

BINARY CLASSIFICATION

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

Logistic regression:

- $\hat{y} = P(y = 1|x) = \sigma(z + b); z = w^T \rightarrow \hat{Y} = \sigma(\Theta^T X)$
- Parameters: $w \in R^{n_x}, b \in R \rightarrow \Theta = \begin{bmatrix} \theta_0 = b \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \quad \left. \vphantom{\begin{bmatrix} \theta_0 = b \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix}} \right\} w$
- PS: $X_0 = 1, x \in R^{n_x + 1}$
- Sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$

Loss error function: $L(\hat{y}, y) = -y \log \hat{y} + (1 - y) \log(1 - \hat{y}) = \frac{1}{2} (\hat{y} - y)^2$

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

GRADIENT DESCENT

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

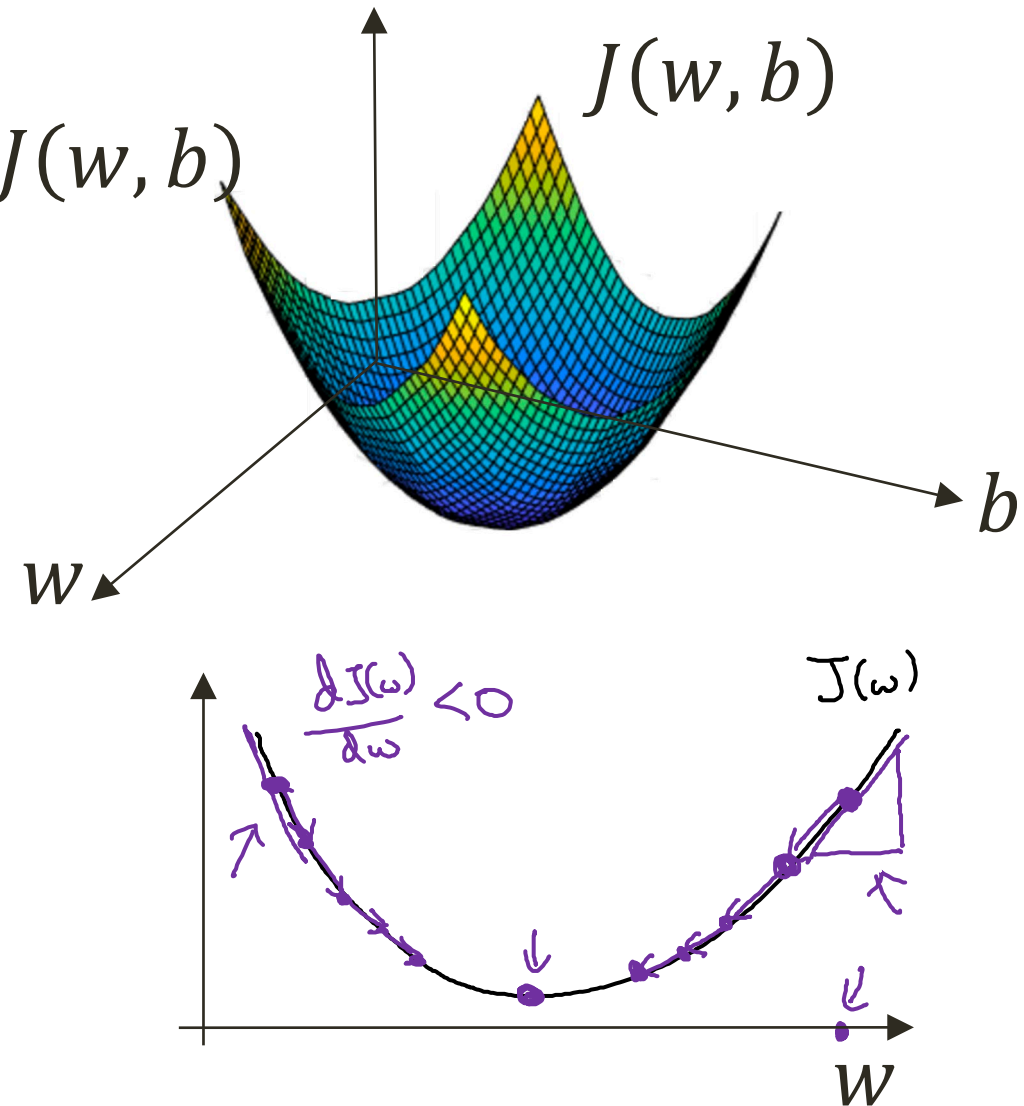
Repeat {

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

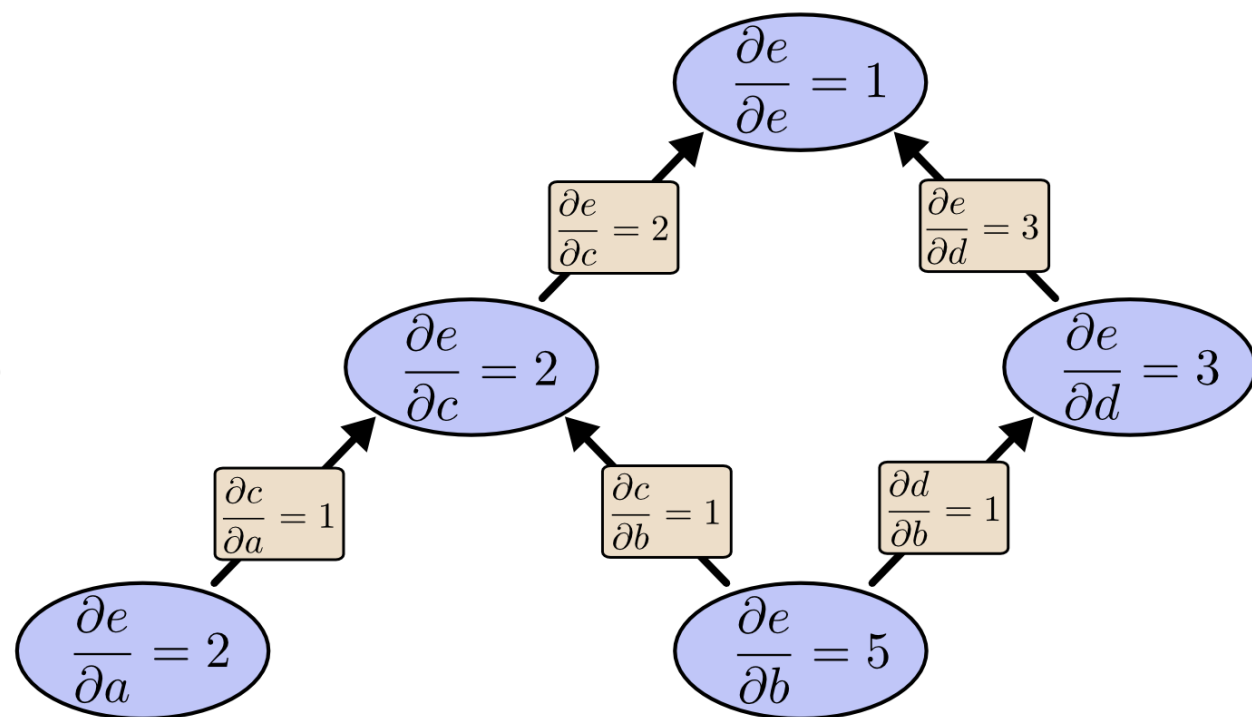
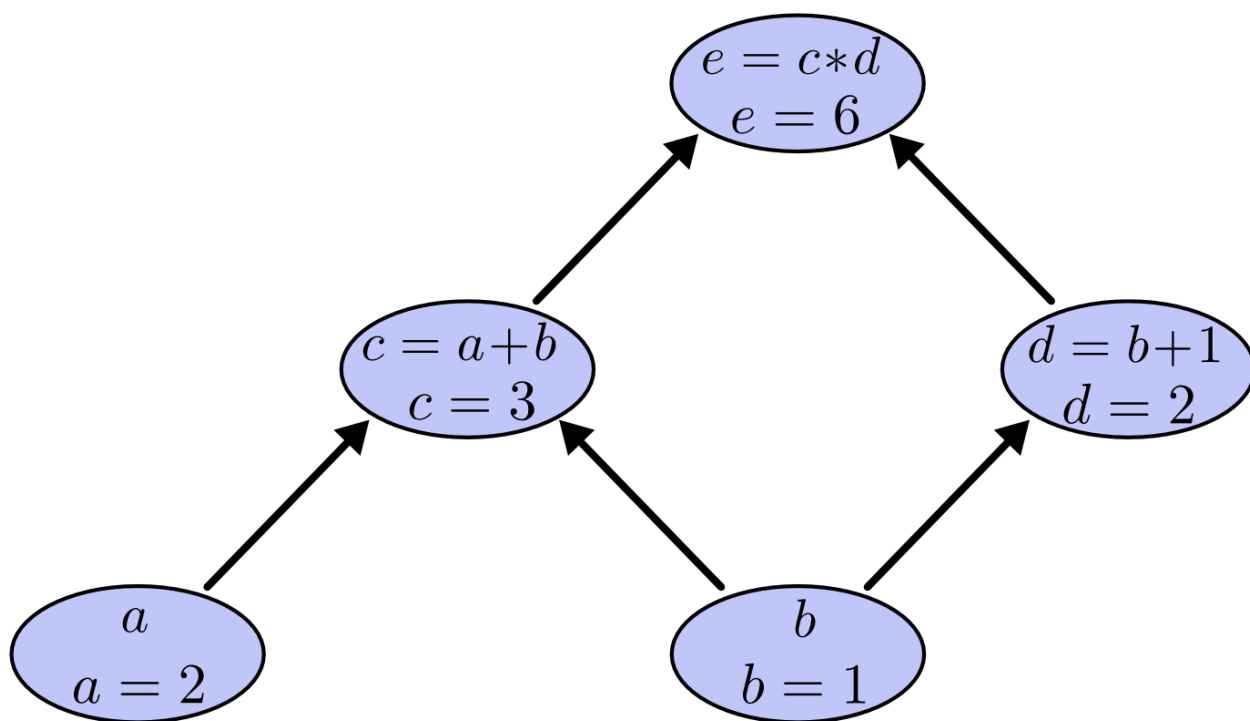
$$b := b - \alpha \frac{dJ(w, b)}{db}$$

}

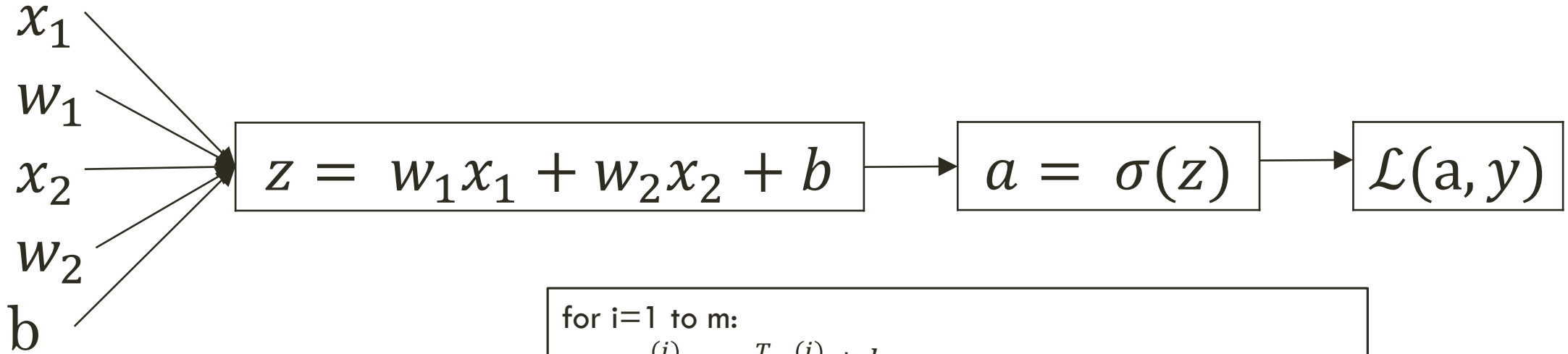
$$\begin{aligned} \alpha &= \text{learning rate} \\ dw &= \frac{dJ(w, b)}{dw} \\ db &= \frac{dJ(w, b)}{db} \end{aligned}$$



COMPUTATION GRAPH



LOGISTIC REGRESSION DERIVATIVES



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

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

$$dw = x \cdot dz$$

$$db = dz$$

for i=1 to m:

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

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

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

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

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

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

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

$$J /= m; dw_1 /= m; dw_2 /= m; db /= m$$

VECTORIZATION

Non vectorized: For loop → Slow

Vectorized: Tensor calculations → Fast

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

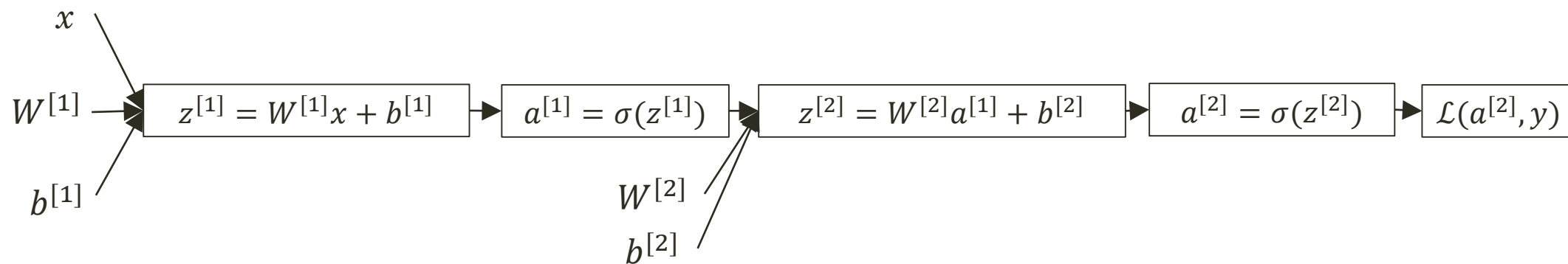
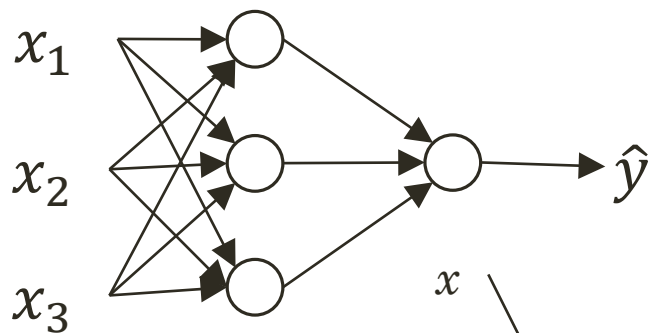
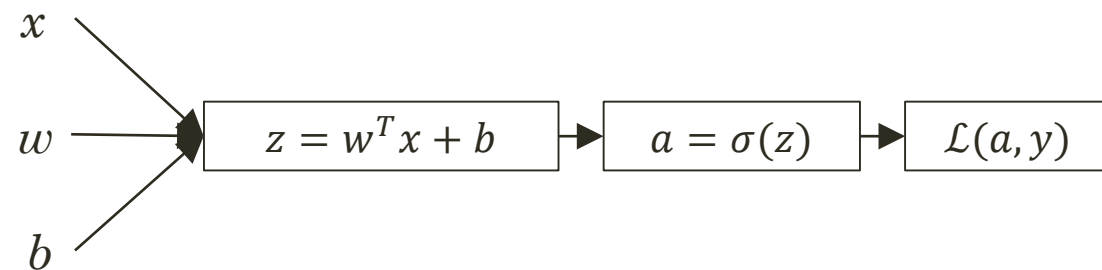
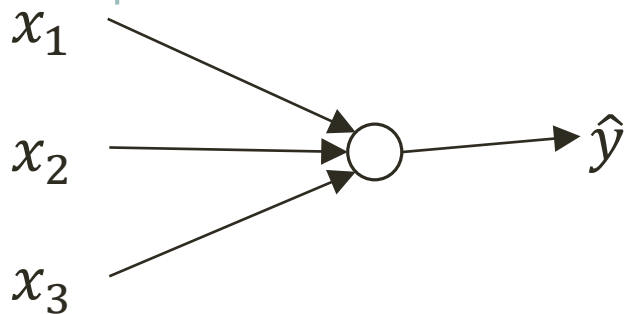
Python
Broadcasting



Vectorizing logistic regression:

- $Z = [z^{(1)} \dots z^{(n+1)}] = w^T X + [b \dots b] = [w^T x^{(1)} + b \dots w^T x^{(2)} + b] = \text{np.dot}(w.T, X) + b$
- $A = [a^{(1)} \dots a^{(m)}] = \sigma(Z)$
- $dZ^{[L]} = A^{[L]} - Y = [a^{(1)} - y^{(1)} \dots a^{(m)} - y^{(m)}]; dZ^{[l]} = w^{[l+1]T} dZ^{[l+1]} * g^{[l]'}(Z^{[l]})$
- $db^{[l]} = \frac{1}{n} \text{np.sum}(dZ^{[l]}, \text{axis} = 1, \text{keepdims} = \text{True}), dw^{[l]} = \frac{1}{n} dZ^{[l]} A^{[l-1]T}$
- $w := w - \alpha dw; b := b - \alpha db$

NEURAL NETWORK



NEURAL NETWORK REPRESENTATION

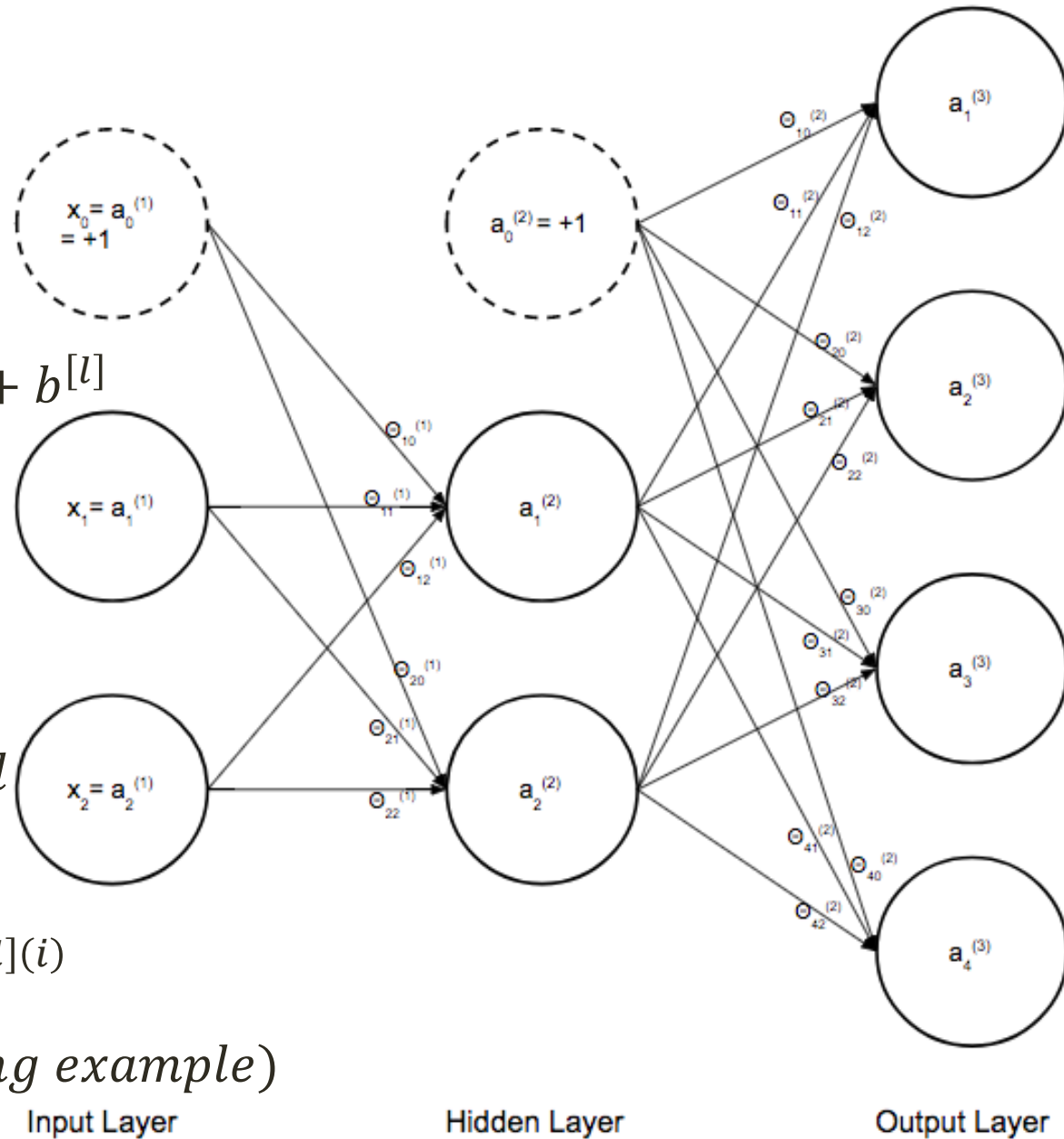
$$z^{[l]} = \begin{bmatrix} \vdots \\ z_i^{[l]} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ w_i^{[l]T} X + b_i^{[l]} \\ \vdots \end{bmatrix} = W^{[l]}x + b^{[l]}$$

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

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

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

PS: $Z^{[1]}, A^{[1]} \in (\#hidden\ units, \#training\ example)$



MORE ON VECTORIZED IMPLEMENTATION

$$z^{1} = \underbrace{w^{[1]} x^{(1)}}_{\text{purple}} + \cancel{b^{[1]}}_{\text{red}}, \quad z^{[1](2)} = \underbrace{w^{[1]} x^{(2)}}_{\text{green}} + \cancel{b^{[1]}}_{\text{red}}, \quad z^{[1](3)} = \underbrace{w^{[1]} x^{(3)}}_{\text{yellow}} + \cancel{b^{[1]}}_{\text{red}}$$

$$w^{[1]} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{bmatrix} \quad w^{[1]} x^{(1)} = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} \quad w^{[1]} x^{(2)} = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} \quad w^{[1]} x^{(3)} = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

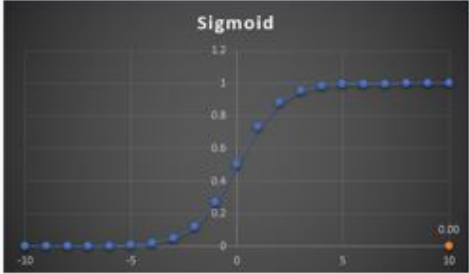
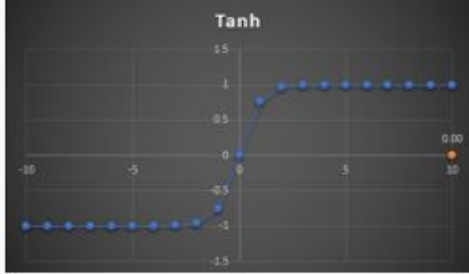
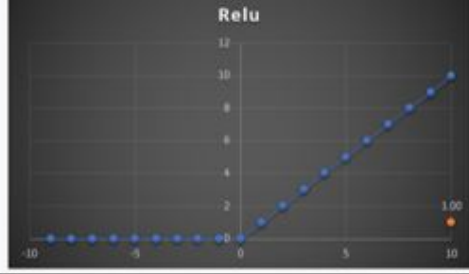

$$w^{[1]} \begin{bmatrix} | & | & | & \dots \\ x^{(1)} & x^{(2)} & x^{(3)} & \dots \\ | & | & | & \dots \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet & \dots \\ \bullet & \bullet & \bullet & \dots \\ \bullet & \bullet & \bullet & \dots \\ \bullet & \bullet & \bullet & \dots \end{bmatrix} = \begin{bmatrix} | & | & | & \dots \\ z^{1} & z^{[1](2)} & z^{[1](3)} & \dots \\ | & | & | & \dots \end{bmatrix} = z^{[1]}$$

$\hat{X} \quad \quad \quad w^{[1]} \hat{X} = z^{[1]}$

$z^{[1]} = w^{[1]} X + b^{[1]}$

(Note: In the diagram, the bias term $b^{[1]}$ is shown as a vector of zeros, indicated by red arrows and the word "zero".)

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 (<u>relu</u>)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
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

$$\begin{aligned}Z^{[1]} &= W^{[1]}X + b^{[1]} \\A^{[1]} &= g^{[1]}(Z^{[1]}) \\Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\A^{[2]} &= g^{[2]}(Z^{[2]}) \\&\vdots \\A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y}\end{aligned}$$

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

WEIGHT INITIALIZATION AND DIMENSIONALITY

Zero initialization \rightarrow Activations will be the same

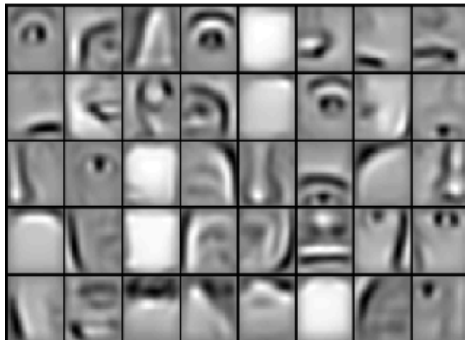
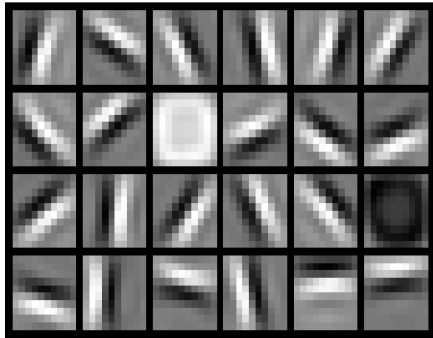
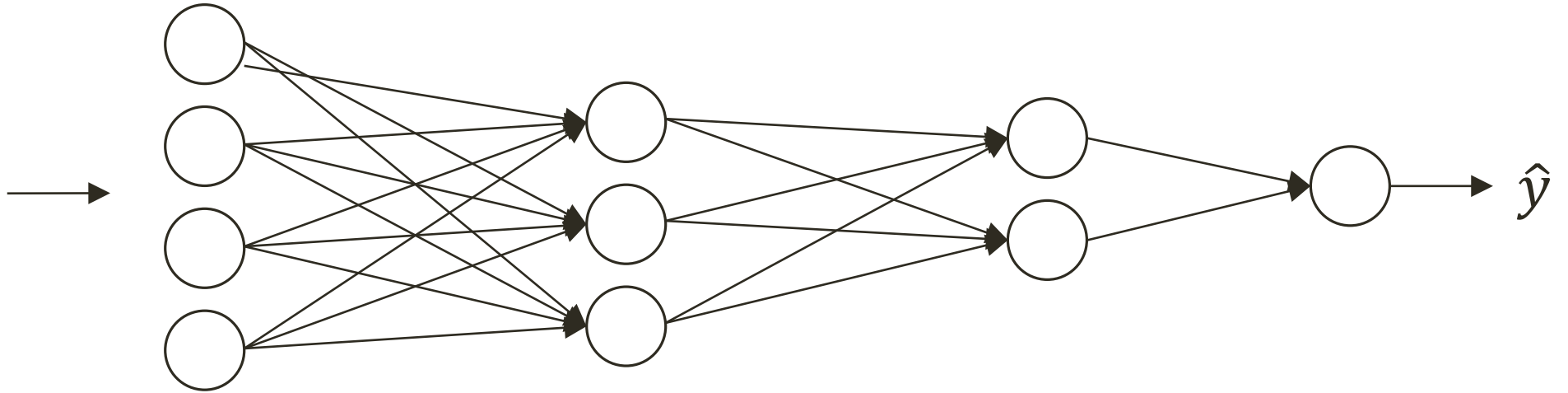
Random initialization \rightarrow **Good**

PS: b can be initialized to zero

Notations:

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

DEEP REPRESENTATION



HYPERPARAMETERS

Parameters: W, b

Hyperparameters: *Learning rate α , #iterations, #hidden layer L ,
#hidden units, choice of activation functions*

PS: Applying deep learning is a very empirical process

