

The introduction of Tensorflow with Python

Parallel session of UCSAS 2020

Jun Jin, PhD student

Department of Statistics
University of Connecticut

September 29, 2020

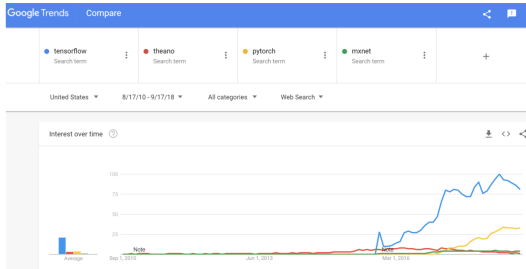


Table of contents

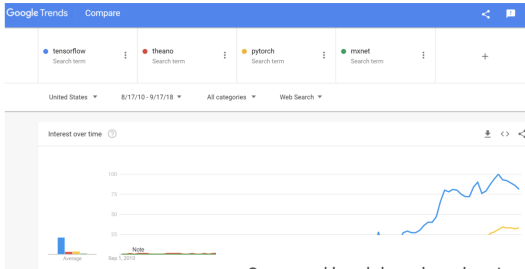
- 1 Tensorflow
- 2 Installation
- 3 Concepts and Basis
- 4 Framework
- 5 Example (Regression)
- 6 Environments
 - Text environments
 - Mathematical environments
- 7 Thanks



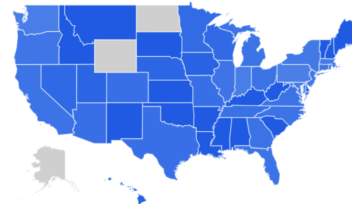
Advantage



Advantage



● tensorflow ● theano ● pytorch ● mxnet



What is Tensor?

- Tensor is the data with dimension.
- 0-d tensor: scalar

$$c = 5$$

- 1-d tensor: vector

$$c = \begin{pmatrix} 1 \\ \vdots \\ 5 \end{pmatrix}$$

- 2-d tensor: matrix

$$c = \begin{pmatrix} 1 & \cdots & 5 \\ \vdots & \ddots & \vdots \\ 5 & \cdots & 5 \end{pmatrix}$$



Why Tensor special?

- Vector and matrix operations are dominant in machine learning and deep learning.

Example

$$(GD) : \hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla E_n \left[\ell \left(y, \hat{y} \left(\hat{\beta}^{(t)} \right) \right) \right]$$

- GPU structure leads to a powerful ability to solve linear tensor operations.

How to install Tensorflow in Python

- Anaconda management (GUI): click "environment" -> choose "Not installed" -> search "tensorflow"
- Anaconda Prompt:
`conda install tensorflow`
- Pip:
`pip install tensorflow`
- Verify whether your installation is successful: (open jupyter notebook or run at .py file)

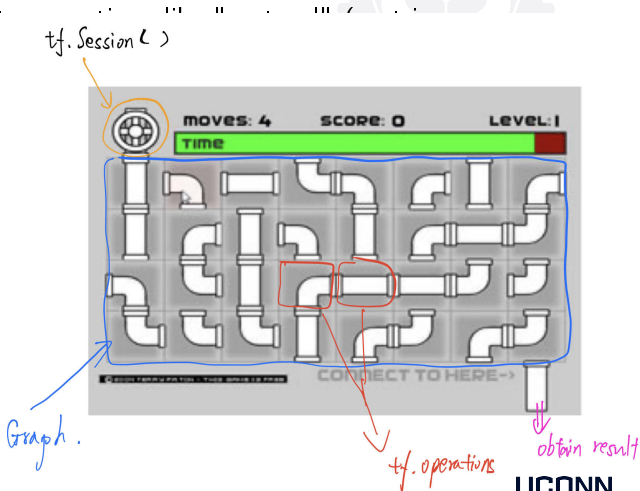
```
1 import tensorflow as tf
2 import tensorflow.compat.v1 as tfv1
3 tf.__version__
```

3 key concepts in tensorflow programming

- Operations: Data operations like "matmul" (matrix multiplication), "add".
- Graph: Build GRAPH which represents the data flow of the computation.
- Session: Run SESSION which executes the operations on the graph.

3 key concepts in tensorflow programming

- Operations: Data multiplication),
- Graph: Build computation.
- Session: Run S graph.



Basic widget: Constant

The constant can be either scalar, vector, or matrix. It is just a concept which is opposite with variable. Variable can be changed in the future by using `tf.assign()` method, while the constant can not.

```
1 s = tf.constant(2)
2 m = tf.constant([[1, 2], [3, 4]])
```

With these two node, we can conduct a GRAPH and SESSION example. Here we use "tensorboard" for GRAPH and `tf.Session()` function for SESSION.

A simplest graph and session

```
1 g = tf.Graph()
2 with g.as_default():
3     s = tf.constant(2)
4     m = tf.constant([[1, 2], [3, 4]])
5     mmul = s*m
6 g
```

Here, we have already completed a graph, to "open the faucet", we use `tf.Session()`, like:

```
1 with tfv1.Session(graph=g) as sess:
2     print(sess.run(mmul))
```

Finally, we get the result. Remark: The `tf.Session()` only exist in tensorflow v1, so we call this function under v1 platform.

Basic widget: Variable

Variables are used to hold and update parameters. To create a variable,

```
1 w = tf.Variable(tf.ones((2,2))) # 2*2 matrix with all elements as 1
```

Alert

The variable must be initialized immediately after "the data faucet is open" (i.e. `tfv1.Session()`).

```
1 with tfv1.Session() as sess:  
2     sess.run(tfv1.global_variables_initializer())  
3     print(sess.run(w))
```

Basic widget: Variable

As we mentioned earlier, variable can be assigned with a new value.

```
1 with tfv1.Session() as sess:
2     sess.run(tfv1.global_variables_initializer())
3     print(sess.run(w))
4     # Change w to a 2*2 matrix with all elements as 0
5     sess.run(w.assign(tf.zeros((2,2))))
6     print(sess.run(w))
```

Basic widget: Placeholder

We can notice that although variable is more flexible than constant, it still need a initial value. In practice, sometimes the value of a parameter is determined by the actual data. So we need the placeholder to tell the PC, there is a variable, but I don't tell you the value, I will give you the value when I open the data faucet.

```
1 import numpy as np
2 node1 = tfv1.placeholder(tf.float32, shape = [1,2])
3 node2 = tfv1.placeholder(tf.float32, shape = [1,2])
4 w_linear = tf.matmul(node1,w) + node2
5 with tfv1.Session() as sess:
6     sess.run(tfv1.global_variables_initializer())
7     print(sess.run(w))
8     print(sess.run(w_linear, feed_dict={node1:np.matrix([1.0,2.0]),
9         node2:np.matrix([1.0,2.0])}))
```

Basic Operations: add, subtract, multiply, divide, multiplication



Given x and y ,

- $x+y$ (element-wise): `tf.add(x,y)`
- $x-y$ (element-wise): `tf.subtract(x,y)`
- $x*y$ (element-wise): `tf.multiply(x,y)`
- x/y (element-wise): `tf.divide(x,y)`
- $x*y$ (matrix style): `tf.matmul(x,y)`

Basic widget: If



xxx

Basic widget: While

xxx



Session and Graph



XXX

Logo

The woodmark used in the footer can be determined by the `logo` option and specialized by the `campus` option:

```
\useoutertheme[campus=health,logo=stacked]{uconn}
```

- `campus` understands the values `health` (UConn Health), `jdh` (John Dempsey Hospital), `averypoint`, `hartford`, `stamford`, `waterbury`, and `none`.
- `logo` understands `single` (no annotation), `side`, `stacked`, and `none` (no logo, and no footline).

Passing other values is equivalent to passing no value.

Background

The faded oak leaf background image is activated by the watermark option, which defaults to none but can also be set to side or corner:

```
\useoutertheme[watermark=corner]{uconn}
```

Like the others used in this theme, the watermark image comes from the bulk logos download at UConn Brand Standard.

Symbols

The classic (black) oak leaf is encoded as a symbol `\oakleaf`, and can be used in mathematical environments, e.g.

$$\text{\oakleaf}^{\mathbb{C}} = \text{\oakleaf} \otimes_{\mathbb{R}} \mathbb{C},$$

or in text (`\text{\oakleaf}`).

An inverted-color oak leaf is encoded as `\oakleafbox` (see the slide on claims and proofs).

Blocks

Block

This is a text block, formatted as in the default template.

Alert block

This is an alert text block, also formatted by default.

More may be done in future to customize the block environments.
Suggestions are welcome!



Definitions and examples

Definition

A **definition** block is by default formatted like a text block.

Text formatting like `\bfseries` (boldface) can be called within blocks.

Example

An *example* is also a block, formatted a bit differently.

Claim and proof environments

Theorem


Claim blocks, including lemma, theorem, and corollary, use slant-shaped font (`\slshape`) to emphasize their contents.

Proof.

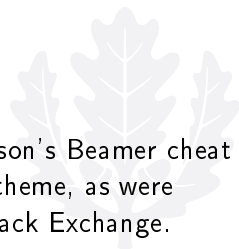
This proof block shows that the customary open square tombstone has been changed to a UConn oak leaf (white inside a black square).



Corollary

The box oak leaf can also be used inline: 

Acknowledgments



The Beamer class users guide and Thierry Masson's Beamer cheat sheet were indispensable aides to building this theme, as were several helpful discussions on the T_EX–L_AT_EX Stack Exchange.

See the UConn Brand Standards website, in particular the PowerPoint templates, for more detail on the images and layouts used here.

Please send feedback to Cory at brunson@uchc.edu.