# NEURAL NETWORKS & DEEP LEARNING

By:
Satish Deshbhratar

# BINARY CLASSIFICATION

- $(x, y), x \in R^{n_x}, y \in \{0,1\}$
- $m: \#Training\ examples$
- $m_{test}: \#Test\ examples$
- $X = \begin{bmatrix} \vdots & \cdots & \vdots \\ x^{(1)} & \cdots & x^{(m)} \\ \vdots & \cdots & \vdots \end{bmatrix} \in (n_x, m)$
- $Y = [\,\mathrm{y}^{(1)} \ \dots \ \mathrm{y}^{(\mathrm{m})}] \in (1, m)$

**<u>Logistic regression</u>:**

- $\hat{y} = P(y = 1|x) = \sigma(z + b); z = w^T \rightarrow \hat{Y} = \sigma(\Theta^{\mathrm{T}}\mathrm{X})$
- <u>Parameters</u>: $w \in R^{n_x}, b \in R \ \rightarrow \Theta = \begin{bmatrix} \theta_0 = b \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$

w

- <u>PS</u>: $X_0 = 1, x \in R^{n_x} + 1$
- <u>Sigmoid function</u>: $\sigma(z) = \frac{1}{1+e^{-z}}$

**<u>Loss error function</u>:** $L(\hat{y}, y) = -ylog\hat{y} + (1 - y)\log(1 - \hat{y}) = \frac{1}{2}(\hat{y} - y)^2$

**<u>Cost function</u>:** $J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

# GRADIENT DESCENT

Want to find $w, b$ that minimize $J(w, b)$

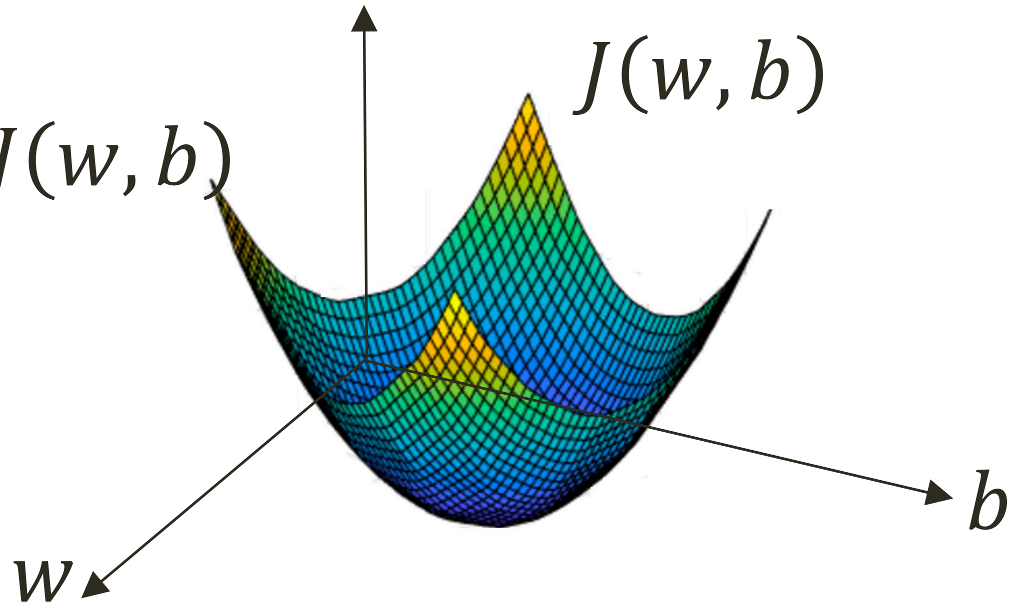
Repeat {

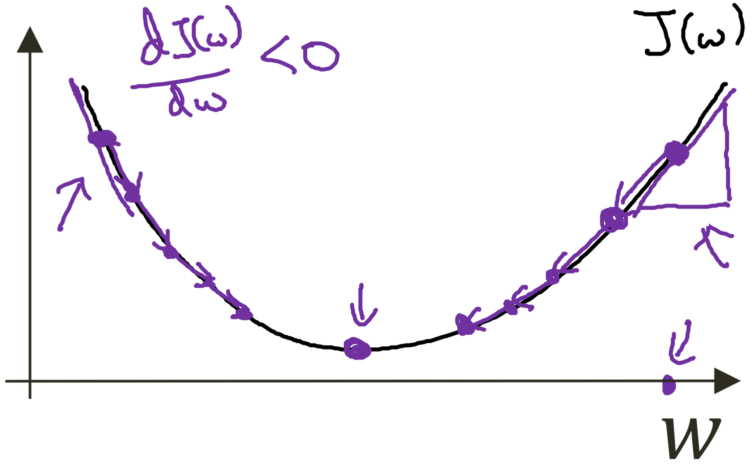
$$w := w - \alpha \frac{dJ(w,b)}{dw}$$

$$b := b - \alpha \frac{dJ(w,b)}{db}$$

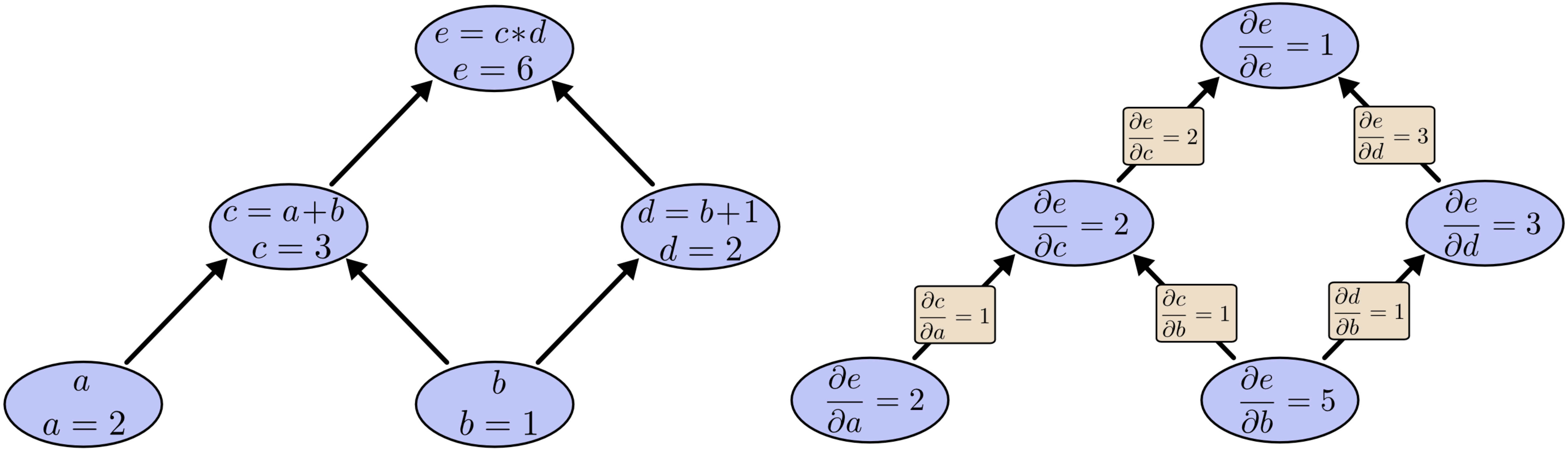
}

$$\boxed{\begin{gathered} \alpha = learning\ rate \\ dw = \frac{dJ(w,b)}{dw} \\ db = \frac{dJ(w,b)}{db} \end{gathered}}$$



# COMPUTATION GRAPH



# LOGISTIC REGRESSION DERIVATIES

$$x_1, w_1, x_2, w_2, b \rightarrow \boxed{z = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(\mathrm{a}, y)}$$

$$dz = \frac{dL}{dz} = a - y$$

$$da = \frac{dL}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$dw = x.dz$$

$$db = dz$$

for i=1 to m:

$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma\left(z^{(i)}\right)$$
$$J += -[y^{(i)} loga^{(i)} + \left(l1 - y^{(i)}\right)\log(1 - a^{(i)})$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$
$$dw_1 += x_1^{(i)} dz^{(i)}$$
$$dw_2 += x_2^{(i)} dz^{(i)}$$
$$db += dz^{(i)}$$

$$J /= m; dw_1 /= m; \mathrm{dw}_2 /= m; db /= m$$

# VECTORIZATION

**Non vectorized:** For loop → Slow

**Vectorized:** Tensor calculations → Fast

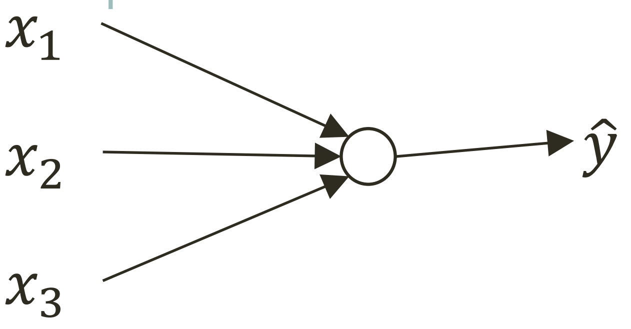
- Whenever possible, avoid explicit for-loops
- $u = Av \ \rightarrow u = np.dot(A, v)$
- $w_1, w_2 \ldots \rightarrow one\ W$

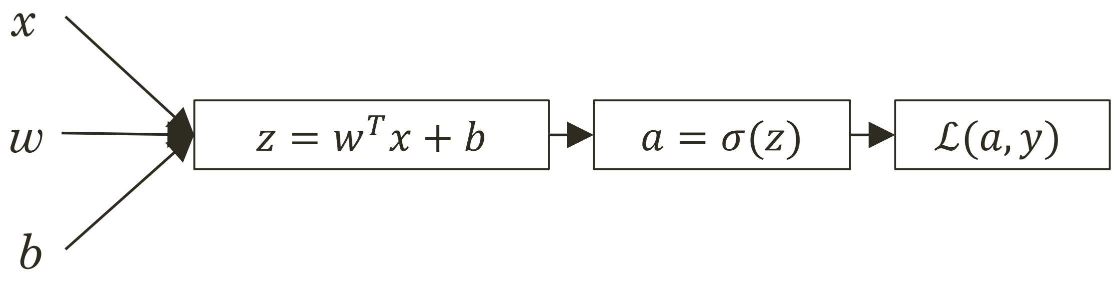
Python
Broadcasting

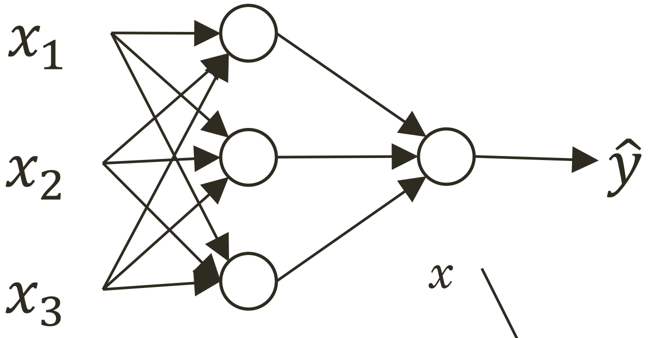
**Vectorizing logistic regression:**

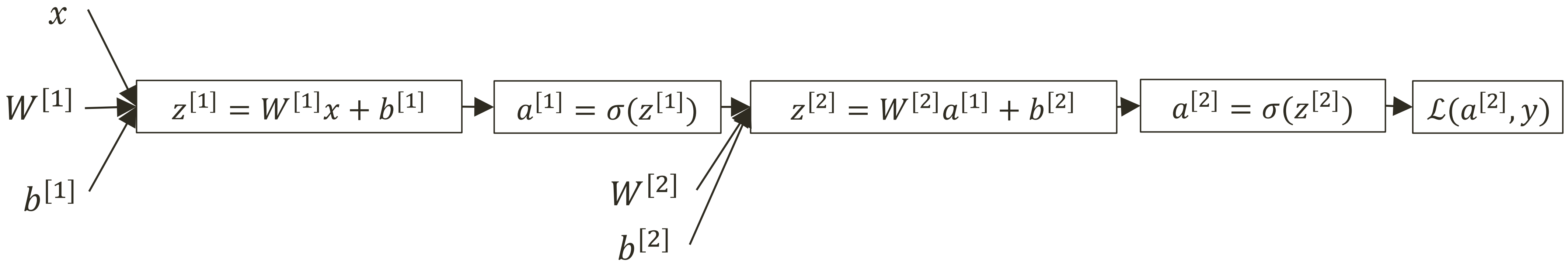
- $Z = \left[z^{(1)} \ldots z^{(n+1)}\right] = w^T X + [b \ldots b] = \left[w^T x^{(1)} + b \ldots w^T x^{(2)} + b\right] = \boldsymbol{np.dot(w.T, X) + b}$
- $A = \left[a^{(1)} \ldots a^{(m)}\right] = \boldsymbol{\sigma(Z)}$
- $dZ^{[L]} = \boldsymbol{A^{[L]} - Y} = \left[a^{(1)} - y^{(1)} \ldots a^{(m)} - y^{(m)}\right]; dZ^{[l]} = w^{[l+1]T} dZ^{[l+1]} * g^{[l]\prime}\left(Z^{[l]}\right)$
- $db^{[l]} = \boldsymbol{\frac{1}{n} np.sum\left(dZ^{[l]}, axis = 1, keepdims = True\right)}, dw^{[l]} = \boldsymbol{\frac{1}{n} dZ^{[l]} A^{[l-1]T}}$
- $w \coloneqq \boldsymbol{w - \alpha dw}; \ \ b \coloneqq \boldsymbol{b - \alpha db}$

# NEURAL NETWORK



# NEURAL NETWORK REPRESENTATION

$$z^{[l]} = \begin{bmatrix} \vdots \\ z_i^{[l]} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ w_i^{[l]T} X + b_i^{[l]} \\ \vdots \end{bmatrix} = W^{[l]}x + b^{[l]}$$

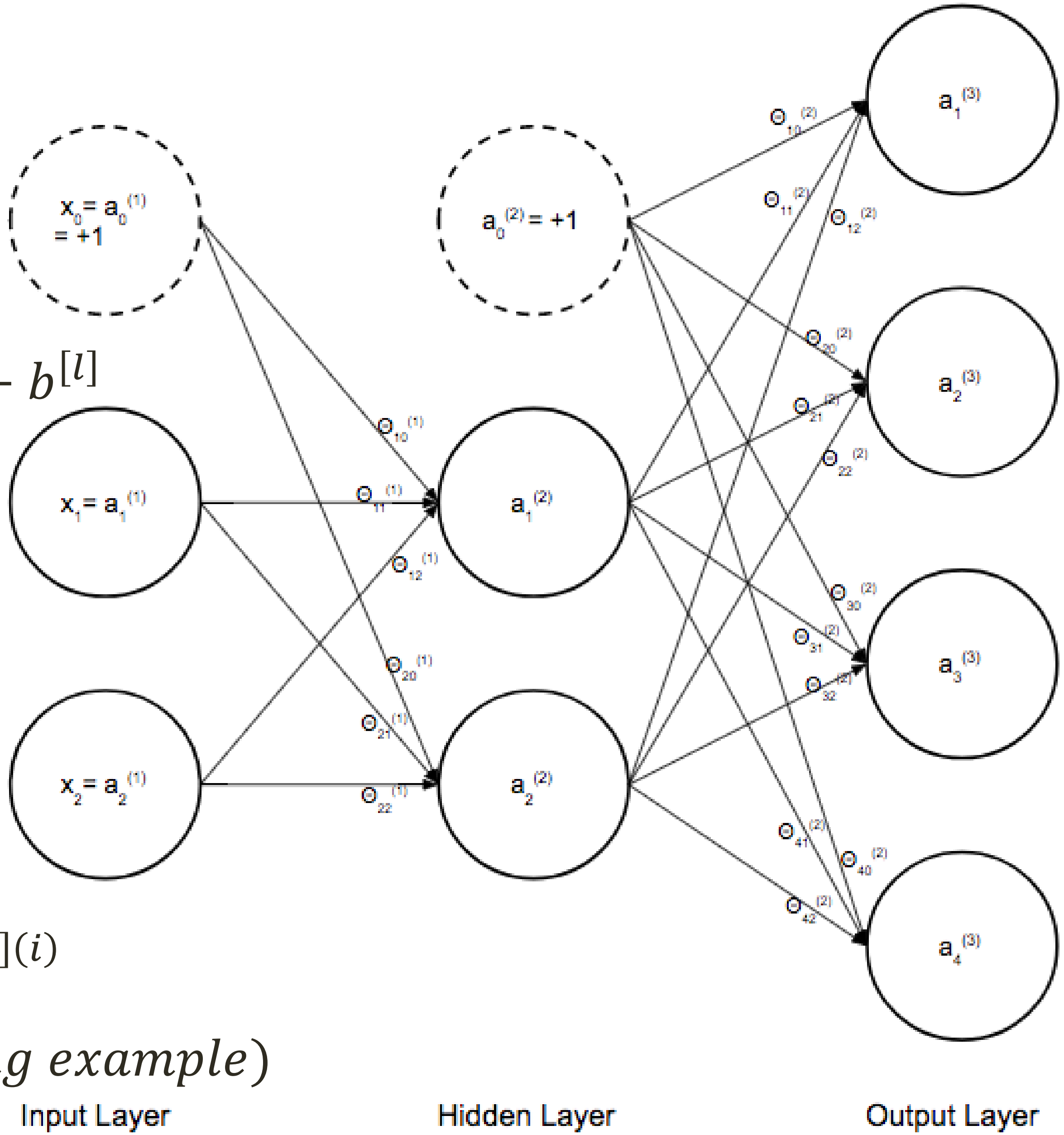
$$a^{[l]} = \begin{bmatrix} \vdots \\ a_i^{[l]} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \sigma\left(z_i^{[l]}\right) \\ \vdots \end{bmatrix} = \sigma(z^{[l]})$$

$a_j^{[l](i)} = activation\ in\ node\ j\ in\ layer\ l$

$in\ example\ i$

➔ $for\ i = 1\ to\ m$: $Calculate\ z^{[l](i)}, a^{[l](i)}$

**PS**: $\mathrm{Z}^{[l]}, \mathrm{A}^{[l]} \in (\#hidden\ units, \#training\ example)$



# MORE ON VECTORIZED IMPLEMENTATION

$z^{[1](1)} = W^{[1]}x^{(1)} + \cancel{b^{[1]}}$ $0$, $\quad z^{[1](2)} = W^{[1]}x^{(2)} + \cancel{b^{[1]}}$ $0$, $\quad z^{[1](3)} = W^{[1]}x^{(3)} + \cancel{b^{[1]}}$ $0$

$$W^{[1]} = \begin{bmatrix} \text{———} \\ \text{———} \\ \text{———} \\ \text{———} \end{bmatrix} \qquad W^{[1]}x^{(1)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(2)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad W^{[1]}x^{(3)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

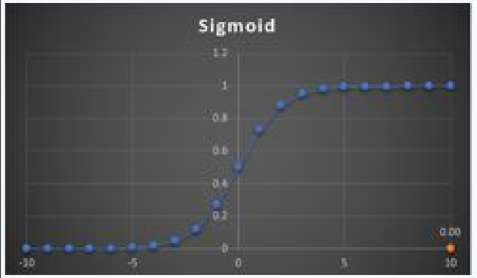
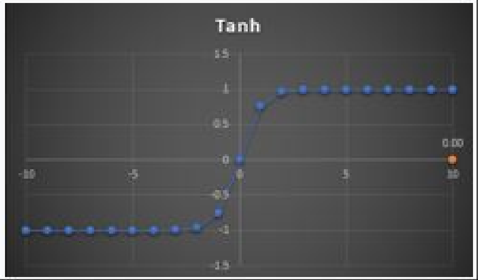
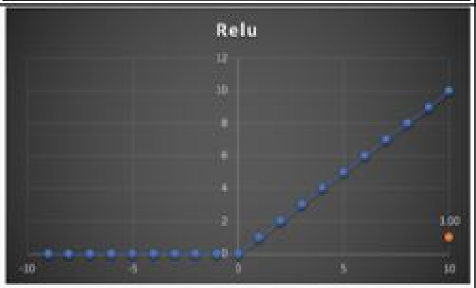
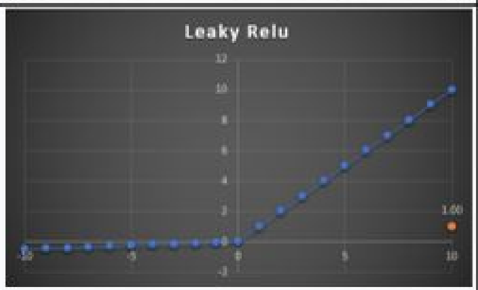
$$W^{[1]} \underbrace{\begin{bmatrix} | & | & | & \\ x^{(1)} & x^{(2)} & x^{(3)} & \cdots \\ | & | & | & \end{bmatrix}}_{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} | & | & | & \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} & \cdots \\ | & | & | & \end{bmatrix} = Z^{[1]}$$

$+b^{[1]}$ $\quad +b^{[1]}$ $\quad +b^{[1]}$

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

$$W^{[1]}x^{(i)} = z^{[1](i)}$$

# ACTIVATION FUNCTIONS

| Name | Plot | Equation | Derivative |
|---|---|---|---|
| Sigmoid |   | $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ | $f'(x) = f(x)(1-f(x))$ |
| Tanh |   | $f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$ | $f'(x) = 1 - f(x)^2$ |
| Rectified Linear Unit (relu) |   | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Leaky Rectified Linear Unit (Leaky relu) |   | $f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |

# SUMMARY: FORWARD AND BACKWARD PROPAGATION

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}\left(Z^{[L]}\right) = \hat{Y}$$

$$dZ^{[L]} = A^{[L]} - Y$$
$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]^T}$$
$$db^{[L]} = \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[L]}, axis = 1, keepdims = True)$$
$$dZ^{[L-1]} = dW^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$
$$\vdots$$
$$dZ^{[1]} = dW^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[1]^T}$$
$$db^{[1]} = \frac{1}{m} np.\mathrm{sum}(\mathrm{d}Z^{[1]}, axis = 1, keepdims = True)$$

# WEIGHT INITIALIZATION AND DIMENSIONALITY

Zero initialization → Activations will be the same

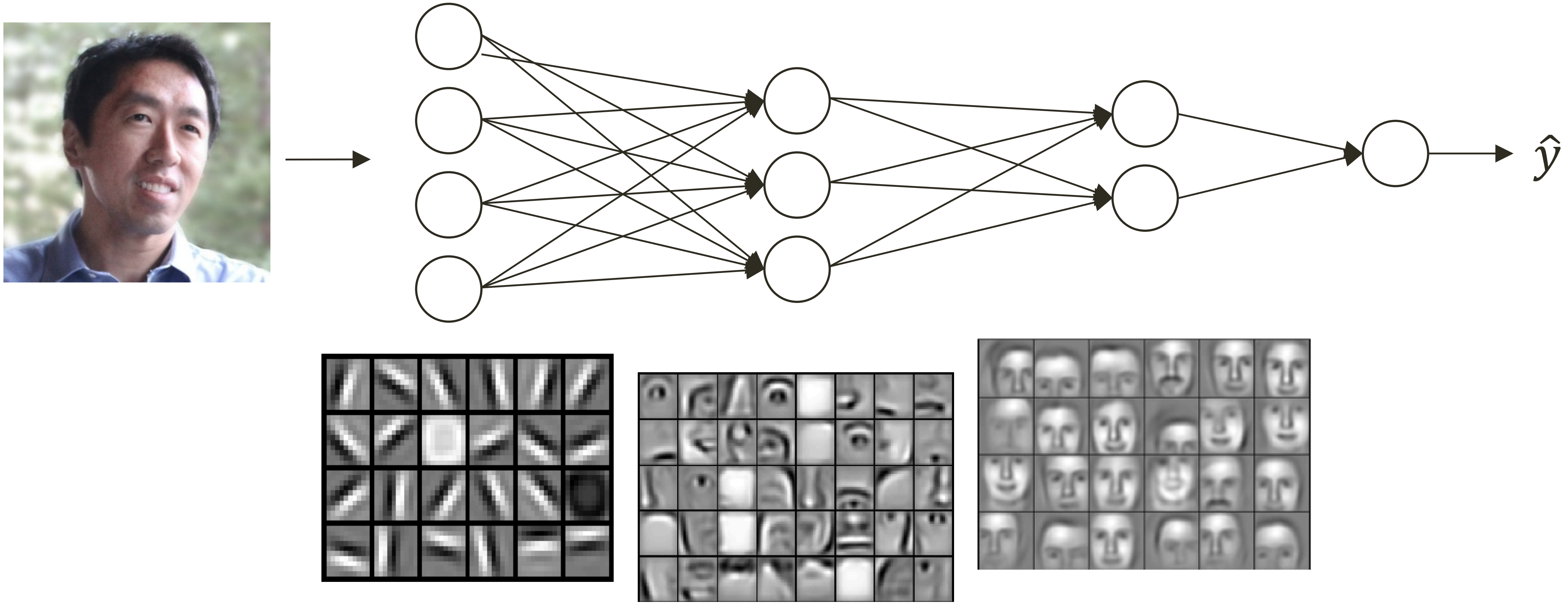
Random initialization → **Good**

PS: b can be initialized to zero

**Notations**:

- $n^{[l]} = \#units\ in\ layer\ l$
- $W^{[l]}, dW^{[l]} \in (n^{[l]}, n^{[l-1]})$
- $A^{[l]}, Z^{[l]}, b^{[l]}, db^{[l]} \in (n^{[l]}, 1)$
- $A^{[l]}, dA^{[l]}, Z^{[l]}, dZ^{[l]} \in (n^{[l]}, m)$

# DEEP REPRESENTATION



# HYPERPARAMETERS

**Parameters**: $W, b$

**Hyperparameters**: $Learning\ rate\ \alpha, \#iterations, \#hidden\ layer\ L,$

$\#hidden\ units, choice\ of\ activation\ functions$

**PS**: Applying deep learning is a very empirical process