An Introduction to Google's Machine Learning Tool TensorFlow - Part One

Dr. Frazer K. Noble

Outline

In today's lecture, we will:

- Introduce TensorFlow.
- Work through getting the tools for, and installing, TensorFlow.
- Introduce core TensorFlow concepts.
- Work though a number of examples.
- Write some programs in Python using TensorFlow.

Today, you will need the following tools[1]:

- Git [4].
- Anaconda [2].
- Visual Studio Code [7].

[1] Additional tools are required to run on a NVIDIA GPU.

Introduction I

Machine learning "teaches computers to do what humans and animals do: learn from experience" [1].

"Machine learning algorithms use computational algorithms to 'learn' information directly from data, without relying on predetermined models" [1].

The more data, the better.

Introduction II

TensorFlow is:

> "An open-source software library for Machine Intelligence" [6].

It is a "software library used for numerical computation using data flow graphs. Nodes in the graph represent mathematical operation, while the graph's edges represent multi-dimensional data arrays (tensors) communicated between them." [6].

For example:

$$C = A + B$$

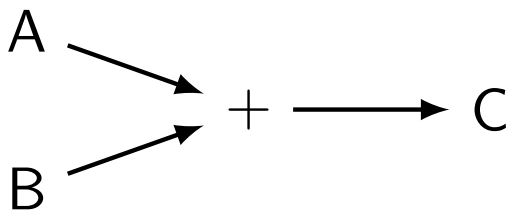

Figure: (Left) a mathematical expression, and (Right) its equivalent graph.

Getting the Necessary Tools

To get started with TensorFlow, you will need a Python distribution and an Integrated Development Environment (IDE).

We will be using Anaconda and Visual Studio Code (Code for short).

Anaconda is an "open data science platform" and a "high performance distribution of Python" [2].

Download Anaconda from: `https://www.continuum.io/downloads`.

Code is a "streamlined code editor with support for development operations like debugging, task running, and version control". It "provides the tools a developer needs for a quick code-build-debug cycle" [7].

Download Code from: `https://code.visualstudio.com/`.

Run the downloaded .exe and install Code.

Installing TensorFlow with Anaconda

Detailed instructions to install TensorFlow can be found at: `https://www.tensorflow.org/install/`.

We will be installing TensorFlow on Windows.

To install TensorFlow in an Anaconda environment:

1. Download and install Anaconda for Windows.
2. Open a command prompt.
3. Create a conda environment named "tf" with the following command:

   conda create -n tf python=3.5
4. Activate the conda environment with the following command:

   activate tf
5. To install the CPU only version of TensorFlow, use the following command:

   pip install tensorflow

Validating your TensorFlow Installation

To validate your TensorFlow installation:

1. Open a command prompt.
2. Activate your TensorFlow environment with the following command:

   activate tf

3. Open Code with the following command:

   code

4. Create a file named test.py and enter the following program:

```
import tensorflow as tf
hello = tf.constant('Hello!')
sess = tf.Session()
print(sess.run(hello))
```

5. Browse to test.py's directory and run the following command:

   python test.py

Getting Started

Detailed instructions to get started with programming in TensorFlow can be found at:
`https://www.tensorflow.org/get_started/get_started`.

It is expected you know the following:

- How to program Python.
- How arrays work.
- The basics of machine learning.

The following link will help you get started with programming in Python:
`https://www.python.org/about/gettingstarted/`.

Tensors

A tensor is the central unit of data in TensorFlow.

A tensor consists of a set of values shaped into an array of any number of dimensions. A tensor's rank is the number of dimensions.

For example:

```
[1., 2., 3.] # a rank 1 tensor; a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

TensorFlow Core Tutorial

The import statement for a TensorFlow program is as follows:

```
import tensorflow as tf
```

This gives Python access to TensorFlow's classes, methods, and symbols.

The Computational Graph I

A TensorFlow Core program consists of two sections:

1. Building the computational graph.
2. Running the computational graph.

A computational graph is a series of TensorFlow operations arranged into a graph of nodes.

A constant is a type of node.

```
A = tf.constant(value=1.0, dtype=tf.float32)
```

It takes no inputs, and outputs a value that it stores internally.

The Computational Graph II

To evaluate a node, the computational graph must be run in a session. A session "encapsulates the control and state of the TensorFlow runtime" [3].

The following code creates a Session object and invokes its run method to run the computational graph to evaluate A.

Running the computational graph as follows:

```
sess = tf.Session()
print(sess.run(A))
```

the output is:

```
[1.0]
```

Example I

We can build more complicated computations by combining Tensor nodes with operations.

Consider the following program[2]:

```
import tensorflow as tf

A = tf.constant(value=1.0, dtype=tf.float32)
B = tf.constant(value=2.0, dtype=tf.float32)

C = tf.add(A, B)

  sess = tf.Session()

print(sess.run(C))
```

Listing 1: main.py

Example II

The line

```
import tensorflow as tf
```

gives the program access to TensorFlow's classes, methods, and symbols.

The lines

```
A = tf.constant(value=1.0, dtype=tf.float32)
B = tf.constant(value=2.0, dtype=tf.float32)
```

create two constant nodes.

Example III

The line

```
C = tf.add(A, B)
```

creates a node that adds its inputs together.

The lines

```
sess = tf.Session()
print(sess.run(C))
```

create a Session objects, invokes its run method, and prints the output to the terminal.

[2] A more complete example can be found at `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_1.py`.

Example I

In the previous example, constant nodes were used, but this means that the graph outputs a constant value, which isn't particularly useful.

Consider the following program[3]:

```
import tensorflow as tf

A = tf.placeholder(dtype=tf.float32, shape=None, name='A')
B = tf.placeholder(dtype=tf.float32, shape=None, name='B')

C = tf.add(A, B)

  sess = tf.Session()

print(sess.run(C, feed_dict={A: 1.0, B: 2.0))
```

Listing 2: main.py

Example II

The line

```
import tensorflow as tf
```

gives the program access to TensorFlow's classes, methods, and symbols.

The lines

```
A = tf.placeholder(dtype=tf.float32, shape=None, name='A')
B = tf.placeholder(dtype=tf.float32, shape=None, name='B')
```

create two placeholder nodes.

Example III

The line

```
C = tf.add(A, B)
```

creates a node that adds its inputs together.

The lines

```
sess = tf.Session()
print(sess.run(C, feed_dict={A: 1.0, B: 2.0))
```

create a Session objects, invokes its run method - substituting values into the placeholders, and prints the output to the terminal.

[3] A more complete example can be found at `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_2.py`.

Activity I

Perform a complex calculation.

Given the following equation and input, write a program in Python, using TensorFlow, to determine the output of the following function.

$$y(x) = 3 * x^2 + 10 * x + 5 \text{ where, } x = 2.0 \quad (1)$$

Activity II

The expected output is: $y(x) = 37.0$

An example is available at: `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/activity_1.py`.

Example I

Consider the following program[4]

```
import tensorflow as tf

x = tf.placeholder(dtype=tf.float32, shape=
    None, name='x')
y = tf.placeholder(dtype=tf.float32, shape=
    None, name='y')

M = tf.Variable(initial_value=1.0, dtype=tf.
    float32)
c = tf.Variable(initial_value=0.1, dtype=tf.
    float32)

y_ = M *x + c

loss = tf.losses.mean_squared_error(y, y_)
training = tf.train.GradientDescentOptimizer(
    learning_rate=0.1).minimize(loss)

init = [tf.global_variables_initializer(), tf.
    local_variables_initializer()]

with tf.Session() as s:

    s.run(init)

    for i in range(0, 100, 1):
        l, _ = s.run([loss, training], feed_dict
            ={x : [1.0, 2.0, 3.0], y: [2.0,
            4.0, 6.0]})

        if i%10 == 0:
            print(l)
```

Listing 3: main.py

Example II

The line

```
import tensorflow as tf
```

gives the program access to TensorFlow's classes, methods, and symbols.

The lines

```
x = tf.placeholder(dtype=tf.float32, shape=None, name='x')
y = tf.placeholder(dtype=tf.float32, shape=None, name='y')
```

create two placeholder nodes.

Example III

The lines

```
M = tf.Variable(initial_value=1.0, dtype=tf.float32)
c = tf.Variable(initial_value=0.1, dtype=tf.float32)
```

create two trainable variables.

The line

```
y_ = M*x + c
```

defines a graph; where, a node multiplies its input and its output is added to another input.

Example IV

The lines

```
loss = tf.losses.mean_squared_error(y, y_)
training = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(loss)
```

define a loss function, which "measures how far apart the current model is from the provided data" [3], and creates an optimiser to change the graph's variables' values.

The line

```
init = [tf.global_variables_initializer(), tf.local_variables_initializer()]
```

initialises all the global and local variables.

Example V

The lines

```
with tf.Session() as s:

    s.run(init)

    for i in range(0, 100, 1):
        l, _ = s.run([loss, training], feed_dict={x : [1.0, 2.0, 3.0], y: [2.0, 4.0, 6.0]})

        if i%10 == 0:
            print(l)
```

create a Session object, invokes its run method - substituting values into placeholders, and prints out every 10^{th} iteration's loss value.

[4] A more complete example can be found at `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_3.py`.

Activity I

Fit a polynomial function to data.

Given the following data, write a program in Python, using TensorFlow, to determine a polynomial's optimum coefficients.

Table: Data

x	y
0.0	0.0
1.0	2.0
2.0	8.0
3.0	18.0

x	y
4.0	32.0
5.0	50.0
6.0	72.0
7.0	98.0

x	y
8.0	128.0
9.0	162.0

Activity II

An appropriate polynomial is

$$y = a0 * x^2 + a1 * x + a2;$$

where, $[a0, a1, a2] = [2.0, 0, 0]$.

An example is available at: `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/activity_2.py`.

A Multi-Layer Perceptron Neural Network I

A Multi-Layer Perceptron (MLP) neural network is a type of feed-forward artificial neural network, which "maps sets of input data onto a set of appropriate outputs" [5] .

"A MLP consists of multiple layers of nodes in a directed graph" [5].

Each node is a neuron with an activation functions, for example:

$$y(v_i) = tanh(v_i), \text{ or } y(v_i) = (1 + e^{-v_i})^{-1}; \tag{2}$$

where,

$$v_i = \Sigma_{i=0}^{N} (W_i \times x_i) + b_i. \tag{3}$$

A Multi-Layer Perceptron Neural Network II

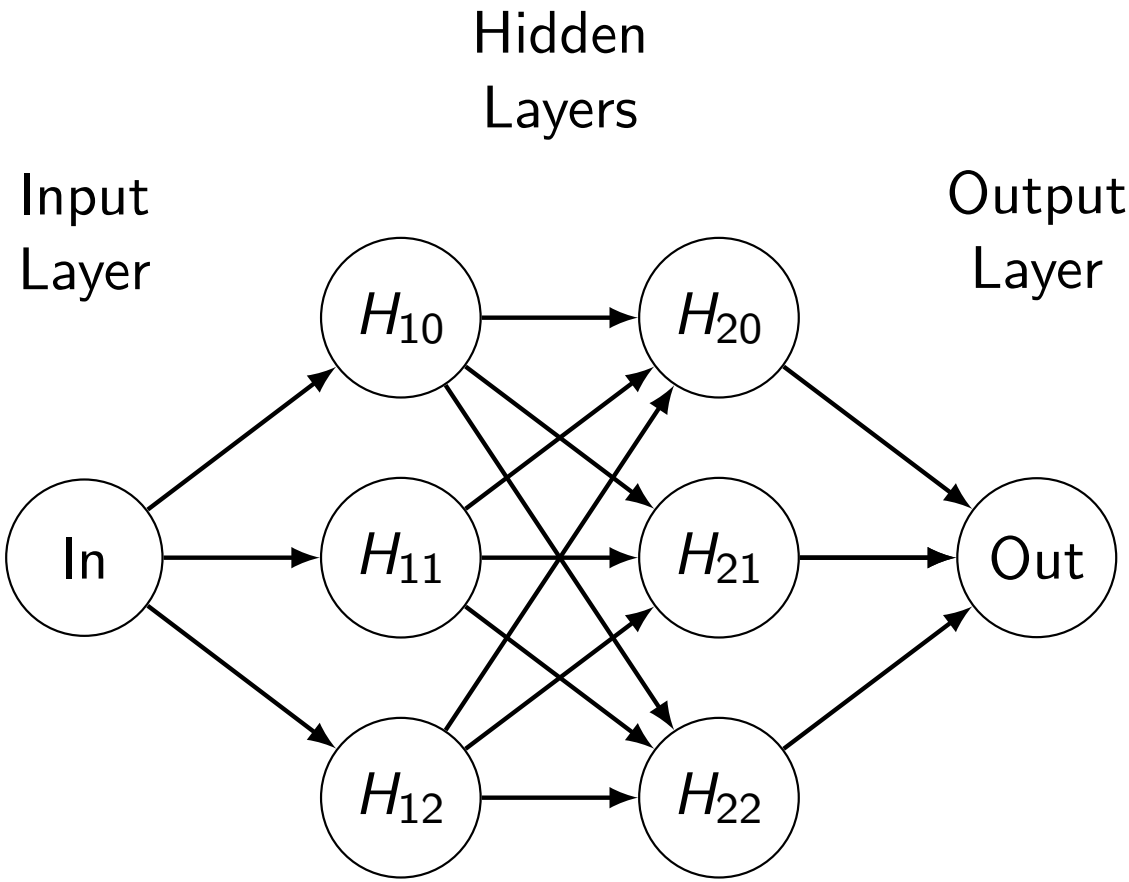

Figure: A Multi-Layer Perceptron (MLP) neural network. Here, there are two hidden layers, each with three neurons. The input and output layers have one neuron each.

Homework I

An example Multi-Layer Perceptron neural network

Consider the program available at `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_4.py`.

Do the following:

1. Read the program's source code.
2. Write comments for each line in the program.
3. Print out the commented source code.

Remember, # comments a line in Python.

TensorFlow's documentation is available at `https://www.tensorflow.org/api_docs/`.

Conclusion

In today's lecture, we have:

- Introduced TensorFlow.
- Worked through getting the tools for, and installing, TensorFlow.
- Introduced core TensorFlow concepts.
- Worked though a number of examples.
- Written some programs in Python using TensorFlow.

Questions?

Bibliography I

88174_92991v00_machine_learning_section1_ebook.pdf.
https://au.mathworks.com/content/dam/mathworks/tag-team/Objects/i/88174_92991v00_machine_learning_section1_ebook.pdf.
(Accessed on 06/07/2017).

Download anaconda now! — continuum.
https://www.continuum.io/downloads.
(Accessed on 06/01/2017).

Getting started with tensorflow — tensorflow.
https://www.tensorflow.org/get_started/get_started.
(Accessed on 06/02/2017).

Git.
https://git-scm.com/.
(Accessed on 06/01/2017).

Bibliography II


Multilayer perceptron - wikipedia.
`https://en.wikipedia.org/wiki/Multilayer_perceptron`.
(Accessed on 06/07/2017).


Tensorflow.
`https://www.tensorflow.org/`.
(Accessed on 06/01/2017).


Visual studio code - code editing. redefined.
`https://code.visualstudio.com/`.
(Accessed on 06/01/2017).