Getting started with TensorFlow 2

In this tutorial we will introduce the basic concepts of TensorFlow 2. Let's start with importing the tensorflow and numpy libraries and making sure that TensorFlow version is at least 2.0.

```
import tensorflow as tf
import numpy as np
tf.__version__
```

```
'2.2.0-rc3'
```

Basics

Now let us look at a simple example of multiplying two numbers:

```
a = tf.constant(2)
b = tf.constant(10)
c = tf.multiply(a,b)
print(c)
```

```
tf.Tensor(20, shape=(), dtype=int32)
```

The variables `a`, `b`, and `c` are of the Tensor type. Tensors are a generalization of vectors and matrices to higher dimensions and are the basic building blocks of TensorFlow.

In this course you will work a lot with matrices, so let us multiply two matrices:

```
W = tf.constant([[1, 2], [3, 4]], dtype=np.float32)
x = tf.constant([[1], [0]], dtype=np.float32)
b = tf.constant([[0.1], [0.2]])
Z = tf.add(tf.matmul(W,x), b)
print(Z)
```

```
tf.Tensor(
[[1.1]
 [3.2]], shape=(2, 1), dtype=float32)
```

Note how we have specified the type of the values in the `W` and `x` matrices using `dtype=np.float32`, otherwise integer values may have been assumed.

It is very important that the shapes match. You can inspect the shape of a tensor by printing it:

```
print(W)
print(x)
```

```
tf.Tensor(
[[1. 2.]
 [3. 4.]], shape=(2, 2), dtype=float32)
tf.Tensor(
[[1.]
 [0.]], shape=(2, 1), dtype=float32)
```

The `*` and `+` operation are overloaded, however, note that `*` is overloaded with `tf.multiply()`, which performs element-wise multiplication:

```
print(W*x + b)
```

```
tf.Tensor(
[[1.1 2.1]
 [0.2 0.2]], shape=(2, 2), dtype=float32)
```

TensorFlow implements many useful functions, e.g. the sigmoid function, which can be applied to scalar values:

```
print(tf.sigmoid(0.0))
```

```
tf.Tensor(0.5, shape=(), dtype=float32)
```

or tensors:

```
Y = tf.sigmoid(Z)
print(Y)
```

```
tf.Tensor(
[[0.7502601 ]
 [0.96083426]], shape=(2, 1), dtype=float32)
```

Gradient descent

Let us now try to find the minimum of the function

$f(x) = x^2 - 8x + 16$

If we rewrite it to $(x - 4)^2$, we can immediately see that $x = 4$ will minimize f. We will use this simple example to show how TensorFlow can be used to automatically find the value of x that minimizes f.

1. Define the variable `x` and assign it an initial value 10. The argument `name='x'` is useful for printing the value of a tensor:

```
x = tf.Variable(10, name='x', trainable=True, dtype=tf.float32)
```

2. Define the function `f` that will be minimized. The function takes no arguments and returns a tensor expression:

```
def f():
  return x*x - 8*x + 16
```

3. Define the optimizer. TensorFlow implements many optimizers and ways to train Here, we will use the stochastic gradient descent, with learning rate 0.1:

```
opt = tf.keras.optimizers.SGD(learning_rate=0.1)
```

4. Run the optimizer. Here, we minimize the loss function `f` with respect to the variable `x`, and run it for 100 iterations:

```
for i in range(100):
  opt.minimize(f, var_list=[x])
print(x)
```

```
<tf.Variable 'x:0' shape=() dtype=float32, numpy=4.000001>
```

The value of `x` is very close to 4, as expected.

As next steps, you may like to learn more about custom training in TensorFlow, or visit the tutorials.