

구글 TensorFlow 첫걸음

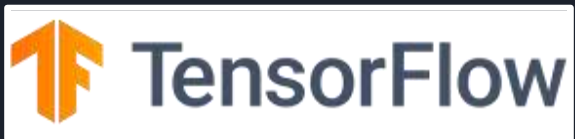
김환희

2019.03.23

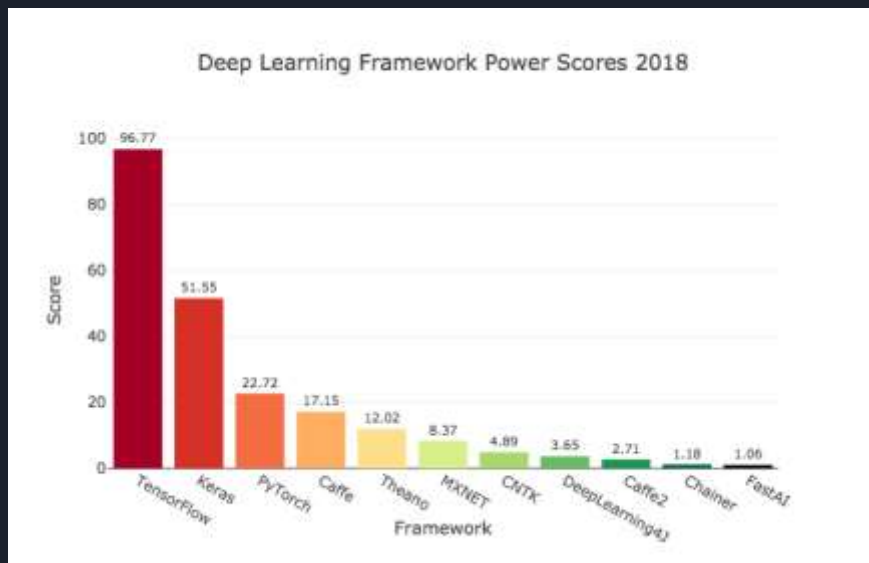
목차

- Tensorflow 소개
- Tensorflow Dev Summit 2019 요약 (Tensorflow 2.0 발표)
- Tensorflow 2.0 Sample code

Tensorflow 소개



- 2015년 오픈소스로 공개된 Google Brain 의 머신러닝 프로젝트
- 세계에서 가장 널리 쓰이는 Deep Learning Framework





- Tensorflow, CNTK, Theano 등의 저수준 API 를 좀 더 간편하게 사용하기 위한 고수준 API
- 2017년 중반부터 Tensorflow 에 정식으로 통합
- 2.0 버전부터 Tensorflow 의 기본 고수준 API 가 됨
 - 즉, 간결한 코드를 사용하는 고수준 API 작업은 `tf.keras` 를 기본으로 사용하도록 통일됨





- 코드 길이 비교
- XOR 문제

Tensorflow

```
import numpy as np
import tensorflow as tf

# training set
X_train = np.array([[0,0], [0,1], [1,0], [1,1]])
T_train = np.array([[0], [1], [1], [0]])

# placeholder
X = tf.placeholder(tf.float32, [None, 2])
T = tf.placeholder(tf.float32, [None, 1])

# variable
W1 = tf.Variable(tf.truncated_normal([2,4]))
b1 = tf.Variable(tf.zeros([4]))
W2 = tf.Variable(tf.truncated_normal([4,1]))
b2 = tf.Variable(tf.zeros([1]), dtype=tf.float32)

# model
A1 = tf.matmul(X, W1)+b1
Z1 = tf.sigmoid(A1)
A2 = tf.matmul(Z1, W2)+b2
Z2 = tf.sigmoid(A2)

learn_rate = 0.1
Cost = tf.reduce_mean(tf.reduce_sum(tf.square(Z2-T), 1))
train = tf.train.GradientDescentOptimizer(learn_rate).minimize(Cost)

predict = Z2

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

for i in range(5000):
    _train, _Cost = sess.run([train, Cost], feed_dict={X:X_train, T:T_train})

    _predict = sess.run([predict], feed_dict={X:X_train})
    print("predict=", _predict)
    print("result=", np.array(np.array(_predict)>0.5, np.int))
```

Keras

```
import numpy as np
import tensorflow as tf

X_train = np.array([[0,0], [0,1], [1,0], [1,1]])
T_train = np.array([[0], [1], [1], [0]])

model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(4, input_dim=2))
model.add(tf.keras.layers.Activation('sigmoid'))
model.add(tf.keras.layers.Dense(1))
model.add(tf.keras.layers.Activation('sigmoid'))

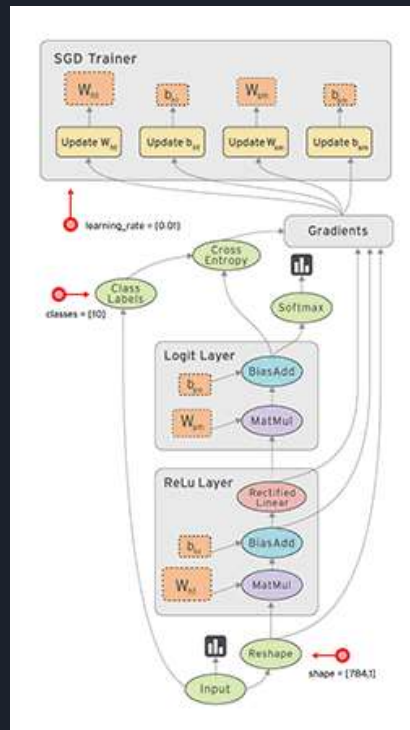
sgd = tf.keras.optimizers.SGD(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer=sgd)

model.fit(X_train, T_train, batch_size=1, epochs=5000)

_predict = model.predict_proba(X_train)
print("predict=", _predict)
print("result=", np.array(np.array(_predict)>0.5, np.int))
```

Tensorflow 의 특징


- 데이터 플로우 그래프
 - 수학 계산과 데이터의 흐름을 그래프로 표현
 - `tf.Graph` 로 빌드 후 `tf.Session` 으로 실행 필수
 - 1.x 버전 : `session.run()` 으로 실행해야만 각 노드에 있는 값을 확인할 수 있었음
→ 디버그가 어렵고 사용이 불편
 - 2.0 버전에서 개선됨



Tensorflow 의 특징

- Scalable
 - node 수가 많아져도 안정적으로 계산
 - 미국 에너지 연구 과학 컴퓨팅 센터(NERSC) 에서 Tensorflow 를 사용해서 슈퍼컴퓨터를 이용한 기후 예측
- CPU/GPU/TPU(2.0 버전부터) 지원
 - 같은 코드로 학습 가능
 - 분산 처리를 할 때도 최소한의 코드 추가로 가능
- 다양한 언어 지원
 - Python, C, C++, Javascript, Java, Swift, R, C#, Go, Rust, Julia, ...

Tensorflow 의 특징

- Google Colab 지원 
 - Jupyter Notebook 과 완전히 호환되는 코드 공유 플랫폼
 - GPU, TPU 를 사용한 가속 가능
 - 현재 1.13.1 버전 Tensorflow 는 import 후 바로 사용 가능하고, 간단한 bash 설치 명령으로 2.0 버전도 사용 가능

```
!pip install tf-nightly-gpu-2.0-preview  
  
import tensorflow as tf  
  
print(tf.__version__)  
  
>> 2.0.0-dev20190314
```

```
import tensorflow as tf  
  
print(tf.__version__)  
  
>> 1.13.1
```

Tensorflow Dev Summit 2019 요약 (Tensorflow 2.0 발표)

Tensorflow Dev Summit 2019

- 2019년 3월 6-7일, 미국 캘리포니아 Sunnyvale 에 있는 Google Event Center 에서 진행됨 (통산 세번째)
- 1,000 명의 참가자, 10,000 명의 livestream 시청자
- 35개의 youtube 영상 플레이리스트
- **Tensorflow 2.0** 을 비롯한 다양한 업데이트 내용과 넷이즈, 우버, 알리바바 등 Tensorflow 를 사용하고 있는 기업들, 연구진들의 사례 발표

Tensorflow 2.0

- 1.0 버전(2017년 1월) 이후 두 번째 메이저 릴리즈
- 편의성과 사용성에 중심을 두고, 내부 구조도 변화
- 일부는 1.x 버전부터 도입되었지만 2.0 에서 더 새롭게 변화하기도 함 (`tf.keras`, `tf.estimator` 등)

Tensorflow 2.0 주요 변화

- API Cleanup
- Eager execution (즉시 실행 모드)
- Functions, not sessions
- High-level API (`tf.keras`)

API Cleanup

- 1.x 버전에서는 어떤 일을 하기 위한 여러 가지 방법이 존재
- 2.0 에서는 하나만 남기고 모두 정리 (`tf.optimizer`, `tf.metrics`, `tf.losses`, `tf.layers...`)
- 1.x 버전 사용자의 혼란을 막기 위해 호환 코드를 제공 (`tf.compat.v1`). 하지만 `tf.contrib` 는 미포함



```
[4] !tf_upgrade_v2 --infile text_generation.py --outfile text_generation_upgraded.py

[+] INFO line 4:0: Renamed 'tf.enable_eager_execution' to 'tf.compat.v1.enable_eager_exe
INFO line 240:16: Renamed 'tf.train.AdamOptimizer' to 'tf.compat.v1.train.AdamOptimi
INFO line 332:21: Added keywords to args of function 'tf.multinomial'
INFO line 332:21: Renamed 'tf.multinomial' to 'tf.random.categorical'
INFO line 375:12: Renamed 'tf.train.AdamOptimizer' to 'tf.compat.v1.train.AdamOptimi
INFO line 392:21: tf.losses.sparse_softmax_cross_entropy requires manual check. tf.l
INFO line 392:21: Renamed 'tf.losses.sparse_softmax_cross_entropy' to 'tf.compat.v1.
TensorFlow 2.0 Upgrade Script
-----
Converted 1 files
Detected 0 issues that require attention
-----

Make sure to read the detailed log 'report.txt'
```

Eager execution (즉시 실행 모드)

- `tf.Session()` 이 `tf.compat.v1` 아래로 이동되고,
즉시 실행 모드가 기본으로 활성화 됨

```
import tensorflow as tf
print(tf.executing_eagerly())

>> True
```


Eager execution (즉시 실행 모드)

- 기존에는 `tf.Session()` 안에서만 수행 가능했던 연산을 Session 없이 자유롭게 쓸 수 있음

```
import tensorflow as tf

with tf.Session():
    result = tf.add(2,3).eval()
    print(result)

>> 5
```

TF 1.x

```
import tensorflow as tf

result = tf.add(2,3).numpy()
print(result)

>> 5
```

TF 2.0

Functions, not sessions

- `tf.Session()` 대신 `@tf.function` 을 Python 함수 앞에 decorator 로 사용할 수 있음
- 이전과 동일하게 사용가능 + 그래프로 컴파일 됨
 - 빠른 실행, GPU & TPU 지원, SavedModel export 가능

```
def simple_nn_layer(x, y):  
    return tf.nn.relu(tf.matmul(x, y))  
  
x = tf.random.uniform((3, 3))  
y = tf.random.uniform((3, 3))  
  
simple_nn_layer(x, y)
```

```
@tf.function  
def simple_nn_layer(x, y):  
    return tf.nn.relu(tf.matmul(x, y))  
  
x = tf.random.uniform((3, 3))  
y = tf.random.uniform((3, 3))  
  
simple_nn_layer(x, y)
```

Functions, not sessions

- `@tf.function` 을 사용한 함수에서 호출되는 다른 함수도 그래프에 자동으로 포함됨(AutoGraph)

```
def linear_layer(x):  
    return 2 * x + 1  
  
@tf.function  
def deep_net(x):  
    return tf.nn.relu(linear_layer(x))  
  
deep_net(tf.constant([1, 2, 3]))  
  
>> <tf.Tensor: id=57, shape=(3,), dtype=int32, numpy=array([3, 5, 7], dtype=int32)>
```

Functions, not sessions

- `@tf.function` 을 사용하면 `tf.cond`, `tf.case`, `tf.while_loop` 같은 control flow 를 `tf.Session()` 을 통해 실행할 필요가 없음

```
import tensorflow as tf

x = tf.constant(1.)
bool = tf.constant(True)
res = tf.cond(bool, lambda: tf.add(x, 1.), lambda: tf.add(x, 10.))

with tf.Session():
    result = res.eval()
    print(result)

>> 2.0
```

TF 1.x

```
@tf.function
def add_1_or_10(x, b):
    if b:
        x += 1
    else:
        x += 10
    return x

result = add_1_or_10(tf.constant(1), True).numpy()
print(result)

>> 2
```

TF 2.0

High-level API (tf.keras)

- `tf.keras` 는 편리한 문법으로 작은 프로젝트에 사용
- `tf.estimator` 는 깊고 넓은 구조의 큰 프로젝트에 사용
- 2.0 에서는 두 개를 합쳐서 하나의 `tf.keras` 로 제공
 - 즉, 간결한 코드를 사용하는 고수준 API 작업은 `tf.keras` 를 기본으로 사용하도록 통일됨

High-level API (tf.keras)

- 두 코드의 차이
 - 겉으로는 동일하지만, 내부에서는 각각 Session 과 Eager mode 를 사용해서 동작

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax'),
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

TF 1.x

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax'),
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

TF 2.0

High-level API (tf.keras)

- RNN Layer 는 사용자가 어떤 device 를 사용하고 있느냐에 따라 최적의 퍼포먼스를 낼 수 있는 Layer 들이 달라졌기 때문에 코드가 복잡해지는 원인이 되었음

```
# TF1.x: RNN layers
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Embedding(1000, 64, input_length=10))

if tf.test_is_gpu_available():
    model.add(tf.keras.layers.CudnnLSTM(32))
else:
    model.add(tf.keras.layer.LSTM(32))
```

High-level API (tf.keras)

- TF 2.0 에서는 하나의 호출만 남기고, 런타임에 최적의 Layer 를 선택

```
# TF2.0: RNN layers
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(1000, 64, input_length=10))

# This will use a Cudnn kernel when a GPU is available, otherwise calls basic LSTM()
model.add(tf.keras.layers.LSTM(32))
```


Tensorflow 2.0 Sample code

Simple CNN with flower_photos Dataset

- 텐서플로우 공식 홈페이지의 ["Build an Image input pipeline"](#) 을 TF 2.0 에 맞게 수정했습니다.
- TF 2.0 설치, Dataset, tf.keras, 텐서보드 등의 내용을 포함합니다.
- [Google Colab 코드 링크](#)

Reference

- [텐서플로우 시작하기 \(github\)](#)
- [TensorFlow Dev Summit 2019 \(Youtube playlist\)](#)
- [2019 Tensorflow Dev Summit 요약 \(blog\)](#)
- [François Chollet \(케라스 founder\) 의 Twitter & colab 게시물](#)
- [Tensorflow free course \(Udacity\)](#)
- [Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning \(Coursera\)](#)
- [Tensorflow 공식 사이트](#)

Reference

- 2019 TF dev summit(2) - Introducing Tensorflow 2.0 and its high-level APIs
- 텐서플로우 2.0에서 달라지는 점
- Effective Tensorflow 2.0

Thank you!