TensorFlow
- tensor and others -

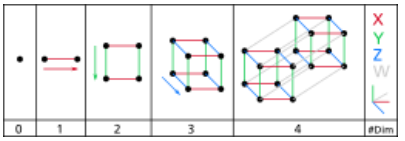
Aug. 2019

Ando Ki, Ph.D.
adki@future-ds.com

Table of contents

Tensor Ranks and Shapes

- Tensor: n-dimensional array or list
- Tensor: static type and dynamic dimensions
  - ▶ 데이터 타입은 불변, 모양은 필요에 따라 변경
- **Rank**: number of dimensions of the tensor
- **Shape**: size of each dimension



Rank 0: zero-dimensional array ➔ a point; a scalar
Eg: s=48.3
Shape []

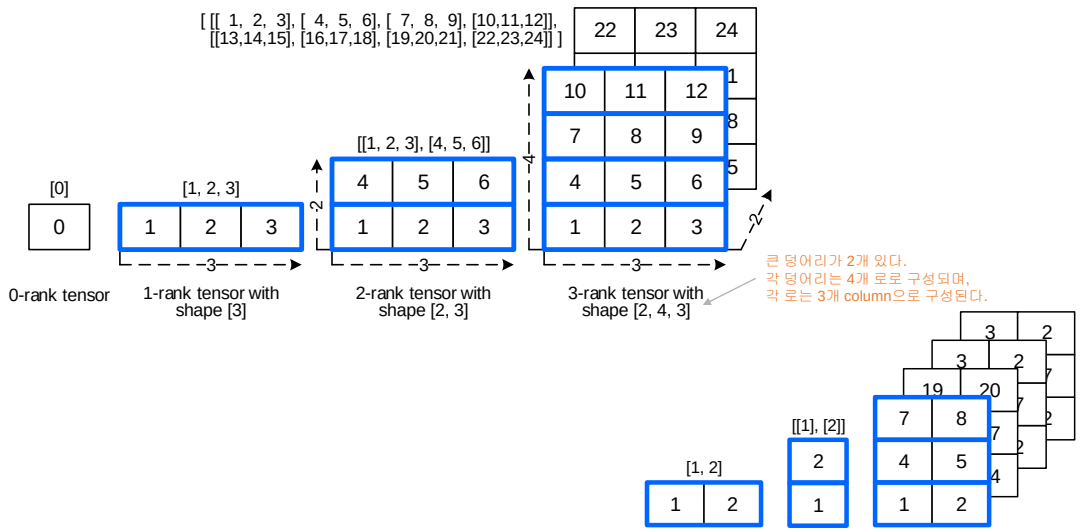
Rank 1: one-dimensional array ➔ points on a line; a vector
Eg: v=[1, 2, 3, 4, 5]
Shape [5]

Rank 2: two-dimensional array ➔ lines on a surface; a matrix with two coordinates and pxq entries.
Eg: m = [[1, 2], [3, 4], [5, 6]] ➔ row-major
Shape [3, 2]

Rank 3: three-dimensional array ➔ surface of a cube; pxqxr entries
Eg: t = [[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[ ], [ ], [ ]], [[ ], [ ], [ ]]] # details are not shown
Shape [3, 3, 3]

3

Tensor Ranks and Shapes



4

Tensor rank and shape and type

- 'tensor.py'
  - how to define tensors ➔ three different ways
- 'matrix.py'
  - figure out rank and shape

See: ~/tensorflow-projects/tensor

5

Ways to define tensor: tensor.py

See: ~/tensorflow-projects/tensor

```
import tensorflow as tf
import numpy as np

# Define a 2x2 matrix in 3 different ways
m1 = [[1.0, 2.0], [3.0, 4.0]]
m2 = np.array([[1.0, 2.0], [3.0, 4.0]], dtype=np.float32)
m3 = tf.constant([[1.0, 2.0], [3.0, 4.0]])
# Print the type for each matrix
print(type(m1)) # <type 'list'>
print(type(m2)) # <type 'numpy.ndarray'>
print(type(m3)) # <class 'tensorflow.python.framework.ops.Tensor'>
# Create tensor objects out of the different types
t1 = tf.convert_to_tensor(m1, dtype=tf.float32)
t2 = tf.convert_to_tensor(m2, dtype=tf.float32)
t3 = tf.convert_to_tensor(m3, dtype=tf.float32)
# Notice that the types will be the same now
print(type(t1)) # <class 'tensorflow.python.framework.ops.Tensor'>
print(type(t2)) # <class 'tensorflow.python.framework.ops.Tensor'>
print(type(t3))
```

Three different ways of defining matrix

Create tensor from different types

6

Ways to define tensor: tensor.py

- This example shows how define arrays
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/tensor
  - Step 2: see the codes
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host] python tensor.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/tensor
[user@host] python tensor.py
```


Ways to define tensor: matrix.py

See: ~/tensorflow-projects/tensor

```
import tensorflow as tf

# Define a 2x1 matrix
matrix1 = tf.constant([[1., 2.]])
# Define a 1x2 matrix
matrix2 = tf.constant([[1], [2]])
# Define a rank 3 tensor
myTensor = tf.constant([ [[1,2],[3,4],[5,6]],[[7,8],[9,10],[11,12]] ])
# Try printing the tensors
print(matrix1)
print(matrix2)
print(myTensor)
sess = tf.Session()
result = sess.run(matrix1)
print result
result = sess.run(matrix2)
print result
result = sess.run(myTensor)
print result
```

Defining matrix

Print information, not value

Need to call run() to evaluate values of tensors

Ways to define tensor: matrix.py

- This example shows how define arrays
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/tensor
  - Step 2: see the codes
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host] python matrix.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/tensor
[user@host] python matrix.py
```

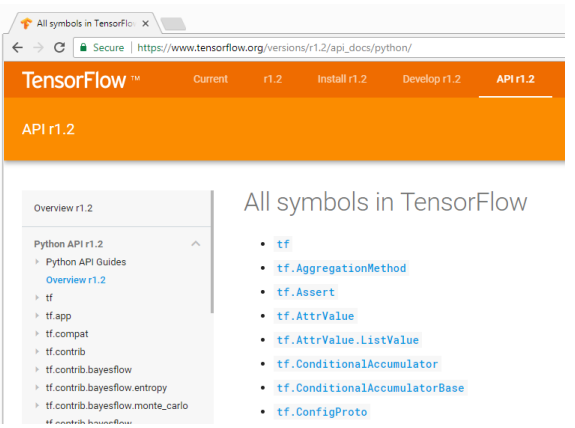

Tensor data types

- Data type specifies the kind of values the tensor contains.
  - C = tf.constant(3, dtype=tf.int32)
  - P = tf.placeholder(dtype=tf.bool)
  - M = tf.Variable([,], **dtype=tf.float32**)

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_UINT16	tf.uint16	16 bits unsigned integer.
DT_STRING	tf.string	Variable length byte arrays. Each element of a Tensor is a byte array.
DT_BOOL	tf.bool	Boolean.
DT_COMPLEX64	tf.complex64	Complex number made of two 32 bits floating points: real and imaginary parts.
DT_COMPLEX128	tf.complex128	Complex number made of two 64 bits floating points: real and imaginary parts.
DT_QINT8	tf.qint8	8 bits signed integer used in quantized Ops.
DT_QINT32	tf.qint32	32 bits signed integer used in quantized Ops.
DT_QUINT8	tf.quint8	8 bits unsigned integer used in quantized Ops.

How to get man page for TensorFlow

- Visit following web-page for Python
  - https://www.tensorflow.org
    - versions → r1.2 → api_docs → python



TensorFlow modules

- app module: Generic entry point script.
- compat module: Functions for Python 2 vs. 3 compatibility.
- contrib module: contrib module containing volatile or experimental code.
- errors module: Exception types for TensorFlow errors.
- estimator module: Estimator: High level tools for working with models.
- feature_column module: FeatureColumns: tools for ingesting and representing features.
- flags module: Implementation of the flags interface.
- gfile module: Import router for file_io.
- graph_util module: Helpers to manipulate a tensor graph in python.
- image module: Image processing and decoding ops.
- layers module: This library provides a set of high-level neural networks layers.
- logging module: Logging utilities.
- losses module: Loss operations for use in neural networks.
- metrics module: Evaluation-related metrics.
- nn module: Neural network support.
- python_io module: Python functions for directly manipulating TFRecord-formatted files.
- pywrap_tensorflow module: pywrap_tensorflow wrapper that exports all symbols with RTLD_GLOBAL.
- resource_loader module: Resource management library.
- saved_model module: Convenience functions to save a model.
- sets module: Tensorflow set operations.
- spectral module: Spectral operators (e.g. FFT, RFFT).
- summary module: Tensor summaries for exporting information about a model.
- sysconfig module: System configuration library.
- test module: Testing.
- tools module
- train module: Support for training models.
- user_ops module: All user ops.

tf.placeholder

- 'placeholder'
  - a variable that will be assigned data to at a later time
    - with 'feed_dict' argument of 'sess.run()'
  - it is a function (while 'Variable' is class)
  - Python style function prototype: tf.placeholder(dtype,shape,name)
  - args:
    - dtype: type of elements in the tensor to be fed
    - shape: optional shape of the tensor (any shape can be fed if it is not specified)
    - name: optional name of the tensor
  - returns:
    - a tensor with specified shape and type

`tf.placeholder`

```
placeholder(
    dtype,
    shape=None,
    name=None
)
```

```
x = tf.placeholder(tf.float32, shape=(1024, 1024))
y = tf.matmul(x, x)

with tf.Session() as sess:
  print(sess.run(y))  # ERROR: will fail because x was not fed.

  rand_array = np.random.rand(1024, 1024)
  print(sess.run(y, feed_dict={x: rand_array}))  # Will succeed.
```

refer to 'tf.placeholder_with_default()'

13

tf.Variable

- 'Variable'
  - in-memory buffers containing tensors
  - it is class (while 'placeholder' is a function) → note capital 'V'.
  - must be explicitly initialized by using
    - variable initializer: initializer()
    - variable assign: assign()
    - variable initializer: tf.variables_initializer()
    - global initializer: tf.global_variables_initializer()
  - Python style function prototype: tf.Variable(initial_value,dtype,name)
  - args:
    - initial_value: a tensor as initial value
    - dtype: type of elements in the tensor to be fed
    - name: optional name of the tensor
  - returns:
    - a tensor with specified shape and type

14

Cost functions

- Absolute error
  - Sum of absolute errors
    - sum(|t – y|)
  - Mean absolute errors (MAE)
    - 1/n*sum(|t-y|)
- Squared error loss
  - Sum of squared errors
    - sum((t - y)**2)
  - Mean squared errors (MSE)
    - 1/n*sum((t-y)**2)
  - Root mean square errors (RMSE)
    - (MSE)**1/2
- Cross-entropy loss
  - For classification after softmax
  - Sum of cross-entropy loss
    - -sum(t*log(y))
    - or - sum[t*log(y) + (1-t)*log(1-y)]

- tf.reduce_sum()
- tf.reduce_mean()
- tf.abs()
  - tf.reduced_mean(tf.abs())
- tf.square()
  - tf.reduced_mean(tf.squre())
- tf.nn.softmax_cross_entropy_with_logits()

Optimization algorithms

- class tf.train.GradientDescentOptimizer
- class tf.train.AdagradOptimizer
- class tf.train.MomentumOptimizer
- class tf.train.AdamOptimizer

㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems