vectorization

$\boxed{\text{getting rid of for loops in calculations}}$

↳ effectuer des operations mathematiques sur l'ensemble des données

example

$$z = w^T x + b \qquad w, x \in \mathbb{R}^n$$

non vectorized implementation

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

vectorized implementation

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

example 2

$$u = Av$$

$$u_i = \sum_j A_{ij} \cdot v_j$$

non vectorized implementation

```
u = np.zeros ((n,1))
for i in range (n):
    for j in range (n)
        u[i] += A[i,j] * v[j]
```

vectorized implementation

```
u = np.dot (A, v)
```

example 3 : vectors and matrix valued functions

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \qquad u = \begin{bmatrix} e^{v_1} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

non vectorized

```
u = np.zeros((n,1))
for i in range (n)
    u[i] = math.exp (v[i])
```

vectorized

```
u = np.exp (v)
```

$ab - (a+c) + bc$

$b(a+c) - (a+c)$

python | numpy vectors

a = np.random.randn(5)

a.shape = (5,) : rank 1 arrays

~~[illegible]~~

$a = [0,5;\ -0,2;\ 0,95;\ 0,8;\ 0,7]$

a.T

$\downarrow$

$[0,5;\ -0,2;\ 0,95;\ 0,8;\ 0,7]$

pas de transpose

np.dot(a, a.T) $\in \mathbb{R}$

a = np.random.randn(5,1)

a.shape = (5,1)

a.T.shape = (1,5)

$-\ R_{?}(0,9)$

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 5 & 10 \end{bmatrix}$$

$(3,4) + \ldots = (3,4)$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 4 & 2 \end{bmatrix}$$

$ab + ac \Rightarrow b-c = a(b+c)$

$(a,1)$

$$- \sum_{i=1}^{m} y^{(i)} \log\left(a^{[2](i)}\right)$$

$$Y = [\, y^{(1)}, \ldots, y^{(m)} \,]$$

$$A^{[2]} = A_2 = [\, A^{[2](1)}, \ldots, A^{[2](m)} \,]$$

__first method__

```
logprobs = np.multiply(np.log(A2, Y)
cost = -np.sum(logprobs)
```

__second method__

```
cost = -(np.dot(np.log(A2), Y)) .squeeze()
```

~~+ np.dot(np.log(1-A2), (1-Y)))~~

intuition about derivatives

$f(a) = 3a$

$a = 2 \Rightarrow f(2) = 6$

$a = 2.001 \quad f(a) = 6.003$

$\frac{df(a)}{da} = 3$: slope

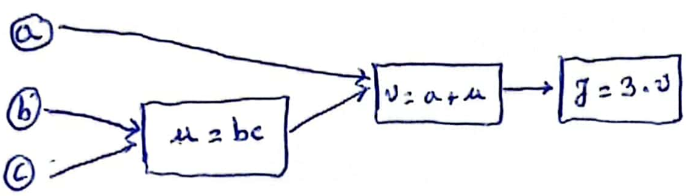
Computation graphs

$J(a,b,c) = 3(a + bc)$

$u = bc$

$v = a + u$

$J = 3.v$



derivatives with a computation graph

$\frac{dJ}{dv}$? $\quad J = 3.v$

$$\frac{dJ}{dv} = 3$$

$\frac{\partial J}{\partial a}$? $\quad \frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{\partial v}{\partial a} = 3.1 = 3$

$\frac{\partial J}{\partial u} = \frac{dJ}{dv} \cdot \frac{\partial v}{\partial u} = 3.1 = 3$

$\frac{\partial J}{\partial b} = \frac{dJ}{dv} \cdot \frac{\partial v}{\partial u} \cdot \frac{\partial u}{\partial b} = 3.1.c = c$

$\frac{\partial J}{\partial c} = \frac{dJ}{dv} \cdot \frac{\partial v}{\partial u} \cdot \frac{\partial u}{\partial c} = 3.1.b = b$

Broadcasting in numpy

	apples	beef	Eggs	potatoes
carb	56.0	0.0	4.4	68.0
protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

calculate % of calories from carb, prot, fat
can we do it without explicit for loop

```
# sum of the columns
sums = A.sum(axis=0)
```

percentages

[59, 239, 155.4, 76.9]

```
percentages = 100 x A / sums.reshape(1,4)
```

A is a (3,4) ~~vector~~ matrix
sums is a (1,4) vector

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + 100 \Leftrightarrow \begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix}$$

$$[1, 2, 3, 4] + 100 \Leftrightarrow [1, 2, 3, 4] + [100, 100, 100, 100]$$

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + [100 \;\; 200 \;\; 300] \Leftrightarrow \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}$$

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100\\200\end{bmatrix} \Leftrightarrow \begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix}$$

In general

(m,n) matrix $+ \; - \; * \; /$ $(1,n) \Leftrightarrow (m,n)$ $+ \; - \; * \; /$ (m,n)

(m,n) matrix $+ \; - \; * \; /$ $(m,1) \Leftrightarrow (m,n)$ $+ \; - \; * \; /$ (m,n)