線性代數沒有明確定義，但重要的Python張量操作 !!

```python
import numpy as np
import tensorflow as tf
import torch as tc
```

1.張量 X 資料結構間互相轉換

X --->	list	numpy	pytorch	tensorflow
list	-	np.array(X)	tc.tensor(X)	tf.convert_to_tensor(X)
numpy	X.tolist()	-	tc.tensor(X)	tf.convert_to_tensor(X)
pytorch	X.tolist()	X.numpy()	-	-
tensorflow	-	X.numpy()	-	-

2.常見張量形狀操作!!

A , B 表示 tensor 名稱

名稱	numpy	torch	tensorflow
取得張量形狀	A.shape	A.shape / A.size()	A.shape
取得張量大小	A.size	A.numel()	tf.size(A)
轉置	np.transpose(A,[1,0])	A.permute(1,0)	tf.transpose(A, perm=[1,0])
攤平至1維	A.flatten()	tc.flatten(A)	-
重訂形狀	A.reshape(m,n)	A.view(m,n)	tf.reshape(A,[m,n])
合併張量	np.concatenate([A,B],axis=1)	tc.cat([A,B],dim=1)	tf.concat([A, B], axis=1)
堆積張量	np.stack([A,B],axis=1)	tc.stack([A,B],dim=1)	tf.stack([A,B], axis=1)
壓縮維度 (去1)	np.squeeze(A,axis=0)	tc.squeeze(A,dim=0)	tf.squeeze(A,axis=1)
提升維度 (加1)	np.unsqueeze(A,axis=0)	tc.unsqueeze(A,dim=0)	tf.unsqueeze(A,axis=1)
內存連續化	np.ascontiguousarray(A)	A.contiguous()	-

轉置:

$$
A^{T}:= transpose\bigg(A\equiv(A_{ijk}),[2,0,1]\bigg) = (A_{kij}) \\

A_{.shape} = (I,J,K) \quad A^{T}_{.shape} = (K,I,J) \\

==========備註 : \color{red}{需再使用"內存連續化"才能讓空間儲存連續} =========== \\
$$

$$A^T := transpose\Big(A \equiv (A_{ijk}), [2,0,1]\Big) = (A_{kij})$$

$$A_{.shape} = (I,J,K) \quad A^T_{.shape} = (K,I,)$$

$$==========\text{備註}：\color{red}{\text{需再使用"內存連續化"才能讓空間儲存連續}}==========$$

$$A^T := transpose\Big(A \equiv (A_{ijk}), [2,0,1]\Big) = (A_{kij})$$

$$A_{.shape} = (I,J,K) \quad A^T_{.shape} = (K,I,J)$$

$$==========\text{備註}：\color{red}{\text{需再使用"內存連續化"才能讓空間儲存連續}}==========$$

攤平至1維:

$$B := flatten(A) \Longrightarrow B_{(iJK+jK+k)} := A_{ijk}$$

重訂形狀:

$$D := reshape(A, [I', J', K', T']) \Longrightarrow D_{i'j'k't'} = C_{(i'J'K'T'+j'K'T'+k'T'+t')} = B_{(iJK+jK+k)} = A_{ijk}$$

$$==========\text{備註}：\color{red}{\text{存的值會保持原本順序}}==========$$

$$B := flatten(A), \quad C := flatten(D), \quad IJK = I'J'K'T', \quad N := (iJK + jK + k)$$

$$i' = \left\lfloor \frac{N}{J'K'T'} \right\rfloor, j' = \left\lfloor \frac{N - i'J'K'T'}{K'T'} \right\rfloor, k' = \left\lfloor \frac{N - i'J'K'T' - j'K'T'}{T'} \right\rfloor,$$

合併張量:

$$A_{.shape} = (I, \color{red}{J_1}, K) \quad B_{.shape} = (I, \color{red}{J_2}, K)$$

$$C := concat([A,B], dim = 1)$$

$$C_{.shape} = (I, \color{red}{J_1 + J_2}, K)$$

堆積張量:

$$
\begin{gathered}
A_{.shape} = (I, J, K) \quad B_{.shape} = (I, J, K) \\
C := stack([A, B], dim = 1) \\
D := stack([A, B, A], dim = 0) \\
E := stack([A, A, A, A], dim = 3) \\
C_{.shape} = (I, 2, J, K) \\
D_{.shape} = (3, I, J, K) \\
E_{.shape} = (I, J, K, 4)
\end{gathered}
$$

壓縮維度:

$$
\begin{gathered}
A_{.shape} = (1, I, J, 1, K) \\
B := squeeze(A, dim = 0) \\
C := squeeze(A, dim = [0, 3]) \\
B_{.shape} = (I, J, 1, K) \\
C_{.shape} = (I, J, K)
\end{gathered}
$$

提升維度:

$$
\begin{gathered}
A_{.shape} = (I, J, K) \\
B := unsqueeze(A, dim = 0) \\
C := unsqueeze(A, dim = 2) \\
B_{.shape} = (1, I, J, K) \\
C_{.shape} = (I, J, 1, K)
\end{gathered}
$$

---

3. BroadCasting (定義不同形狀的張量如何運算 !!)

- Numpy : https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html
- Pytorch : https://pytorch.org/docs/stable/notes/broadcasting.html
- Tensorflow: https://www.tensorflow.org/xla/broadcasting

$$
\begin{gathered}
A_{.shape} = (8, 3, 4, 1, 5), B_{.shape} = (3, 1, 4, 5) \\
C := A \oplus B \Longrightarrow C_{.shape} = (8, 3, 4, 4, 5)
\end{gathered}
$$