BÀI ÔN TẬP KIẾN THỨC TOÁN VÀ LẬP TRÌNH

Phần 2: Các phép toán

<table>
<tr><th colspan="2">Mô tả Toán học</th><th>Mã lệnh Python</th></tr>
<tr><td rowspan="3">Các phép toán với vector</td><td>Phép cộng 2 vector $x, y \in R^n$:
$$f(x,y): R^n + R^n \rightarrow R^n$$
$$f(x,y) = x + y, \forall x, y \in R^n$$
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} + \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} x_0 + y_0 \\ x_1 + x_1 \\ \vdots \\ x_{n-1} + y_{n-1} \end{bmatrix}$$
Ví dụ:
$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 5 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$</td><td>Cộng 2 vector (ndarray 1 chiều) trong Python
<pre>x = np.array([1,2,3,4])
y = np.array([5,3,2,1])
z = x + y
print(z)
print(z.ndim)
print(z.shape)</pre></td></tr>
<tr><td>Phép nhân 1 đại lượng vô hướng với 1 vector: $\lambda \in R, x \in R^n$:
$$f(\lambda, x): R^n \rightarrow R^n$$
$$\lambda \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} \lambda x_0 \\ \lambda x_1 \\ \vdots \\ \lambda x_{n-1} \end{bmatrix}$$</td><td>Phép nhân đại lượng vô hướng với vector trong Python
<pre>x = np.array([1,2,3,4])
a = 15
z = a*x
print(z)
print(z.ndim)
print(z.shape)</pre></td></tr>
<tr><td>Phép tích vô hướng 2 vector (thường gọi là dot product), $y \in R^n$:
$$f(x,y): R^n \cdot R^n \rightarrow R$$
$$f(x,y) = \sum_{i=1}^{n} x_i \times y_i$$
Ví dụ:</td><td>Tích vô hướng 2 vector (ndarray 1 chiều) trong Python
<pre>x = np.array([1,2,3,4])
y = np.array([5,3,2,1])
z = np.dot(x,y) #hoặc z = x.dot(y)
print(z)
print(z.ndim)
print(z.shape)</pre></td></tr>
</table>

	$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} \cdot \begin{bmatrix}5\\3\\2\\1\end{bmatrix} = 1.5 + 2.3 + 3.2 + 4.1 = 21$	

	Phép tích hữu hướng (cross product) 2 vector (lưu ý số chiều vector trong trường hợp này ≤ 3).
	$\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix} \otimes \begin{bmatrix}y_0\\y_1\\y_2\end{bmatrix} = \begin{bmatrix}\begin{vmatrix}x_1 & y_1\\x_2 & y_2\end{vmatrix}\\\begin{vmatrix}x_0 & y_0\\x_2 & y_2\end{vmatrix}\\\begin{vmatrix}x_0 & y_0\\x_1 & y_1\end{vmatrix}\end{bmatrix}$
		Tích hữu hướng (cross product) 2 vector trong Python
		`x = np.array([1,2,3])`
		`y = np.array([3,2,1])`
		`z = np.cross(x, y)`
		`print(z)`
		`print(z.ndim)`
		`print(z.shape)`

	Phép nhân các phần tử có vị trí tương ứng (elementwise multiplication) 2 vector:
	$f(x,y): R^n \circ R^n \rightarrow R^n$
	$x^{\circ}y = [x_0.y_0;\ x_1.y_1;\ \cdots;\ x_{n-1}.y_{n-1}]$
	Lưu ý: phép nhân này được sử dụng rất nhiều trong lĩnh vực Học máy và Học sâu.
		Elementwise multiplication
		`x = np.array([1,2,3,4])`
		`y = np.array([5,3,2,1])`
		`z = x*y  #hoặc z = np.multiply(x, y)`
		`print(z)`
		`print(z.ndim)`
		`print(z.shape)`

Các phép toán với ma trận	***Phép cộng 2 ma trận*** $X, Y \in R^{m\times n}$:
	$f(X,Y): R^{m\times n} + R^{m\times n} \rightarrow R^{m\times n}$
	$\begin{bmatrix}x_{11} & \cdots & x_{1n}\\\vdots & \ddots & \vdots\\x_{m1} & \cdots & x_{mn}\end{bmatrix} + \begin{bmatrix}y_{11} & \cdots & y_{1n}\\\vdots & \ddots & \vdots\\y_{m1} & \cdots & y_{mn}\end{bmatrix} = \begin{bmatrix}x_{11}+y_{11} & \cdots & x_{1n}+y_{1n}\\\vdots & \ddots & \vdots\\x_{m1}+y_{m1} & \cdots & x_{mn}+y_{mn}\end{bmatrix}$
		`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`Y = np.array([6,5,4,3,2,1]).reshape((2,3))`
		
		`Z = X + Y`
		`print(Z)`
		`print(Z.ndim, '; ', Z.shape)`

	Phép nhân ma trận với 1 đại lượng vô hướng $\lambda \in R, X \in R^{m\times n}$:
	$f(\lambda, X): R^{m\times n} \rightarrow R^{m\times n}$
	$\lambda\begin{bmatrix}x_{11} & \cdots & x_{1n}\\\vdots & \ddots & \vdots\\x_{m1} & \cdots & x_{mn}\end{bmatrix} = \begin{bmatrix}\lambda x_{11} & \cdots & \lambda x_{1n}\\\vdots & \ddots & \vdots\\\lambda x_{m1} & \cdots & \lambda x_{mn}\end{bmatrix}$
		`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`a = 2`
		`Z = a*X`
		`print(Z)`
		`print(Z.ndim, '; ', Z.shape)`

	Phép chuyển vị ma trận (transpose): $X \in R^{m\times n}$
	$f(X): R^{m\times n} \rightarrow R^{n\times m}$
	$X \rightarrow X^T$
		`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`Z = X.T`
		`print(Z)`
		`print(Z.ndim, '; ', Z.shape)`

	Phép nhân 2 ma trận $X \in R^{m\times n}, Y \in R^{n\times k}$:
	$f(X,Y): R^{m\times n} \times R^{n\times k} \rightarrow R^{m\times k}$
	$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}\begin{bmatrix} y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nk} \end{bmatrix} = \sum_{i=1}^{n} a_{ti}.b_{ij}, \forall t = \overline{1,m}, \forall j = \overline{1,k}$
		`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`Y = np.array([6,5,4,3,2,1]).reshape((3,2))`
		`Z = np.dot(X,Y)`
		`print(Z)`
		`print(Z.ndim, '; ', Z.shape)`

	Phép nhân các phần tử có vị trí tương ứng (elementwise multiplication hay còn gọi là ***Hadamard product***) của 2 ma trận:
	$f(X,Y): R^{m\times n} \circ R^{m\times n} \rightarrow R^{m\times n}$
	$\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} \circ \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mn} \end{bmatrix} = \begin{bmatrix} x_{11}y_{11} & \cdots & y_{1k} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nk} \end{bmatrix}$
	Lưu ý: phép nhân này được sử dụng rất nhiều trong lĩnh vực Học máy và Học sâu.
		`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`Y = np.array([6,5,4,3,2,1]).reshape((2,3))`
		`Z = np.multiply(X,Y) #hoặc Z = X*Y`
		`print(Z)`
		`print(Z.ndim, '; ', Z.shape)`

	Tạo vector cột dạng ma trận đặc biệt $R^{m\times 1}$	`x = np.array([1,2,3,4,5]).reshape((5,1))`
		`print(x)`
		`print(x.ndim, '; ', x.shape)`

	Lấy ra 1 vector hàng từ ma trận X và chuyển thành ma trận đặc biệt dạng $R^{1\times n}$	`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`m = X.shape[0]`
		`n = X.shape[1]`
		`print('m = ', m, '; n = ', n)`
		`x = X[0, :]`
		`print(x)`
		`print(x.ndim, '; ', x.shape)`
		`x = x.reshape((1,n))`
		`print(x)`
		`print(x.ndim, '; ', x.shape)`

	Lấy ra 1 vector cột từ ma trận X và chuyển thành ma trận đặc biệt dạng $R^{m\times 1}$	`X = np.array([1,2,3,4,5,6]).reshape((2,3))`
		`m = X.shape[0]`
		`n = X.shape[1]`
		`print('m = ', m, '; n = ', n)`
		`x = X[:,1]`
		`print(x)`
		`print(x.ndim, '; ', x.shape)`
		`x = x.reshape((m,1))`
		`print(x)`
		`print(x.ndim, '; ', x.shape)`

<table>
<tr><td rowspan="2"></td><td>Xếp thêm 1 vector cột vào đầu ma trận: $x \in R^m$ hay $x \in R^{m\times 1}, X \in R^{m\times n}$,
$$X = [x, X]$$</td><td>

```
X =
np.array([1,2,3,4,5,6,7,8,9,10,11,12]).reshape((4,3))
x = np.array([1,1,1,1]).reshape((4,1))
X = np.column_stack([x,X])
print(X)
print(X.ndim, '; ', X.shape)
```

</td></tr>
<tr><td>Tách 1 ma trận data ban đầu gồm n cột thành 1 ma trận con X chứa các n-1 cột đầu tiên và 1 vector cột y lấy ra từ cột cuối cùng của data.</td><td>

```
data = np.array([[10,22,13,1],
                 [9,6,5,0],
                 [8,12,4,1],
                 [6,5,7,0]])
X = data[:,:-1]
y = data[:,-1].reshape((data.shape[0], 1))
print(X)
print(X.ndim, '; ', X.shape)
print(y)
print(y.ndim, '; ', y.shape)
```

</td></tr>
<tr><td rowspan="3">Các phép toán với tensor hạng cao (≥ 3)</td><td colspan="2">Ghi chú: Do cấu trúc Tensor được dùng chủ yếu trong các bài toán Học sâu (deep learning), nên ở đây chỉ trình bày một số thao tác cơ bản như khởi tạo Tensor hạng cao ($n \geq 3$), và các thao tác thêm vào/lấy ra các phần tử của Tensor để phục vụ mục đích cho bài tập đọc kho ảnh, lấy dữ liệu ảnh để nhận dạng ở mức độ cơ bản.
Các phép toán với Tensor sẽ được giới thiệu đầy đủ ở học phần Học sâu.</td></tr>
<tr><td>Tạo 1 tensor $T \in R^{m\times n\times k}$ với m = 2, n = 2 và k = 3, các giá trị được sinh ngẫu nhiên.</td><td>

```
import numpy as np
m=2
n=2
k=3
T = np.random.randint(low=-5,
high=5,size=m*n*k).reshape(k,m,n)
print(T)
```

</td></tr>
<tr><td>Tạo 1 tensor $T2 \in R^{m\times n\times k\times q}$ chứa 2 tensor $t1, t2 \in R^{m\times n\times k}$, biết t1 và t2 được sinh ngẫu nhiên</td><td>

```
import numpy as np
m=2
n=2
k=3
t1 = np.random.randint(low=-5,
high=5,size=m*n*k).reshape(k,m,n)
t2 = np.random.randint(low=-5,
high=5,size=m*n*k).reshape(k,m,n)
t3 = np.random.randint(low=-5,
high=5,size=m*n*k).reshape(k,m,n)
t4 = np.random.randint(low=-5,
high=5,size=m*n*k).reshape(k,m,n)
```

</td></tr>
</table>

		`T = np.stack((t1, t2, t3, t4), axis=-1)`
		`print(T.shape)`
		
		`x = T[:,:,:,0]`
		`print(x.shape)`
		
		`m1 = x[0,:,:]`
		`print(m1)`
		`print(m1.shape)`