# — Python And Vectorization —

Explicit for loops takes up a considerable amt of extra time. Instead built in functions are highly efficient and faster

## * Vectorize log. Reg.

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

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} \cdots & x^{(m)} \end{bmatrix}$$

$$Z = w^T X + \begin{bmatrix} b & b \cdots b \end{bmatrix}$$
$$= \begin{bmatrix} (w^T x^{(1)} + b) & (w^T x^{(2)} + b) \end{bmatrix} \cdots (w^T x^{(3)} + b$$

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

Even tho 'b' is just a no. here, python expands it as a $(1 \times m)$ vector. This is called 'Broadcasting'

$$A = \begin{bmatrix} a^{(1)} & a^{(2)} \cdots & a^{(m)} \end{bmatrix} = \sigma(Z)$$

## * Vectorize Gradient

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

$A = [a^{(1)}\ a^{(2)} \dots a^{(m)}]$

$Y = [y^{(1)}\ y^{(2)} \dots y^{(m)}]$

$\rightarrow dZ = A - Y = [dz^{(1)}\ dz^{(2)} \dots dz^{(m)}]$

$= [a^{(1)} - y^{(1)}\ a^{(2)} - y^{(2)} \dots a^{(m)} - y^{(m)}]$

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} np.sum(dz)$

$dw = \frac{1}{m} \cdot X \cdot dz$

Updating,

$w = w - \alpha\, dw$

$b = b - \alpha\, db$