Basic operations

Puteaux, Fall/Winter 2020-2021

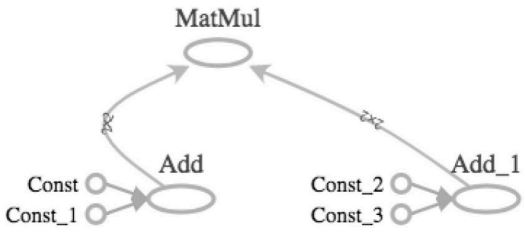
```
################################
##                            ##
##  Deep Learning in Python   ##
##                            ##
################################
```

§2 Introduction to TensorFlow in Python

§2.1 Introduction to TensorFlow

§2.1.2 Basic operations

1. What is a TensorFlow operation?



2. Code of applying the addition operator:

[1]:
```
#Import constant and add from tensorflow
from tensorflow import constant, add

# Define 0-dimensional tensors
A0 = constant([1])
B0 = constant([2])

A0, B0
```

[1]:
```
(<tf.Tensor: shape=(1,), dtype=int32, numpy=array([1], dtype=int32)>,
 <tf.Tensor: shape=(1,), dtype=int32, numpy=array([2], dtype=int32)>)
```

[2]:
```
# Define 1-dimensional tensors
A1 = constant([1, 2])
```

```
B1 = constant([3, 4])

A1, B1
```

[2]:
```
(<tf.Tensor: shape=(2,), dtype=int32, numpy=array([1, 2], dtype=int32)>,
 <tf.Tensor: shape=(2,), dtype=int32, numpy=array([3, 4], dtype=int32)>)
```

[3]:
```
# Define 2-dimensional tensors
A2 = constant([[1, 2], [3, 4]])
B2 = constant([[5, 6], [7, 8]])

A2, B2
```

[3]:
```
(<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
 array([[1, 2],
        [3, 4]], dtype=int32)>,
 <tf.Tensor: shape=(2, 2), dtype=int32, numpy=
 array([[5, 6],
        [7, 8]], dtype=int32)>)
```

[4]:
```
# Perform tensor addition with add()
C0 = add(A0, B0)
C1 = add(A1, B1)
C2 = add(A2, B2)

C0, C1, C2
```

[4]:
```
(<tf.Tensor: shape=(1,), dtype=int32, numpy=array([3], dtype=int32)>,
 <tf.Tensor: shape=(2,), dtype=int32, numpy=array([4, 6], dtype=int32)>,
 <tf.Tensor: shape=(2, 2), dtype=int32, numpy=
 array([[ 6,  8],
        [10, 12]], dtype=int32)>)
```

3. How to perform tensor addition?

- The `add()` operation performs **element-wise addition** with two tensors.
- Element-wise addition requires both tensors to have the same shape:
  - scalar addition:

    $1 + 2 = 3$

  - vector addition:

    $[1, 2] + [3, 4] = [4, 6]$

  - matrix addition:

    $$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

- The `add()` operator is overloaded.

4. How to perform multiplication in TensorFlow?

- **Element-wise multiplication** performed using `multiply()` operation.
- The tensors multiplied must have the same shape:
  - e.g., $[1, 2, 3]$ and $[3, 4, 5]$ or $[1, 2]$ and $[3, 4]$
- **Matrix multiplication** is performed with `matmul()` operator.
  - The `matmul(A, B)` operation multiplies `A` by `B`.
  - The number of columns of `A` must equal the number of rows of `B`.

5. Code of applying the multiplication operators:

[5]:
```
# Import operators from tensorflow
from tensorflow import ones, matmul, multiply

# Define tensors
A0 = ones(1)
A31 = ones([3, 1])
A34 = ones([3, 4])
A43 = ones([4, 3])

A0, A31, A34, A43
```

[5]:
```
(<tf.Tensor: shape=(1,), dtype=float32, numpy=array([1.], dtype=float32)>,
 <tf.Tensor: shape=(3, 1), dtype=float32, numpy=
 array([[1.],
        [1.],
        [1.]], dtype=float32)>,
 <tf.Tensor: shape=(3, 4), dtype=float32, numpy=
 array([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]], dtype=float32)>,
 <tf.Tensor: shape=(4, 3), dtype=float32, numpy=
 array([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]], dtype=float32)>)
```

[6]:
```
A0_A0 = multiply(A0, A0)
A31_A31 = multiply(A31, A31)
A34_A34 = multiply(A34, A34)

A0_A0, A31_A31, A34_A34
```

[6]:
```
(<tf.Tensor: shape=(1,), dtype=float32, numpy=array([1.], dtype=float32)>,
 <tf.Tensor: shape=(3, 1), dtype=float32, numpy=
 array([[1.],
        [1.],
        [1.]], dtype=float32)>,
 <tf.Tensor: shape=(3, 4), dtype=float32, numpy=
 array([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]], dtype=float32)>)
```

[7]:
```
A43_A34 = matmul(A43, A34)

A43_A34
```

[7]:
```
<tf.Tensor: shape=(4, 4), dtype=float32, numpy=
array([[3., 3., 3., 3.],
       [3., 3., 3., 3.],
       [3., 3., 3., 3.],
       [3., 3., 3., 3.]], dtype=float32)>
```

6. How to sum over tensor dimensions?

- The `reduce_sum()` operator sums over the dimensions of a:
  - `tensorreduce_sum(A)` sums over all the dimensions of `A`
  - `reduce_sum(A, i)` sums over the dimension `i`

7. Code of summing over tensor dimensions:

[8]:
```
# Import operations from tensorflow
from tensorflow import ones, reduce_sum

# Define a 2x3x4 tensor of ones
A = ones([2, 3, 4])

A
```

[8]:
```
<tf.Tensor: shape=(2, 3, 4), dtype=float32, numpy=
array([[[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]],

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]], dtype=float32)>
```

[9]:
```
# Sum over all dimensions
B = reduce_sum(A)
```

```
B
```

[9]:
```
<tf.Tensor: shape=(), dtype=float32, numpy=24.0>
```

[10]:
```
# Sum over dimensions 0, 1, and 2
B0 = reduce_sum(A, 0)
B1 = reduce_sum(A, 1)
B2 = reduce_sum(A, 2)

B0, B1, B2
```

[10]:
```
(<tf.Tensor: shape=(3, 4), dtype=float32, numpy=
 array([[2., 2., 2., 2.],
        [2., 2., 2., 2.],
        [2., 2., 2., 2.]], dtype=float32)>,
 <tf.Tensor: shape=(2, 4), dtype=float32, numpy=
 array([[3., 3., 3., 3.],
        [3., 3., 3., 3.]], dtype=float32)>,
 <tf.Tensor: shape=(2, 3), dtype=float32, numpy=
 array([[4., 4., 4.],
        [4., 4., 4.]], dtype=float32)>)
```

8. Practice exercises for basic operations:

▶ Package pre-loading:

[11]:
```
from tensorflow import constant, ones_like, multiply
```

▶ Element-wise multiplication performing practice:

[12]:
```
# Define tensors A1 and A23 as constants
A1 = constant([1, 2, 3, 4])
A23 = constant([[1, 2, 3], [1, 6, 4]])

# Define B1 and B23 to have the correct shape
B1 = ones_like([1, 2, 3, 4])
B23 = ones_like([[1, 2, 3], [1, 6, 4]])

# Perform element-wise multiplication
C1 = multiply(A1, B1)
C23 = multiply(A23, B23)

# Print the tensors C1 and C23
print('C1: {}'.format(C1.numpy()))
print('C23: {}'.format(C23.numpy()))
```

```
C1: [1 2 3 4]
C23: [[1 2 3]
```

```
 [1 6 4]]
```

▶ **Package re-pre-loading:**

[13]:
```
from tensorflow import matmul
```

▶ **Matrix multiplication predictions practice:**

[14]:
```
# Define features, params, and bill as constants
features = constant([[2, 24], [2, 26], [2, 57], [1, 37]])
params = constant([[1000], [150]])
bill = constant([[3913], [2682], [8617], [64400]])

# Compute billpred using features and params
billpred = matmul(features, params)

# Compute and print the error
error = bill - billpred
print(error.numpy())
```

```
[[-1687]
 [-3218]
 [-1933]
 [57850]]
```

9. Practice question for summing over tensor dimensions:

- There is a matrix, `wealth`. This contains the value of the bond and stock wealth for five individuals in thousands of dollars.
- `wealth` $= \begin{bmatrix} 11 & 50 \\ 7 & 2 \\ 4 & 60 \\ 3 & 0 \\ 25 & 10 \end{bmatrix}$
- The first column corresponds to bonds, and the second corresponds to stocks. Each row gives the bond and stock wealth for a single individual. Use `wealth`, `reduce_sum()`, and `.numpy()` to determine which statements are correct about `wealth`.

  ☐ The individual in the first row has the highest total wealth (i.e., stocks + bonds).

  ☐ Combined, the 5 individuals hold $\$50,000$ in stocks.

  ☒ Combined, the 5 individuals hold $\$50,000$ in bonds.

  ☐ The individual in the second row has the lowest total wealth (i.e., stocks + bonds).

▶ **Package pre-loading:**

[15]:
```
from tensorflow import constant, reduce_sum
```

▶ **Data pre-loading:**

[16]:
```
wealth = constant([[11, 50], [7, 2], [4, 60], [3, 0], [25, 10]])
```

▶ **Question-solving method:**

[17]:
```
wealth
```

[17]:
```
<tf.Tensor: shape=(5, 2), dtype=int32, numpy=
array([[11, 50],
       [ 7,  2],
       [ 4, 60],
       [ 3,  0],
       [25, 10]], dtype=int32)>
```

[18]:
```
wealth.numpy()
```

[18]:
```
array([[11, 50],
       [ 7,  2],
       [ 4, 60],
       [ 3,  0],
       [25, 10]], dtype=int32)
```

[19]:
```
reduce_sum(wealth)
```

[19]:
```
<tf.Tensor: shape=(), dtype=int32, numpy=172>
```

[20]:
```
reduce_sum(wealth, 0)
```

[20]:
```
<tf.Tensor: shape=(2,), dtype=int32, numpy=array([ 50, 122], dtype=int32)>
```

[21]:
```
reduce_sum(wealth, 1)
```

[21]:
```
<tf.Tensor: shape=(5,), dtype=int32, numpy=array([61,  9, 64,  3, 35],
dtype=int32)>
```