CHAPTER III
Tensors and Tensor Arithmetic

PHOK Ponna and NEANG Pheak

Institute of Technology of Cambodia
Department of Applied Mathematics and Statistics (AMS)

2022–2023

Contents

1. What are Tensors
2. Tensors in Python
3. Tensor Arithmetic
4. Tensor Product

Contents

1 What are Tensors

2 Tensors in Python

3 Tensor Arithmetic

4 Tensor Product

Definition 1

A **tensor** is a generalization of vectors and matrices and is easily understood as a multidimensional array.
In the general case, an array of numbers arranged on a regular grid with a variable number of axes is known as a tensor.
A vector is a one-dimensional or first order tensor and a matrix is a two-dimensional or secondorder tensor. Tensor notation is much like matrix notation with a capital letter representing a tensor and lowercase letters with subscript integers representing scalar values within the tensor.

Example 1

Below defines a $3 \times 3 \times 3$ three-dimensional tensor T with dimensions index as $t_{i,j,k}$.

$$T = \begin{pmatrix} t_{1,1,1} & t_{1,2,1} & t_{1,3,1} \\ t_{2,1,1} & t_{2,2,1} & t_{2,3,1} \\ t_{3,1,1} & t_{3,2,1} & t_{3,3,1} \end{pmatrix}, \begin{pmatrix} t_{1,1,2} & t_{1,2,2} & t_{1,3,2} \\ t_{2,1,2} & t_{2,2,2} & t_{2,3,2} \\ t_{3,1,2} & t_{3,2,2} & t_{3,3,2} \end{pmatrix}, \begin{pmatrix} t_{1,1,3} & t_{1,2,3} & t_{1,3,3} \\ t_{2,1,3} & t_{2,2,3} & t_{2,3,3} \\ t_{3,1,3} & t_{3,2,3} & t_{3,3,3} \end{pmatrix}$$

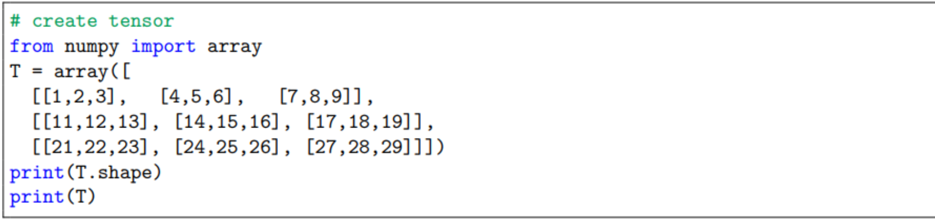
Remark 1

Many of the operations that can be performed with scalars, vectors, and matrices can bereformulated to be performed with tensors. As a tool, tensors and tensor algebra is widely used in the fields of physics and engineering. Some operations in machine learning such as the training and operation of deep learning models can be described in terms of tensors.

Contents

1. What are Tensors
2. Tensors in Python
3. Tensor Arithmetic
4. Tensor Product

Like vectors and matrices, tensors can be represented in Python using the N-dimensional array(ndarray). A tensor can be defined in-line to the constructor of array() as a list of lists. The example below defines a $3 \times 3 \times 3$ tensor as a NumPy ndarray. Three dimensions is easier to wrap your head around. Here, we first define rows, then a list of rows stacked as columns, thena list of columns stacked as levels in a cube.



```
# create tensor
from numpy import array
T = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
print(T.shape)
print(T)
```

Running the example first prints the shape of the tensor, then the values of the tensor itself. You can see that, at least in three-dimensions, the tensor is printed as a series of matrices, one for each layer. For this 3D tensor, axis 0 specifies the level (like height), axis 1 specifies the column, and axis 2 specifies the row.

Contents

1. What are Tensors
2. Tensors in Python
3. Tensor Arithmetic
4. Tensor Product

As with matrices, we can perform element-wise arithmetic between tensors. In this section, we will work through the four main arithmetic operations.

Tensor Addition

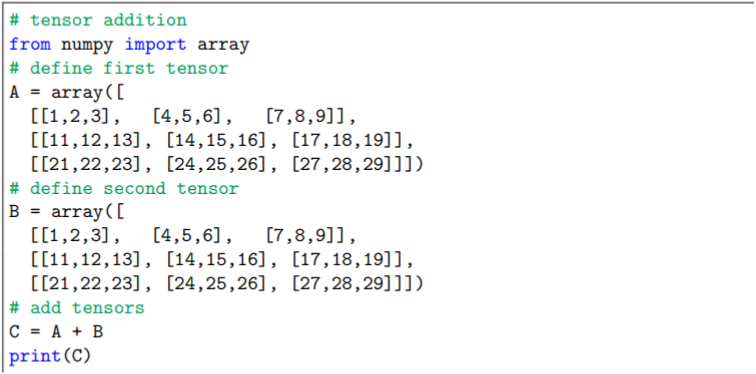
The element-wise addition of two tensors with the same dimensions results in a new tensor with the same dimensions where each scalar value is the element-wise addition of the scalars in the parent tensors.

$$A = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} \end{pmatrix}, \begin{pmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1,1} & b_{1,2,1} & b_{1,3,1} \\ b_{2,1,1} & b_{2,2,1} & b_{2,3,1} \end{pmatrix}, \begin{pmatrix} b_{1,1,2} & b_{1,2,2} & b_{1,3,2} \\ b_{2,1,2} & b_{2,2,2} & b_{2,3,2} \end{pmatrix}$$

$$C = A + B$$

In NumPy, we can add tensors directly by adding arrays.



```
# tensor addition
from numpy import array
# define first tensor
A = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# define second tensor
B = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# add tensors
C = A + B
print(C)
```

Running the example prints the addition of the two parent tensors.

Tensor Subtraction

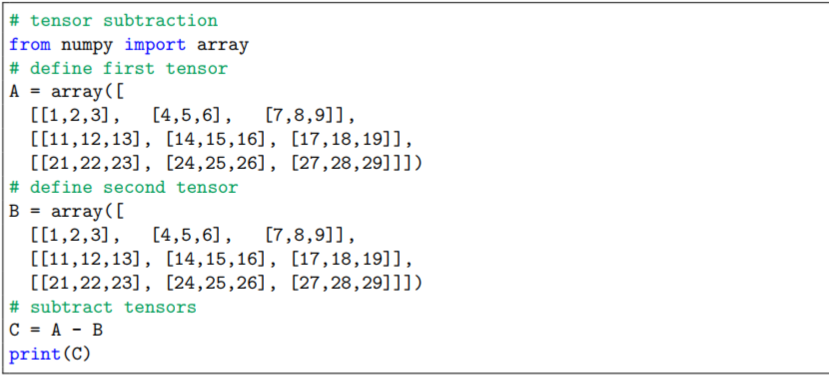
The element-wise subtraction of one tensor from another tensor with the same dimensions results in a new tensor with the same dimensions where each scalar value is the element-wisesubtraction of the scalars in the parent tensors.

$$A = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} \end{pmatrix}, \begin{pmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1,1} & b_{1,2,1} & b_{1,3,1} \\ b_{2,1,1} & b_{2,2,1} & b_{2,3,1} \end{pmatrix}, \begin{pmatrix} b_{1,1,2} & b_{1,2,2} & b_{1,3,2} \\ b_{2,1,2} & b_{2,2,2} & b_{2,3,2} \end{pmatrix}$$

$$C = A - B$$

In NumPy, we can subtract tensors directly by subtracting arrays.



```
# tensor subtraction
from numpy import array
# define first tensor
A = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# define second tensor
B = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# subtract tensors
C = A - B
print(C)
```

Running the example prints the result of subtracting the first tensor from the second.

Tensor Hadamard Product

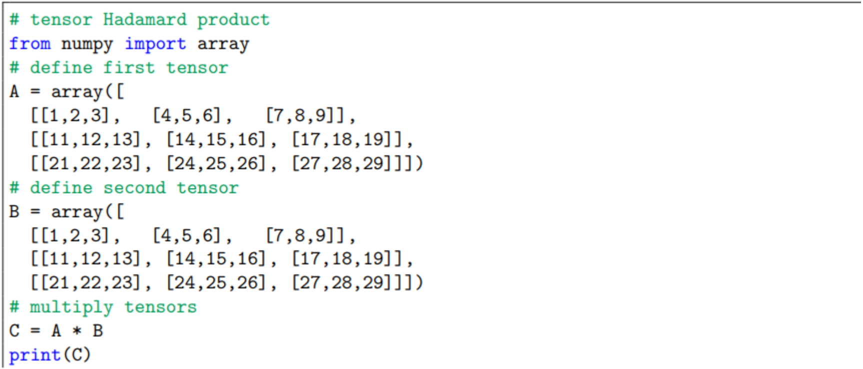
The element-wise multiplication of one tensor with another tensor with the same dimensions results in a new tensor with the same dimensions where each scalar value is the element-wise multiplication of the scalars in the parent tensors. As with matrices, the operation is referred to as the Hadamard Product to differentiate it from tensor multiplication. Here, we will use the $\circ$ operator to indicate the Hadamard product operation between tensors.

$$A = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} \end{pmatrix}, \begin{pmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1,1} & b_{1,2,1} & b_{1,3,1} \\ b_{2,1,1} & b_{2,2,1} & b_{2,3,1} \end{pmatrix}, \begin{pmatrix} b_{1,1,2} & b_{1,2,2} & b_{1,3,2} \\ b_{2,1,2} & b_{2,2,2} & b_{2,3,2} \end{pmatrix}$$

$$C = A \circ B$$

In NumPy, we can multiply tensors directly by multiplying arrays.



```
# tensor Hadamard product
from numpy import array
# define first tensor
A = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# define second tensor
B = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# multiply tensors
C = A * B
print(C)
```

Running the example prints the result of multiplying the tensors.

Tensor Division

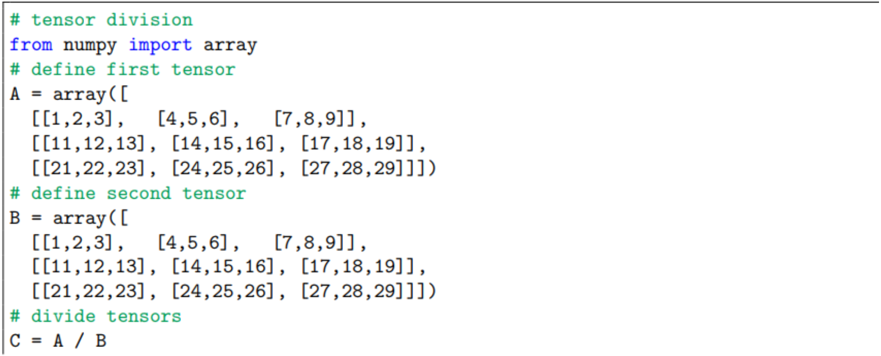
The element-wise division of one tensor with another tensor with the same dimensions results in a new tensor with the same dimensions where each scalar value is the element-wise division of the scalars in the parent tensors.

$$A = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} \end{pmatrix}, \begin{pmatrix} a_{1,1,2} & a_{1,2,2} & a_{1,3,2} \\ a_{2,1,2} & a_{2,2,2} & a_{2,3,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1,1} & b_{1,2,1} & b_{1,3,1} \\ b_{2,1,1} & b_{2,2,1} & b_{2,3,1} \end{pmatrix}, \begin{pmatrix} b_{1,1,2} & b_{1,2,2} & b_{1,3,2} \\ b_{2,1,2} & b_{2,2,2} & b_{2,3,2} \end{pmatrix}$$

$$C = \frac{A}{B}$$

In NumPy, we can divide tensors directly by dividing arrays.



```
# tensor division
from numpy import array
# define first tensor
A = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# define second tensor
B = array([
  [[1,2,3],   [4,5,6],   [7,8,9]],
  [[11,12,13], [14,15,16], [17,18,19]],
  [[21,22,23], [24,25,26], [27,28,29]]])
# divide tensors
C = A / B
```

Running the example prints the result of dividing the tensors.

Contents

1. What are Tensors
2. Tensors in Python
3. Tensor Arithmetic
4. Tensor Product

The tensor product operator is often denoted as a circle with a small x in the middle. We will denote it here as $\otimes$. Given a tensor A with q dimensions and tensor B with r dimensions, the product of these tensors will be a new tensor with the order of $q + r$ or, said another way, $q + r$ dimensions. The tensor product is not limited to tensors, but can also be performed on matricesand vectors, which can be a good place to practice in order to develop the intuition for higherdimensions. Let's take a look at the tensor product for vectors.

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix}$$

$$b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$C = a \otimes b$$

$$C = \begin{pmatrix} a_1 \times \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ a_2 \times \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_1 \times b_1 & a_1 \times b_2 \\ a_2 \times b_1 & a_2 \times b_2 \end{pmatrix}$$

Let's take a look at the tensor product for matrices.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

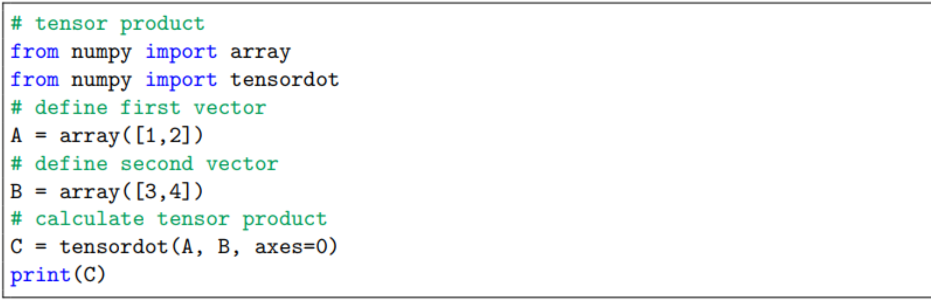
$$C = A \otimes B$$

$$C = \begin{pmatrix} a_{1,1} \times \begin{pmatrix} b_{1,1}, b_{1,2} \\ b_{2,1}, b_{2,2} \end{pmatrix} & a_{1,2} \times \begin{pmatrix} b_{1,1}, b_{1,2} \\ b_{2,1}, b_{2,2} \end{pmatrix} \\ a_{2,1} \times \begin{pmatrix} b_{1,1}, b_{1,2} \\ b_{2,1}, b_{2,2} \end{pmatrix} & a_{2,2} \times \begin{pmatrix} b_{1,1}, b_{1,2} \\ b_{2,1}, b_{2,2} \end{pmatrix} \end{pmatrix}$$

Or, unrolled:

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} & a_{1,1} \times b_{1,2} & a_{1,2} \times b_{1,1} & a_{1,2} \times b_{1,2} \\ a_{1,1} \times b_{2,1} & a_{1,1} \times b_{2,2} & a_{1,2} \times b_{2,1} & a_{1,2} \times b_{2,2} \\ a_{2,1} \times b_{1,1} & a_{2,1} \times b_{1,2} & a_{2,2} \times b_{1,1} & a_{2,2} \times b_{1,2} \\ a_{2,1} \times b_{2,1} & a_{2,1} \times b_{2,2} & a_{2,2} \times b_{2,1} & a_{2,2} \times b_{2,2} \end{pmatrix}$$

The tensor product can be implemented in NumPy using the tensordot() function. The function takes as arguments the two tensors to be multiplied and the axis on which to sum the products over, called the sum reduction. To calculate the tensor product, also called the tensor dot product in NumPy, the axis must be set to 0. In the example below, we define two order-1 tensors (vectors) with and calculate the tensor product.



```
# tensor product
from numpy import array
from numpy import tensordot
# define first vector
A = array([1,2])
# define second vector
B = array([3,4])
# calculate tensor product
C = tensordot(A, B, axes=0)
print(C)
```

Running the example prints the result of the tensor product. The result is an order-2 tensor(matrix) with the lengths 2×2.