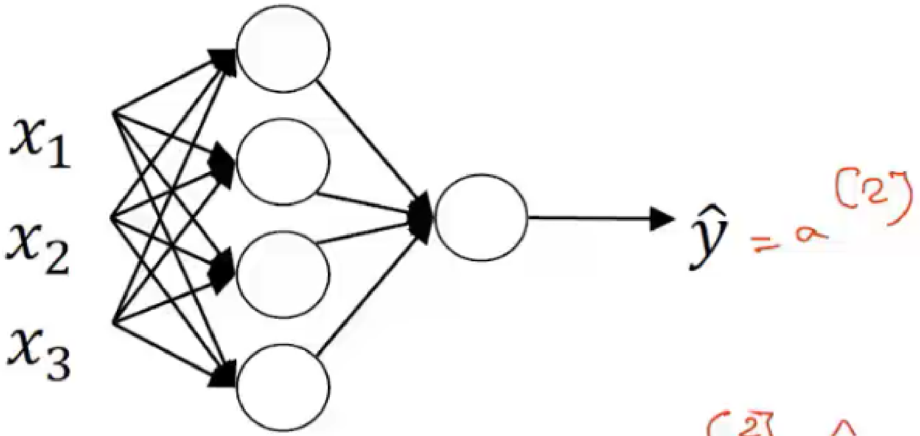
$$\underset{4\times 1}{z^{[1]}} = \underset{(4\times 3)}{W^{[1]}} \underset{3\times 1}{\overset{a^{[0]}}{\cancel{x}}} + \underset{4\times 1}{b^{[1]}}$$

$$\underset{4\times 1}{a^{[1]}} = \sigma(\underset{4\times 1}{z^{[1]}})$$

$$\underset{1\times 1}{z^{[2]}} = \underset{1\times 4}{W^{[2]}} \underset{4\times 1}{a^{[1]}} + \underset{1\times 1}{b^{[2]}}$$

$$\underset{1\times 1}{a^{[2]}} = \sigma(\underset{1\times 1}{z^{[2]}})$$

Vectorizing Across Multiple Examples



$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

$X \longrightarrow a^{[2]} = \hat{y}$

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(n)}$

$a^{[2](i)}$ ← example i
↑ layer 2

for $i=1$ to n

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

Vectorizing across multiple examples

for i = 1 to n:

$\boxed{z^{[1](i)}} = W^{[1]}x^{(i)} + b^{[1]}$

$\boxed{a^{[1](i)}} = \sigma(z^{[1](i)})$

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

$\boxed{a^{[2](i)}} = \sigma(z^{[2](i)})$

$Z^{[1]} = W^{[1]}X + b^{[1]}$

$A^{[1]} = \sigma(Z^{[1]})$

$\rightarrow Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

$\rightarrow A^{[2]} = \sigma(Z^{[2]})$

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ \vdots & \vdots & & \vdots \end{bmatrix}_{n_x, n}$$

$$Z^{[1]} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](n)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

example

hidden unit

$$A^{[1]} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](n)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Recap

- Vectorization on Logistic Regression
- Vectorization on Neural Networks