

The introduction of TensorFlow with Applications in Sports Analytics

Parallel session of UCSAS 2021

Jun Jin, PhD candidate

Department of Statistics
University of Connecticut

October 2, 2021

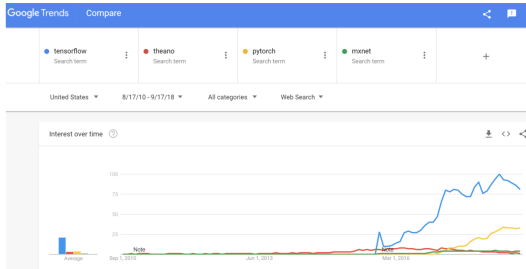


Table of contents

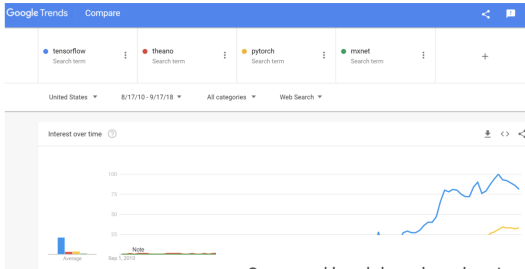
- ① Tensorflow
- ② Installation
- ③ Concepts and Basis
- ④ Framework
- ⑤ Example with simulated data (Regression)
- ⑥ Example with real-world data (Classification)
- ⑦ Thanks



Advantage

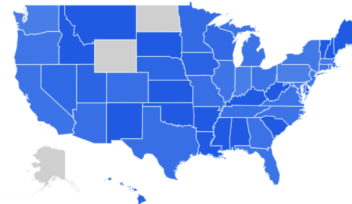


Advantage



Compared breakdown by subregion

● 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?

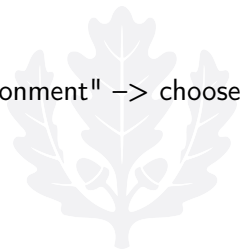
- Vector, matrix operations and gradient (derivative) are dominant in machine learning and deep learning.

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

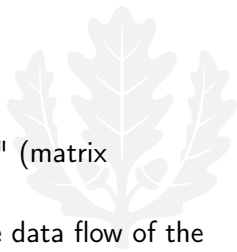


- Anaconda management (GUI): click "environment" -> choose "Not installed" -> search "tensorflow"
- Anaconda Prompt:
`conda install tensorflow`
- Pip:
`pip install tensorflow`
- Verify whether your installation is successful: (open jupyter notebook or run at .py file)

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

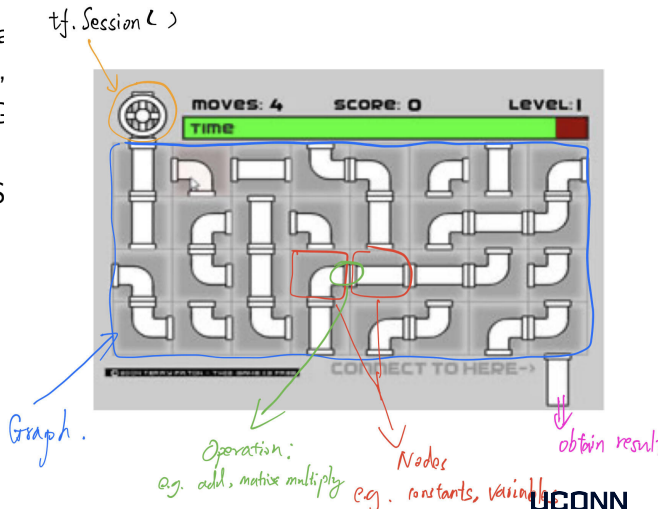
3 key concepts in tensorflow programming

- 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 (e.g. matrix multiplication),
- Graph: Build computation.
- Session: Run S graph.



Tensorflow V1 versus V2

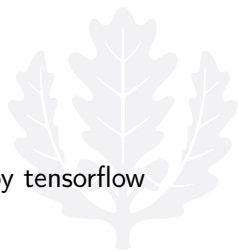
- V1: Nodes \rightarrow Operations \rightarrow Graph \rightarrow Session \rightarrow import the data
- V2: import the data \rightarrow Nodes \rightarrow Operations (eager execution)

Pros and Cons:

- V1 Pros: Integrity, a big picture of view
- V1 Cons: Lack of flexibility
- V2 Pros: Flexibility, you can always see the output step by step
- V2 Cons: Further efforts are frequently needed to pack the code up into a final package or class function

Therefore, V1 is frequently used in final delivery but V2 is good for debugging.

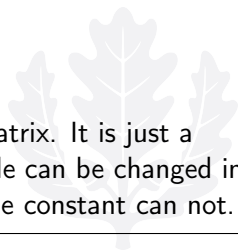
Tensorflow V1 versus V2



Good news: V1 and V2 are both accessible by tensorflow package.

- Default is V2 (`import tensorflow as tf`)
- Use `"import tensorflow.compat.v1 as tfv1"` to call the functions and methods under V1

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

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. Another important difference with the constant is that it be treated as variable in the calculation of gradient. 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

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)`
- $x < y$ (judgment): `tf.less(x,y)`
- $x > y$ (judgment): `tf.greater(x,y)`
- $x \leq y$ (judgment): `tf.less_equal(x,y)`



Gradient

One of the most powerful thing for the Tensorflow is that it can calculate the gradient with a high flexibility and high speed. The core lines are:

```
1 with tf.GradientTape() as tape:  
2     y1 = <Expression of x1, x2,... >  
3     y2 = <Expression of y1>  
4     tape.gradient(y2,[x1,x2,...])
```

It returns the gradients of y2 with respect to x1, x2,... separately.

Alert

The gradient operation can only return value when it is with respect to a trainable variable.

Gradient



```
1 x = tf.Variable(3.0)
2 with tf.GradientTape() as tape:
3     y = x**2
4     dy_dx = tape.gradient(y, x)
5     # dy = 2x * dx
6 with tfv1.Session() as sess:
7     sess.run(tfv1.global_variables_initializer())
8     print(sess.run(dy_dx))
```

Basic Logic: If

The core line is:

```
1 tf.cond(tf_statement ,A,B)
```

It implement a function that if `tf_statement` is true, then run A, otherwise, run B. By the way, if we want to choose from options more than 2, we can use `tf.switch_case()` function, that is not included in this session.

Example

```
1 t1 = tf.constant(1)
2 t2 = tf.constant(2)
3 def f1(): return t1+t2
4 def f2(): return t1-t2
5 res = tf.cond(t1<t2, f1, f2)
6 with tfv1.Session() as sess:
7     print(sess.run(res))
```

It will return 1+2 as 1 is indeed less than 2.

Basic Logic: While

The core line is:

```
1 tf.while_loop(tf_statement, body, variables)
```

It implement a function that if `tf_statement` is true, then run `body`, until `tf_statement` is false. The variables position need a tuple or list including all the tensorflow variables needed in the `body` part and `tf_statement` part. The loop will return the variables after the final iteration.

Example

```
1 t1 = tf.constant(1)
2 t2 = tf.constant(5)
3 body = (tf.add(t1,1), t2)
4 def cond(t1, t2): return t1 < t2
5 def body(t1, t2):
6     t1 = tf.add(t1, 1)
7     return (t1, t2)
8 res = tf.while_loop(cond, body, (t1, t2))
9 with tf.Session() as sess:
10     print(sess.run(res))
```

Framework

With all the knowledge we have learned, we now turn to something practical. Generally, we'd like the model training and model predicting to be simple, we prefer a calling style likes the following:

```
1 model = model_creation_function()
```

Firstly, we use a certain `model_creation_function` to implement the initialization of variables and parameters we need, actually, it create a *class* data in Python. Then:

```
1 model.fit(x=..., y=...)
```

we hope this class type model hold a function called "fit", and the input should be x and y (in most cases of supervised learning).

Finally,

```
1 predict_value = model.predict(x=...)
```

we also hope this class type model hold functions for prediction, evaluation and etc.

Linear Regression Exercise

Given $X \in \mathbb{R}^{n \times 2}$ and $Y \in \mathbb{R}^n$, find a $\beta \in \mathbb{R}^2$ which is the OLS estimator for model

$$Y = X\beta + \varepsilon, \varepsilon \sim N(0, \sigma^2).$$

Key idea:

Implement the (loss) objective function:

$$L(\beta) = \|Y - X\beta\|^2.$$

Give any initial value $\hat{\beta}^{(0)} = (0, 0)^T$ and use the Gradient Descent method. The solution of this exercise is attached in the code directory.

Sport data analysis

In this section, our goal is to make prediction on the result of football matches based on the twitter sentiment of 200 games from the English Premier League. Data is from the [Betsentiment.com](https://betsentiment.com) with structure (key columns):

- home: The home team
- away: The away team
- sentiment_score_st_a: The sentiment score of home team at the starting time of bet
- sentiment_score_st_b: The sentiment score of away team at the starting time of bet
- sentiment_score_lt_a: The sentiment score of home team at the last time of bet
- sentiment_score_lt_b: The sentiment score of away team at the last time of bet

Sport data analysis

- `wgted_players_sentiment_a`: The weighted players sentiment score of home team
- `wgted_players_sentiment_b`: The weighted players sentiment score of away team
- `wgted_players_sentiment_st_a`: The weighted players sentiment score of home team at the starting time of bet
- `wgted_players_sentiment_st_b`: The weighted players sentiment score of away team at the starting time of bet
- `final_result`: The final result (home win: 1; home loss: 2, tie: 3)

We run several neural network models with different structures, whose code is attached in the code directory. Thank you for listening. 🍀

Useful info



The material of this session is in [Session Material](#).

Our department website is [Department of Statistics](#).

Our website for statistical data science lab at Uconn is [Data Science Lab](#).