# Unifying Reifying and Symbolic-PyMC Continued

Joseph Willard

2019-24-06

## 1 Introduction

In the last blog post I focused on looking through TensorFlow objects and what could be used to recreate the graph of operations. Considering the analogy given in the first blog I should have enough information now to recreate the `str_optimize` function for TensorFlow. To do this I need the following,

1. A function that takes the Tensorflow operation and creates the graph

1. Functions that search the graph and look for replacements

Below is our original problem from the first blog,

```
""" Seeing if tensorflow has the same issue
"""
import numpy as np
import tensorflow as tf
from tensorflow.python.framework.ops import disable_eager_execution


tf.compat.v1.InteractiveSession()
disable_eager_execution()

X = np.random.normal(0, 1, (10, 10))

S = tf.matmul(X, X, transpose_a=True)

d, U, V = tf.linalg.svd(S)

```

```
D = tf.matmul(U, tf.matmul(tf.linalg.diag(d), V, adjoint_b=True))
ans = S - D
```

<tensorflow.python.client.session.InteractiveSession object at 0x7f9b556ce6a0>

# 2 Graph Reconstruction through TensorFlow part 2

Last blog I only described some of the objects of interest. Now, using `symbolic-pymc` in particular the `tf_dprint` function we can inspect the graph.

```
from symbolic_pymc.tensorflow.printing import tf_dprint

_ = tf_dprint(ans)
```

```
Tensor(Sub):0,shape=[10, 10] "sub:0"
|  Op(Sub) "sub"
|  |  Tensor(MatMul):0,shape=[10, 10] "MatMul:0"
|  |  |  Op(MatMul) "MatMul"
|  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/a:0"|
|  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/b:0"
|  |  Tensor(MatMul):0,shape=[10, 10] "MatMul_2:0"
|  |  |  Op(MatMul) "MatMul_2"
|  |  |  |  Tensor(Svd):1,shape=[10, 10] "Svd:1"
|  |  |  |  |  Op(Svd) "Svd"
|  |  |  |  |  |  Tensor(MatMul):0,shape=[10, 10] "MatMul:0"
|  |  |  |  |  |  |  Op(MatMul) "MatMul"
|  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/a:0"
|  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/b:0"
|  |  |  |  Tensor(MatMul):0,shape=[10, 10] "MatMul_1:0"
|  |  |  |  |  Op(MatMul) "MatMul_1"
|  |  |  |  |  |  Tensor(MatrixDiag):0,shape=[10, 10] "MatrixDiag:0"
|  |  |  |  |  |  |  Op(MatrixDiag) "MatrixDiag"
|  |  |  |  |  |  |  |  Tensor(Svd):0,shape=[10] "Svd:0"
|  |  |  |  |  |  |  |  |  Op(Svd) "Svd"
|  |  |  |  |  |  |  |  |  |  Tensor(MatMul):0,shape=[10, 10] "MatMul:0"
|  |  |  |  |  |  |  |  |  |  |  Op(MatMul) "MatMul"
|  |  |  |  |  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10]
"MatMul/a:0"
```

```
|  |  |  |  |  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10]
"MatMul/b:0"
|  |  |  |  |  |  Tensor(Svd):2,shape=[10, 10] "Svd:2"
|  |  |  |  |  |  |  Op(Svd) "Svd"
|  |  |  |  |  |  |  |  Tensor(MatMul):0,shape=[10, 10] "MatMul:0"
|  |  |  |  |  |  |  |  |  Op(MatMul) "MatMul"
|  |  |  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/a:0"
|  |  |  |  |  |  |  |  |  |  Tensor(Const):0,shape=[10, 10] "MatMul/b:0"
```

Inspecting the output the top layer (furthest left) represents the subtraction that took place. Each subsequent step right moves effectively one step down in the list of operations until the original inputs are reached.

From this point the next step is to write a function that can replace the the below portion, Tensor(MatMul):0, shape=[10, 10] "MatMul_2:0"

Op(MatMul) "MatMul_2"

Tensor(Svd):1, shape=[10, 10] "Svd:1"

Op(Svd) "Svd"

Tensor(MatMul):0, shape=[10, 10] "MatMul:0"

Op(MatMul) "MatMul"

Tensor(Const):0, shape=[10, 10] "MatMul/a:0"

Tensor(Const):0, shape=[10, 10] "MatMul/b:0"

Tensor(MatMul):0, shape=[10, 10] "MatMul_1:0"

Op(MatMul) "MatMul_1"

Tensor(MatrixDiag):0, shape=[10, 10] "MatrixDiag:0"

Op(MatrixDiag) "MatrixDiag"

Tensor(Svd):0, shape=[10] "Svd:0"



Op(Svd) "Svd"

Tensor(MatMul):0, shape=[10, 10] "MatMul:0"

With the following,

Tensor(MatMul):0, shape=[10, 10] "MatMul:0"

Op(MatMul) "MatMul"

Tensor(Const):0, shape=[10, 10] "MatMul/a:0"

Tensor(Const):0, shape=[10, 10] "MatMul/b:0"

## 3 Creating an optimizing function