Init.: $J=0, dw_1=0, dw_2=0, db=0$

For $i=1:m$

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

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

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

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

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$ } $n=2$

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

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

$J /= m$

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

The value of dw is cumulative in the code (i.e. 1 dw variable)

$$dw_1 = \frac{dJ}{dw_1}$$

$w_1 := w_1 - \alpha\, dw_1$

$w_2 := w_2 - \alpha\, dw_2$

$b := b - \alpha\, db$

Vectorisation techniques to get rid of loops

Python and vectorisation

Vectorisation

$z = w^T x + b$; $w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$, $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$, $w \in \mathbb{R}^{n_x}$, $x \in \mathbb{R}^{n_x}$

Non-vectorised:

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

Vectorised:

```
z = np.dot(w,x) + b
```

($w^T x$)

Much faster!

Both GPU & CPU have SIMP (single instruction multiple data)

NN programming guideline:
- Whenever possible, avoid explicit for-loops