Module 2

TensorFlow Primitives

Required

- *Duan, H., Liu, Y., Wang, D., He, L., & Xiao, X. (2019). Prediction of a multi-mode coupling model based on traffic flow tensor data (Links to an external site.). Journal of Intelligent & Fuzzy Systems, 36(2), 1691–1703. https://doi.org/10.3233/JIFS-18804*
- Ramsundar, B., & Zadeh, R. B. (2018). *TensorFlow for deep learning: From linear regression to reinforcement learning*. (pp. Chapter 2). O'Reilly Media, Inc.

Recommended

- Chapters 5-8 in *Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow*
- Li, Z., Sergin, N. D., Yan, H., Zhang, C., & Tsung, F. (2019). *Tensor completion for weakly-dependent data on graph for metro passenger flow prediction (Links to an external site.).*
- Li, Y., Wu, F.-X., & Ngom, A. (2018). A review on machine learning principles for multi-view biological data integration (Links to an external site.). *Briefings in Bioinformatics, 19*(2), 325-340.

▼ What is TensorFlow?

- a programming library available to Python as well as to other programming languages and is used to implement the computational realization of machine-learning algorithms

- it is language built on mathematical structures found in vector calculus and statistics

▼ Is TF declarative?

Yes, like Lisp, it is a declarative programming language in which you specify what needs to be done rather than specifying how things should be done

▼ How is TF declarative?

- TensorFlow creates a description of an operation to a “computation graph.”
- The code only describes a computation and leaves the actual details to the TensorFlow program.

▼ What is a tensor?

A tensor is a generalization of vectors and matrices and is easily understood as a **multidimensional array**.

▼ What is a vector?

A vector is a one-dimensional or first order tensor. **think a list**

▼ What is a matrix?

A matrix is a two-dimensional or second-order tensor.

▼ What is a Rank-0 tensor?

- a scalar
- a scalar contains a single value, and no "axes"

▼ What is a Rank-1 tensor?

- A rank-1 tensor is a vector, a list of real numbers
- Vectors are written as either column vectors or as row vectors

column vectors $\begin{pmatrix} a \\ b \end{pmatrix}$ or as row vectors $\begin{pmatrix} a & b \end{pmatrix}$

- Example:
  - The collection of all column vectors of length two is denoted $R^{2\times1}$ while the set of all row vectors of length two is denoted $R^{1\times2}$
  - Both types of vectors come from the set R^2

▼ What is a Rank-2 tensor?

- A rank-two tensor is a matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- It is referred to as $R^{2\times2}$
- The act of rotating a vector in a plane by angle α is represented by a rank-two tensor
- There are a number of standard math operations on matrices:
  - The matrix **transpose** is a convenient operation that flips a matrix around its diagonal.
  - If two matrices have the same number of rows and the same number of columns, respectively, then matrix addition is defined elementwise.
  - A matrix can be multiplied by a scalar that involves multiplying each element of the matrix by the scalar.
  - Two matrices, A and B, may be multiplied if the number of <u>rows</u> of A equals the number of <u>columns</u> of B.
    - In general, for matrix multiplication, $AB \neq BA$.
    - If A has shape (m, n) and B has shape (n, k) then C = AB has shape (m,k).

▼ What is a Rank-3 tensor?

- A rank-3 tensor is essentially a rectangular prism of numbers, e.g., $R^{3\times3\times3}$

- The shape of a Rank-3 tensor is (3,3,3). An arbitrary element of that tensor is specified using indices (i, j, k).
- **Black and white image**: Consider an image consisting of a grid of 224 x 224 pixels. Each pixel (i, j) is a 1 or a 0 to encode either black or white for each pixel. **The image is a matrix of (224, 224) with each value either a 1 or a 0**.
- **Color image**: If we think about a color image of the same size, each pixel has a color represented by red, green, and blue channels (RGB). As such, every pixel (i, j) in this image is a tuple of numbers (r, g, b) that represent the amount of red, green, and blue in the pixel. The three colors are usually integers ranging from 0 to 255, and each pixel needs all three values. **The color image can be represented as a rank-3 tensor of shape (224, 244, 3)**

▼ What is a vector space?

a collection of vectors where R represents real numbers:

$$V_1 = \mathbb{R}^{d_i}$$

▼ What are some essential TF commands?

```
import tensorflow as tf
#
tf.InteractiveSession()
#TensorFlow provides a number of functions that
#instantiate basic tensors in memory. The simplest
#of these are tf.zeros() and tf.ones(). tf.zeros()
#take a tensor shape (represented as a Python tuple)
#and returns a tensor of that shape filled with zeros.

tf.zeros(2)

#TensorFlow returns a reference to the desired tensor
#rather than the value of the tensor itself. To force
#the value of the tensor to be returned, use the method
#tf.Tensor.eval() of tensor objects:

a = tf.zeros(2)
a.eval()

a = tf.zeros((2, 3))
a.eval()
```

```
#array([[ 0., 0., 0.],
#   [ 0., 0., 0.]], dtype=float32)
b = tf.ones((2,2,2))
b.eval()
#array([[[ 1., 1.],
#[ 1., 1.]],
#[[ 1., 1.],
#[ 1., 1.]]], dtype=float32)

#A tensor can be filled with values you
#desire using the tensor fill command:

b = tf.fill((2, 2), value=5.)
b.eval()
#array([[ 5., 5.], [ 5., 5.]], dtype=float32)
#“tf.constant” is another function, similar to
#tf.fill, which allows for construction of tensors
#that shouldn’t change during the program execution:

a = tf.constant(3)
a.eval()
#3

#In machine learning, neural network weights
#(the nodes) are initialized with random values
#from tensors rather than constants. One way to
#do this would be to select a random value from
#a normal distribution with a given mean and standard
#deviation.

a = tf.random_normal((2, 2), mean=0, stddev=1)
a.eval()
#array([[-0.73437649, -0.77678096],[ 0.51697761, 1.15063596]], dtype=float32)
#In more complex machine learning programs using normal
#distributions for tensors, numerical instability can arise
#if a random value is far from the mean. To compensate for
#this, in order to get to convergence, use tf.truncated_normal().
#The function will resample values from more than two standard
#deviations from the mean.

#tf.random_uniform() behaves like tf.random_normal() except for
#the fact that random values are sampled from the Uniform distribution
#over a specified range.

a = tf.random_uniform((2, 2), minval=-2, maxval=2)
a.eval()
#array([[-1.90391684, 1.4179163 ],[ 0.67762709, 1.07282352]], dtype=float32)

#Addition and scalar multiplication apply to tensors:
c = tf.ones((2, 2))
d = tf.ones((2, 2))
e = c + d
e.eval()
array([[ 2., 2.],[ 2., 2.]], dtype=float32)
```

```
f = 2 * e
f.eval()
#array([[ 4., 4.],[ 4., 4.]], dtype=float32)

#Addition and scalar multiplication apply to tensors:
c = tf.ones((2, 2))
d = tf.ones((2, 2))
e = c + d
e.eval()
#array([[ 2., 2.],[ 2., 2.]], dtype=float32)
f = 2 * e
f.eval()
#array([[ 4., 4.],[ 4., 4.]], dtype=float32)

#Tensor multiplication occurs elementwise and not matrix multiplication:

c = tf.fill((2,2), 2.)
d = tf.fill((2,2), 7.)
e = c * d
e.eval()
#array([[ 14., 14.],[ 14., 14.]], dtype=float32)

#Identity matrices are square matrices that are 0
#everywhere except on the diagonal where they are 1.
#tf.eye() allows for fast construction of identity
#matrices of desired size.

a = tf.eye(4)
a.eval()
#array([[ 1., 0., 0., 0.],[ 0., 1., 0., 0.],
#[ 0., 0., 1., 0.],[ 0., 0., 0., 1.]], dtype=float32)

#Diagonal matrices are another common type of matrix.
#Like identity matrices, diagonal matrices are only nonzero
#along the diagonal. Unlike identity matrices, they may take
#arbitrary values along the diagonal. The easiest way for doing
#this is to invoke tf.range(start, limit, delta). Note that
#limit is excluded from the range and delta is the step size
#for the traversal. The resulting vector can then be fed to
#tf.diag(diagonal), which will construct a matrix with the
#specified diagonal.

r = tf.range(1, 5, 1)
r.eval()
#array([1, 2, 3, 4], dtype=int32)
d = tf.diag(r)
d.eval()
#array([[1, 0, 0, 0],
#   [0, 2, 0, 0],
#   [0, 0, 3, 0],
#   [0, 0, 0, 4]], dtype=int32)

#Use tf.matrix_transpose() to take the transpose of a matrix:
```

```
a = tf.ones((2, 3))
a.eval()
#array([[ 1., 1., 1.],
#[ 1., 1., 1.]], dtype=float32)
at = tf.matrix_transpose(a)
at.eval()
#array([[ 1., 1.],
#   [ 1., 1.],
#   [ 1., 1.]], dtype=float32)

#To multiply two matrices, use tf.matmul.

a = tf.ones((2, 3))
a.eval()
#array([[ 1., 1., 1.],
#[ 1., 1., 1.]], dtype=float32)
b = tf.ones((3, 4))
b.eval()
#array([[ 1., 1., 1., 1.],
#[ 1., 1., 1., 1.],
#[ 1., 1., 1., 1.]], dtype=float32)
c = tf.matmul(a, b)
c.eval()
#array([[ 3., 3., 3., 3.],
#[ 3., 3., 3., 3.]], dtype=float32)

#Tensors in TensorFlow come in a variety of
#types such as tf.float32, tf.float64, tf.int32,
#and tf.int64. It’s possible to create tensors
#of specified types by setting dtype in tensor
#construction functions. Furthermore, given a tensor,
#it’s possible to change its type using casting functions
#such as tf.to_double(), tf.to_float(), tf.to_int32(),
#tf.to_int64(), and others.

a = tf.ones((2,2), dtype=tf.int32)
a.eval()
#array([[0, 0],
#[0, 0]], dtype=int32)
b = tf.to_float(a)
b.eval()
#array([[ 0., 0.],
#[ 0., 0.]], dtype=float32)

#tf.reshape() allows tensors to be converted into tensors with different shapes:

a = tf.ones(8)
a.eval()
#array([ 1., 1., 1., 1., 1., 1., 1., 1.], dtype=float32)
b = tf.reshape(a, (4, 2))
b.eval()
#array([[ 1., 1.], [ 1., 1.], [ 1., 1.], [ 1., 1.]], dtype=float32)
c = tf.reshape(a, (2, 2, 2))
c.eval()
```

```
#array([[[ 1., 1.],
#[ 1., 1.]],
#[[ 1., 1.],
#[ 1., 1.]]], dtype=float32)

#One way to get the shape of a tensor is to use the tf.Tensor.get_shape() method:

a = tf.ones(2)
a.get_shape()
#TensorShape([Dimension(2)])
a.eval()
#array([ 1., 1.], dtype=float32)
b = tf.expand_dims(a, 0)
b.get_shape()
#TensorShape([Dimension(1), Dimension(2)])
b.eval()array([[ 1., 1.]], dtype=float32)
c = tf.expand_dims(a, 1)
c.get_shape()
#TensorShape([Dimension(2), Dimension(1)])
c.eval()
#array([[ 1.],[ 1.]], dtype=float32)
d = tf.squeeze(b)
d.get_shape()
#TensorShape([Dimension(2)])
d.eval()
#array([ 1., 1.], dtype=float32)

#Broadcasting is a term (introduced by NumPy) for when a tensor
#system's matrices and vectors of different sizes can be added together.
#These rules allow for conveniences like adding a vector to every row of
#a matrix.

a = tf.ones((2, 2))
a.eval()
#array([[ 1., 1.],
#[ 1., 1.]], dtype=float32)
b = tf.range(0, 2, 1, dtype=tf.float32)
b.eval()
#array([ 0., 1.], dtype=float32)
c = a + b
c.eval()
#array([[ 1., 2.],
#[ 1., 2.]], dtype=float32)

#In this case vector b is added to every row of matrix a.
#Be sure to set the dtype when doing this, because, otherwise,
#Python will report an error.
```

▼ What is a tf.Graph object?

- A TensorFlow computation is a representation of an instance of the tf.Graph object.
- These graphs are a set of instances of tf.Tensor and tf.Operation objects.
- If we do not specify a tf.Graph to use for operation, the program creates a hidden tf.Graph instance, which can be called using the tf.get_default_graph().

▼ What is a tf.Operation instance?

Calls to operations create a tf.Operation instance that tells the program to execute a matrix multiplication operation during the program run time.

▼ What is a tf.Session() object?

a tf.Session() object stores the context under which a computation is performed

```
sess = tf.Session()
a = tf.ones((2, 2))
b = tf.matmul(a, a)
b.eval(session=sess)
array([[ 2., 2.],
   [ 2., 2.]], dtype=float32)
#alternatively
sess.run(b)
array([[ 2., 2.],
   [ 2., 2.]], dtype=float32)
```

▼ What is the tf.Variable() class?

- TensorFlow provides the tf.Variable() class as a wrapper for the tensors, which allows for stateful computations. These variable objects hold the stored tensors and allow the state of the machine to be recorded and changed.
- Variables within TensorFlow have to be explicitly initialized, and TensorFlow provides an easy way to initialize all variables using the tf.global_variables_initializer
- After initialization, the variable holds a plain tensor, allowing you to use it with the various tensor commands. You can update the value of a variable using the tf.assign() command

- You cannot change the shape (matrix) of a variable after you initialize it. You may change the values but not the dimensions of the matrix.

```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

a = tf.Variable(tf.ones((2, 2)))
a
# tf.Variable 'Variable:0' shape=(2, 2) dtype=float32_ref

a.eval(session=sess)
# array([[ 1., 1.],
#   [ 1., 1.]], dtype=float32)

sess.run(a.assign(tf.zeros((2,2))))
# array([[ 0., 0.],
#   [ 0., 0.]], dtype=float32)
```