TensorFlow I

StartOnAI Deep Learning Tutorial 3

This tutorial introduces TensorFlow and goes over making a simple network

1. What is TensorFlow

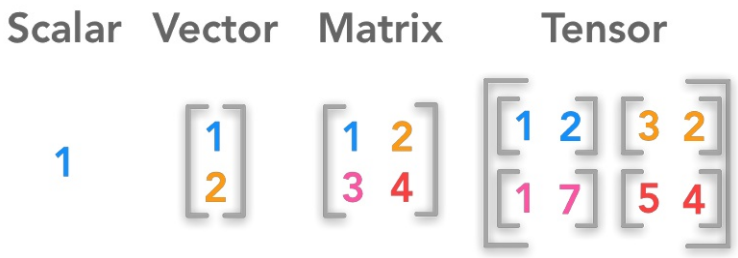
Machine learning is a daunting discipline, but frameworks such as Google's TensorFlow makes it easier to complete ML-related tasks. Compiling data, training, making predictions, and learning is all made easier through this library. It uses Python on the front-end for simplicity but executes its commands behind the scenes using C++. TensorFlow is able to create and train a variety of simple and complex neural networks. It allows for programmers to create *dataflow graphs* which are structures that display how input data is moving through the network's nodes which is important to reduce the *cost* of the network. Each connection between these nodes is known as a *tensor*, or high-dimensional array, hence the name TensorFlow. TensorFlow applications can be run locally and on google's own TensorFlow Processing Unit (TPU).

2. Why TensorFlow is the Industry Standard

There's no doubt that TensorFlow was marketed better than its competitors and being developed by Google gave it an extra edge, but there is far more to support its preference over other frameworks. TensorFlow has readable syntax which is something many developers had issues with previously in the field. It also provides greater functionality which helps carrying out complicated parallel calculations and when it comes to building more complex neural networks. This framework is a *low-level* library that allows for programmers to create their own requirements for the models which is an important necessity for larger applications of AI. For example, in an AI-powered business model accomodating a constantly changing user platform is imperative. Finally, TensorFlow provides tools, such as dataflow graphs, for developers to keep track of how different operations are affecting the network and how any other changes are being processed.

3. More About Tensors

A tensor is essentially a data container which can store N-dimensional data. From a math perspective tensors also include descriptions of the valid linear transformations between tensors, such as the cross product and dot product. Looking at a tensor from a programming perspective, thinking about it as being an object can be helpful.



The above image shows where a tensor fits in with matrices, vectors, and scalars.

```
x = np.array([[[1, 4, 7],
               [2, 5, 8],
               [3, 6, 9]],
              [[10, 40, 70],
               [20, 50, 80],
               [30, 60, 90]],
              [[100, 400, 700],
               [200, 500, 800],
               [300, 600, 900]]])
```

```
print(x)
print('Tensor rank: ' + str((x.ndim)))
```

```
[[[  1   4   7]
  [  2   5   8]
  [  3   6   9]]

 [[ 10  40  70]
  [ 20  50  80]
  [ 30  60  90]]

 [[100 400 700]
  [200 500 800]
  [300 600 900]]]
```

```
Tensor rank: 3
```

Above is an example of a 3-dimensional tensor made using NumPy. First, a NumPy array was created representing a 3D tensor and then it was processed with x.ndim. This returned an output of 3 which is the tensor's dimension.

4. Simple TensorFlow Neural Network for Handwritten Digit Recognition

Before starting this section make sure you have installed TensorFlow onto your machine. This can be done with a simple pip install, or if you want it in a virtual environment you can follow the tutorial from the following link: https://www.tensorflow.org/install/pip

```
pip install tensorflow
```

In this tutorial we will be looking at the data from the MNIST dataset which contains a large selection of handwritten digits. This is known as the 'Hello World' of image recognition. The first step will be to load and prepare the dataset

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

The `mnist` variable now holds the data which was imported from Keras's datasets. Keras is a high-level API built on top of TensorFlow and a tutorial on it can be found on the StartOnAI website. Next, `mnist.load_data()` returns two tuples. The x values are grayscale images and the y values are digit labels 0-9, which are the categories we are attempting to classify. Dividing the values by 255 is to help normalize the data to help with training.

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
```

```
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
```

This statement is setting up a *Sequential* model by taking in 4 parameters as an array. *Flatten* takes the input and converts the input to a single array. *Dense* adds another layer with the first parameter as the output size and the activation of *relu* means this layer will squish the output values in between 0 and 1. *Dropout* is an interesting method which "drops" random nodes from the network. This is important because it prevents *overfitting* which means a model trains the data too well which negatively impacts the performance of the model on new data. Finally, another dense layer is being added with an output size of 10 which represents the digits 0-9. All of these different parameters make up the Sequential network used to process the digits.

```
predictions = model(x_train[:1]).numpy()
tf.nn.softmax(predictions).numpy()
```

This is passing the data from `x_train` into the model and converting it into a NumPy array. The second line converts these into probabilities for each class.

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

Now, the *loss* is being calculated for the model. Right now the model is untrained so the loss is around 2.3 which is expected to get closer to 0 after being trained.

```
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

Compile is setting up the model for training.

```
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test,  y_test, verbose=2)
```

This trains the model for a fixed number of *epochs*. Evaluating the model allows the

developer to see how accurate the model is and what the loss is at.

```
predictions = model.predict(x_test)
print(numpy.argmax(numpy.round(predictions[0])))
```

Finally the data can be tested by using `model.predict()` which provides a prediction for the input data. Congratulations, in this tutorial you have learned about TensorFlow and its benefits, the basic structure of a neural network, and how to classify handwritten digits!

5. More Resources

https://www.tensorflow.org/tutorials
https://www.youtube.com/watch?v=KNAWp2S3w94&feature=youtu.be
https://www.youtube.com/watch?v=6g4O5UOH304
https://www.datacamp.com/community/tutorials/tensorflow-tutorial
https://medium.com/@quantumsteinke/whats-the-difference-between-a-matrix-and-a-tensor-4505fbdc576c

6. References

J., Hadrien. *Difference between a scalar, a vector, a matrix and a tensor.* 2018. *Github.* https://hadrienj.github.io/assets/images/2.1/scalar-vector-matrix-tensor.png

"TensorFlow 2 Quickstart for Beginners." TensorFlow, 1 Apr. 2020, www.tensorflow.org/tutorials/quickstart/beginner.

Yegulalp, Serdar. "What Is TensorFlow? The Machine Learning Library Explained." InfoWorld, 18 June 2019, www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.