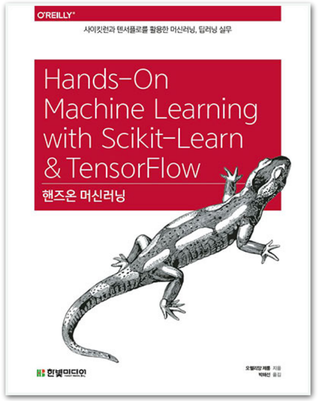


핸즈온 머신러닝

9장. 텐서플로 시작하기

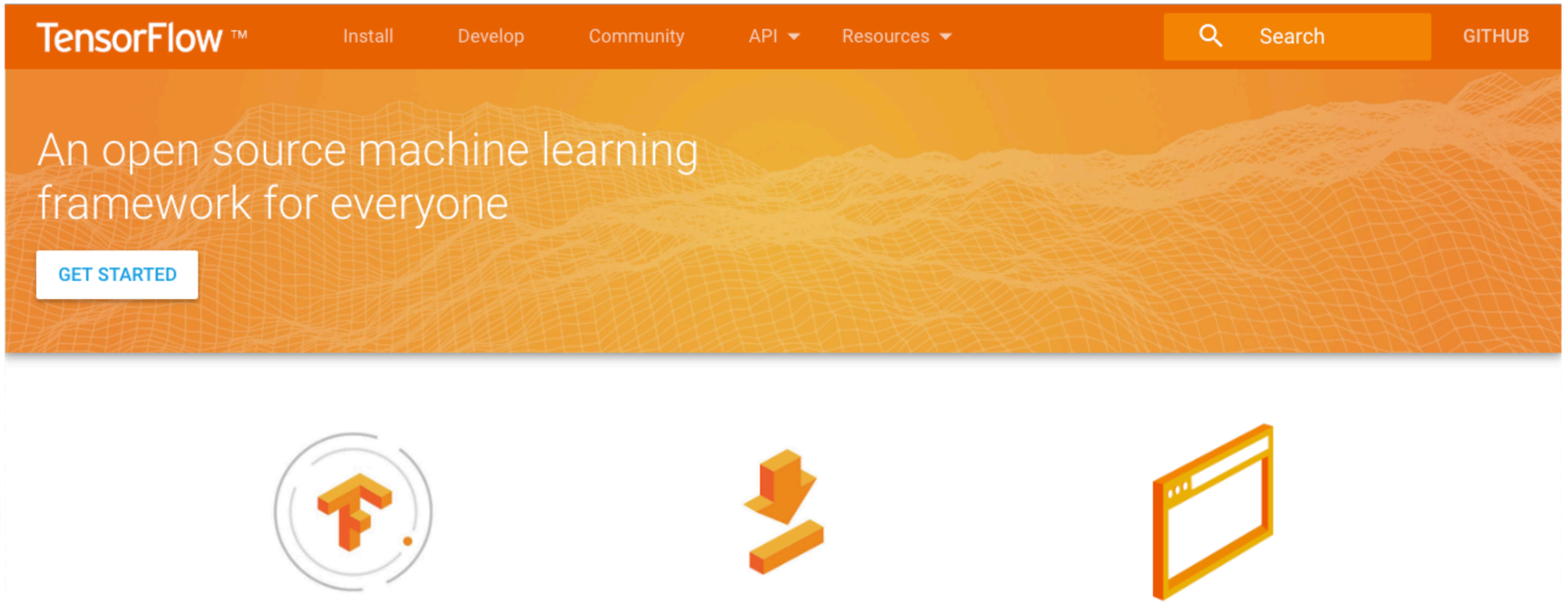
박해선(옮긴이)
haesun.park@tensorflow.blog
https://tensorflow.blog


케라스 창시자에게 배우는 딥러닝
창시자의 철학까지 담았다!
DEEP LEARNING WITH PYTHON
프랑소와 숄레 지음
박해선 옮김
딥러닝, 신경망, 머신 러닝 기초부터
컴퓨터 비전과 텍스트, 시퀀스, 생성 모델을 위한 딥러닝까지
입문하면서 알아야 할 모든 것을 설명한다!
길벗

tensorflow.org

2015.11: 0.5.0 공개

2018.11: 1.12.0 공개



≠ **tensorflow.blog**

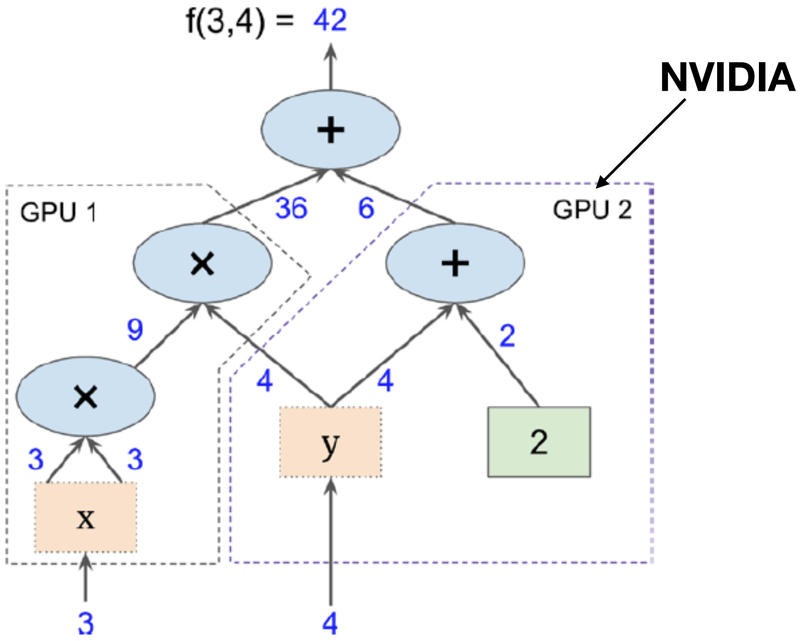
설치 방법

- 리눅스 & 맥
  - CPU : conda install tensorflow
  - GPU : pip install tensorflow-gpu, 그리고 CUDA, cuDNN
- 윈도우즈: conda (https://tensorflow.blog/윈도우즈에-아나콘다-텐서플로우-설치하기/)
  - 64비트 CPU만 지원
  - Python 3.6 버전용 아나콘다 설치(https://repo.anaconda.com/archive/Anaconda3-5.2.0-Windows-x86_64.exe)
  - AVX를 지원하지 않는 CPU: tensorflow 1.5.0 (인텔 프로세스 유틸리티로 확인)

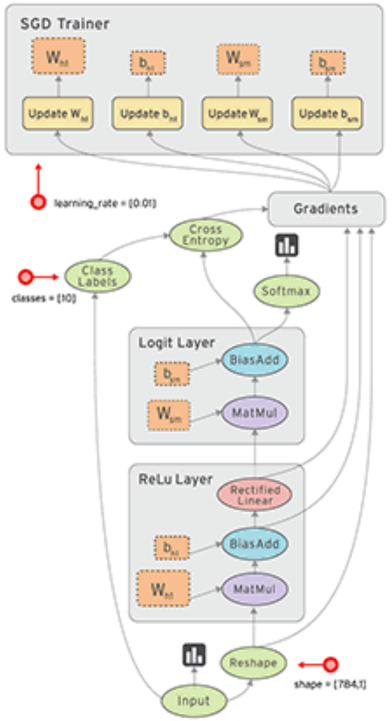
텐서플로 특징

- 윈도, 리눅스, macOS, 모바일(TensorFlow Lite), 웹(TensorFlow JS)
- TF.Learn, TF-slim, prettytensor —> Keras
- cuDNN > C++ API > Python API > Keras
- AutoDiff, TensorBoard, Eager Execution and more…

계산 그래프



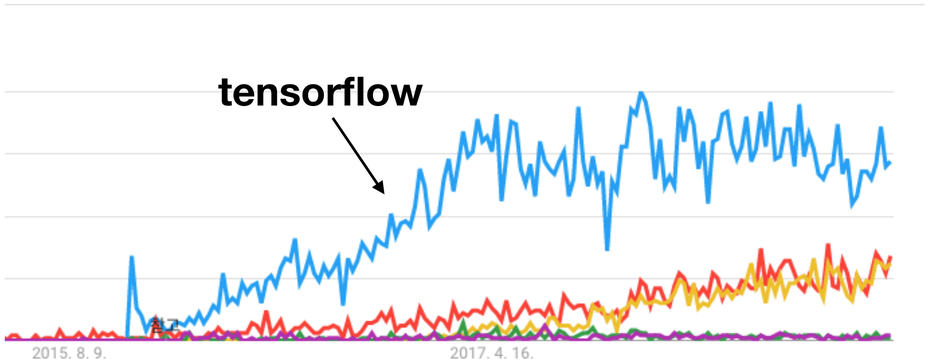
NVIDIA



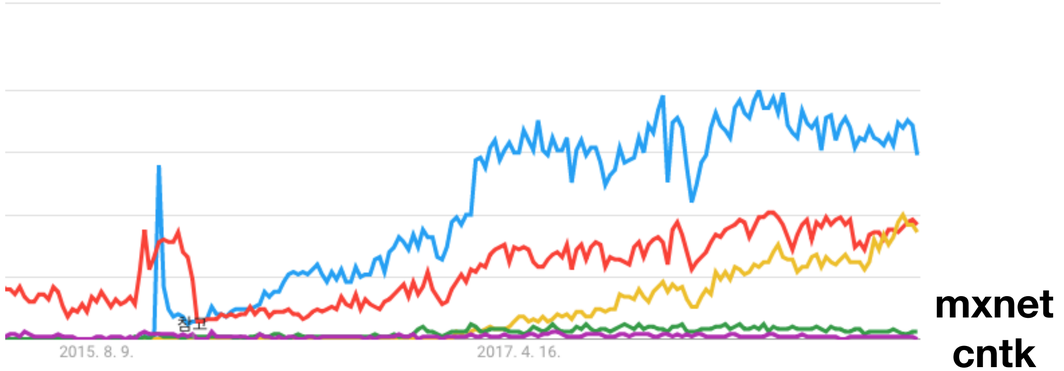
딥러닝 라이브러리

라이브러리	API	플랫폼	개발	년도
카페Caffe	파이썬, C++, 매트랩	리눅스, macOS, 윈도우	양칭 지아Yangqing Jia, 캘리포니아대학교 버클리 캠퍼스 버클리 비전 및 학습 센터(BVLC)	2013
딥러닝4j	자바, 스칼라, 클로저Clojure	리눅스, macOS, 윈도우, 안드로이드	아담 깁슨Adam Gibson, 조쉬 패터슨Josh Patterson	2014
H2O	파이썬, R	리눅스, macOS, 윈도우	H2O.ai	2014
MXNet	파이썬, C++, 그 외	리눅스, macOS, 윈도우, iOS, 안드로이드	DMLC	2015
텐서플로	파이썬, C++	리눅스, macOS, 윈도우, iOS, 안드로이드	구글	2015
씨아노Theano	파이썬	리눅스, macOS, iOS	몬트리올 대학	2010
토치Torch	C++, 루아Lua	리눅스, macOS, iOS, 안드로이드	로난 콜로버트Ronan Collobert, 코레이 카푸추글루Koray Kavukcuoglu, 클레멘트 파라벳Clement Farabet	2002

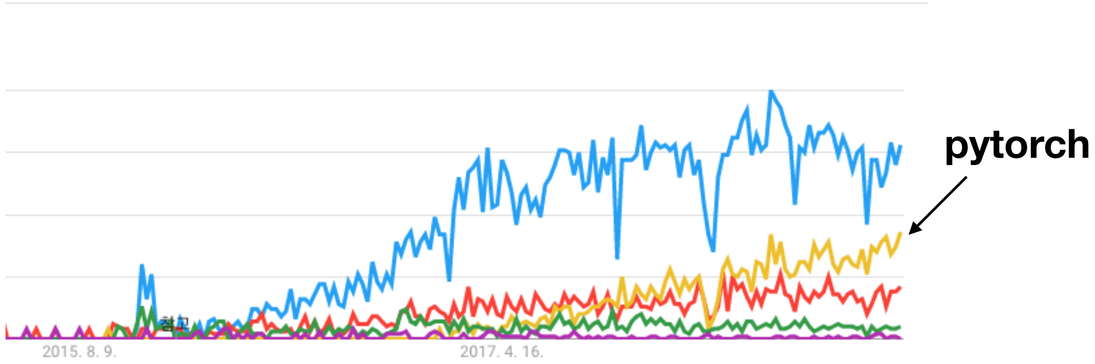
Global Trend



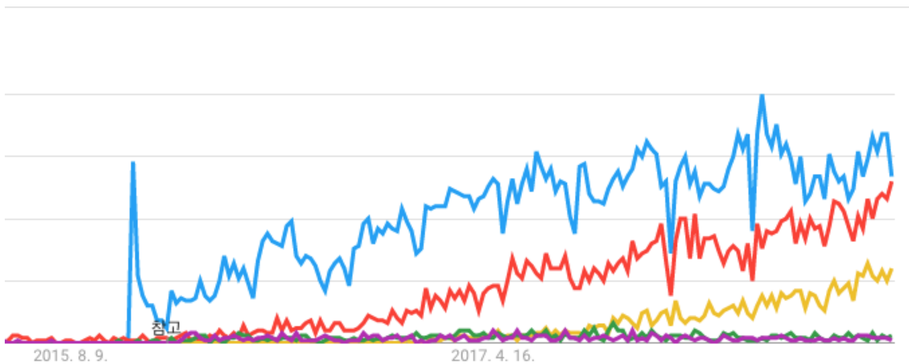
한국



미국



중국



일본

tensorflow vs keras

```
import tensorflow as tf
..
# 네트워크 구성
..
loss = ..
optimizer = ..
training_op = optimizer.minimize(loss)
..
sess = tf.Session()
sess.run(tf.global_variables_initializer())
..
# for loop
..
sess.run(loss, feed_dict={..})
```

```
from tensorflow import keras
# import keras
..
model = keras.Sequential()
# 네트워크 구성
..
model.compile(optimizer=.., loss=.., ..)
model.fit(..)
model.evaluate(..)
model.predict(..)
```

첫 번째 계산 그래프

```
import tensorflow as tf

x = tf.Variable(3, name="x")
y = tf.Variable(4, name="y")
f = x*x*y + y + 2
```

```
f
```

```
<tf.Tensor 'add_1:0' shape=() dtype=int32>
```

```
sess = tf.Session()
sess.run(x.initializer)
sess.run(y.initializer)
result = sess.run(f)
print(result)
```

```
42
```

tf.Operation 객체

tf.get_default_session().run(f)

```
with tf.Session() as sess:
    x.initializer.run()
    y.initializer.run()
    result = f.eval()
```

```
result
```

```
42
```

tf.get_default_session().run(f)

```
init = tf.global_variables_initializer()

with tf.Session() as sess:
    init.run()
    result = f.eval()
```

```
result
```

```
42
```

```
init = tf.global_variables_initializer()
```

```
sess = tf.InteractiveSession()
init.run()
result = f.eval()
print(result)
```

```
42
```

tf.Graph()

```
x1 = tf.Variable(1)
x1.graph is tf.get_default_graph()
```

```
True
```

```
graph = tf.Graph()
with graph.as_default():
    x2 = tf.Variable(2)

x2.graph is graph
```

```
True
```

```
x2.graph is tf.get_default_graph()
```

```
False
```

*** tf.reset_default_graph() : 기본 그래프 초기화**

노드 값의 생애주기

```
w = tf.constant(3)
x = w + 2
y = x + 5
z = x * 3

with tf.Session() as sess:
    print(y.eval())  # 10
    print(z.eval())  # 15
```

```
10
15
```

```
with tf.Session() as sess:
    y_val, z_val = sess.run([y, z])
    print(y_val)  # 10
    print(z_val)  # 15
```

```
10
15
```

x와 w를 두 번씩 평가

그래프 계산 결과는 사라지지만
변수(tf.Variable) 값은 유지됩니다.

한 번에 평가

선형 회귀

$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

```
X = tf.constant(housing_data_plus_bias, dtype=tf.float32, name="X")
y = tf.constant(housing.target.reshape(-1, 1), dtype=tf.float32, name="y")
XT = tf.transpose(X)
theta = tf.matmul(tf.matmul(tf.matrix_inverse(tf.matmul(XT, X)), XT), y)

with tf.Session() as sess:
    theta_value = theta.eval()
```

```
theta_value
```

넘파이 배열

```
array([[-3.7185181e+01],
       [ 4.3633747e-01],
       [ 9.3952334e-03],
       [-1.0711310e-01],
       [ 6.4479220e-01],
       [-4.0338000e-06],
       [-3.7813708e-03],
       [-4.2348403e-01],
       [-4.3721911e-01]], dtype=float32)
```

tensorflow

```
X = housing_data_plus_bias
y = housing.target.reshape(-1, 1)
theta_numpy = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)

print(theta_numpy)
```

```
[[-3.69419202e+01]
 [ 4.36693293e-01]
 [ 9.43577803e-03]
 [-1.07322041e-01]
 [ 6.45065694e-01]
 [-3.97638942e-06]
 [-3.78654265e-03]
 [-4.21314378e-01]
 [-4.34513755e-01]]
```

numpy

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(housing.data, housing.target.reshape(-1, 1))

print(np.r_[lin_reg.intercept_.reshape(-1, 1), lin_reg.coef_.T])
```

```
[[-3.69419202e+01]
 [ 4.36693293e-01]
 [ 9.43577803e-03]
 [-1.07322041e-01]
 [ 6.45065694e-01]
 [-3.97638942e-06]
 [-3.78654265e-03]
 [-4.21314378e-01]
 [-4.34513755e-01]]
```

scikit-learn

그래디언트 수동 계산

```
n_epochs = 1000
learning_rate = 0.01

X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")
y = tf.constant(housing.target.reshape(-1, 1), dtype=tf.float32, name="y")
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
error = y_pred - y
mse = tf.reduce_mean(tf.square(error), name="mse")
gradients = 2/m * tf.matmul(tf.transpose(X), error)
training_op = tf.assign(theta, theta - learning_rate * gradients)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        if epoch % 100 == 0:
            print("에포크", epoch, "MSE =", mse.eval())
        sess.run(training_op)

    best_theta = theta.eval()
```

$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(\theta^T \cdot \mathbf{x}^{(i)} - y^{(i)} \right) x_j^{(i)}$$

잘못된 방식: theta = theta - learning_rate * gradients

```
에포크 0 MSE = 9.161542
에포크 100 MSE = 0.7145004
에포크 200 MSE = 0.56670487
에포크 300 MSE = 0.5555718
에포크 400 MSE = 0.5488112
에포크 500 MSE = 0.5436363
에포크 600 MSE = 0.5396291
에포크 700 MSE = 0.5365092
에포크 800 MSE = 0.53406775
에포크 900 MSE = 0.5321473
```

```
best_theta
```

```
array([[ 2.0685523 ],
       [ 0.8874027 ],
       [ 0.14401656],
       [-0.34770882],
       [ 0.36178368],
       [ 0.00393811],
       [-0.04269556],
       [-0.6614529 ],
       [-0.6375279 ]], dtype=float32)
```

그래디언트 자동 계산

```
n_epochs = 1000
learning_rate = 0.01

X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")
y = tf.constant(housing.target.reshape(-1, 1), dtype=tf.float32, name="y")
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
error = y_pred - y
mse = tf.reduce_mean(tf.square(error), name="mse")
```

```
gradients = tf.gradients(mse, [theta])[0]
```

←

```
gradients = 2/m * tf.matmul(tf.transpose(X), error)
```

```
training_op = tf.assign(theta, theta - learning_rate * gradients)

init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        if epoch % 100 == 0:
            print("에포크", epoch, "MSE =", mse.eval())
        sess.run(training_op)

    best_theta = theta.eval()
```

자동 미분

기법	모든 그래디언트를 계산하기 위한 그래프 순회 수	정확도	임의의 코드 지원	비고
수치 미분	$n_{\text{inputs}} + 1$	낮음	지원	구현하기 쉬움
기호 미분	해당 없음	높음	미지원	상이한 그래프 생성
전진 모드 자동 미분	n_{inputs}	높음	지원	이원수 dual number 사용
후진 모드 자동 미분	$n_{\text{outputs}} + 1$	높음	지원	텐서플로가 사용하는 방식

옵티마이저 사용

```
n_epochs = 1000
learning_rate = 0.01

X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")
y = tf.constant(housing.target.reshape(-1, 1), dtype=tf.float32, name="y")
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
error = y_pred - y
mse = tf.reduce_mean(tf.square(error), name="mse")
```

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(mse)
```

```
gradients = 2/m * tf.matmul(tf.transpose(X), error)
training_op = tf.assign(theta, theta - learning_rate * gradients)
```

```
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        if epoch % 100 == 0:
            print("에포크", epoch, "MSE =", mse.eval())
        sess.run(training_op)

    best_theta = theta.eval()

print("best_theta:")
print(best_theta)
```

모멘텀 옵티마이저

```
n_epochs = 1000
learning_rate = 0.01

X = tf.constant(scaled_housing_data_plus_bias, dtype=tf.float32, name="X")
y = tf.constant(housing.target.reshape(-1, 1), dtype=tf.float32, name="y")
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
error = y_pred - y
mse = tf.reduce_mean(tf.square(error), name="mse")
```

```
optimizer = tf.train.MomentumOptimizer(learning_rate=learning_rate,
                                       momentum=0.9)
```

```
training_op = optimizer.minimize(mse)

init = tf.global_variables_initializer()
```

```
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        sess.run(training_op)

    best_theta = theta.eval()
```

플레이스홀더

```
A = tf.placeholder(tf.float32, shape=(None, 3))
B = A + 5
with tf.Session() as sess:
    B_val_1 = B.eval(feed_dict={A: [[1, 2, 3]]})
    B_val_2 = B.eval(feed_dict={A: [[4, 5, 6], [7, 8, 9]]})

print(B_val_1)
```

```
[[6. 7. 8.]]
```

```
print(B_val_2)
```

```
[[ 9. 10. 11.]
 [12. 13. 14.]]
```

```
X = tf.placeholder(tf.float32, shape=(None, n + 1), name="X")
y = tf.placeholder(tf.float32, shape=(None, 1), name="y")
```

```
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
error = y_pred - y
mse = tf.reduce_mean(tf.square(error), name="mse")
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(mse)

init = tf.global_variables_initializer()
```

```
n_epochs = 10
```

```
batch_size = 100
n_batches = int(np.ceil(m / batch_size))
```

...

```
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        for batch_index in range(n_batches):
            X_batch, y_batch = fetch_batch(epoch, batch_index, batch_size)
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch})

    best_theta = theta.eval()
```

SGD, 미니배치 방식 가능

모델 저장과 복원

저장

```
init = tf.global_variables_initializer()
saver = tf.train.Saver()

with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        if epoch % 100 == 0:
            print("에포크", epoch, "MSE =", mse.eval())
            save_path = saver.save(sess, "/tmp/my_model.ckpt")
        sess.run(training_op)

    best_theta = theta.eval()
    save_path = saver.save(sess, "/tmp/my_model_final.ckpt")
```

복원

```
saver = tf.train.import_meta_graph("/tmp/my_model_final.ckpt.meta")  # 그래프 구조를 로드합니다.
theta = tf.get_default_graph().get_tensor_by_name("theta:0") # 책에는 없습니다.

with tf.Session() as sess:
    saver.restore(sess, "/tmp/my_model_final.ckpt")  # 그래프 상태를 로드합니다.
    best_theta_restored = theta.eval() # 책에는 없습니다.
```

```
INFO:tensorflow:Restoring parameters from /tmp/my_model_final.ckpt
```

```
np.allclose(best_theta, best_theta_restored)
```

```
True
```

tf.summary

```
mse_summary = tf.summary.scalar('MSE', mse)
file_writer = tf.summary.FileWriter(logdir, tf.get_default_graph())
```

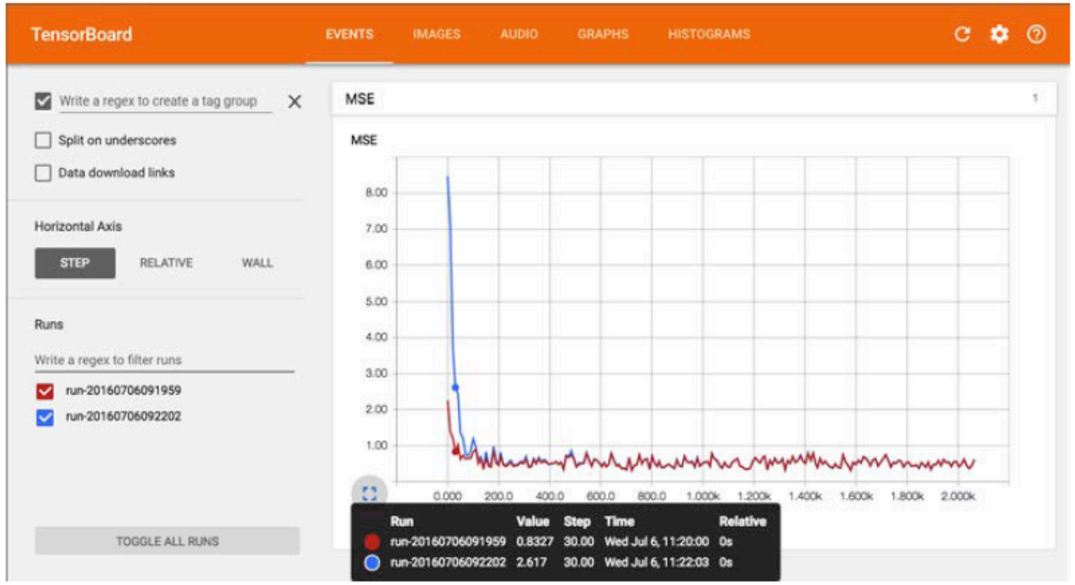
```
n_epochs = 10
batch_size = 100
n_batches = int(np.ceil(m / batch_size))
```

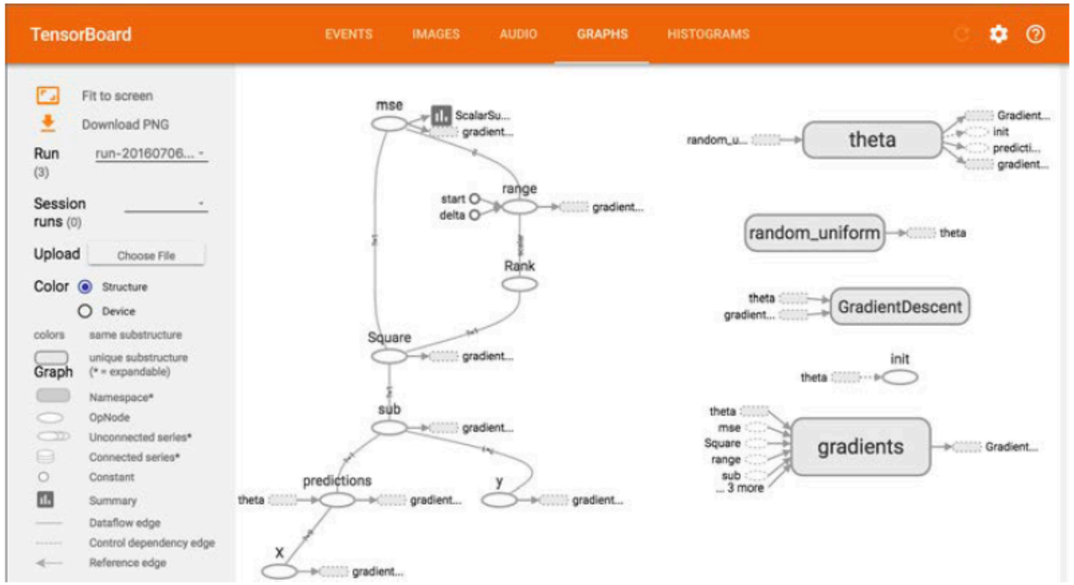
```
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        for batch_index in range(n_batches):
            X_batch, y_batch = fetch_batch(epoch, batch_index, batch_size)
            if batch_index % 10 == 0:
                summary_str = mse_summary.eval(feed_dict={X: X_batch, y: y_batch})
                step = epoch * n_batches + batch_index
                file_writer.add_summary(summary_str, step)
            sess.run(training_op, feed_dict={X: X_batch, y: y_batch})

    best_theta = theta.eval()
```

```
file_writer.close()
```

tensorboard

```
$ tensorboard --logdir tf_logs/
Starting TensorBoard  on port 6006
(You can navigate to http://0.0.0.0:6006)
```





tf.name_scope

```
X = tf.placeholder(tf.float32, shape=(None, n + 1), name="X")
y = tf.placeholder(tf.float32, shape=(None, 1), name="y")
theta = tf.Variable(tf.random_uniform([n + 1, 1], -1.0, 1.0, seed=42), name="theta")
y_pred = tf.matmul(X, theta, name="predictions")
```

```
with tf.name_scope("loss") as scope:
    error = y_pred - y
    mse = tf.reduce_mean(tf.square(error), name="mse")
```

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
training_op = optimizer.minimize(mse)

init = tf.global_variables_initializer()

mse_summary = tf.summary.scalar('MSE', mse)
file_writer = tf.summary.FileWriter(logdir, tf.get_default_graph())
```

```
n_epochs = 10
batch_size = 100
n_batches = int(np.ceil(m / batch_size))

with tf.Session() as sess:
    sess.run(init)
```

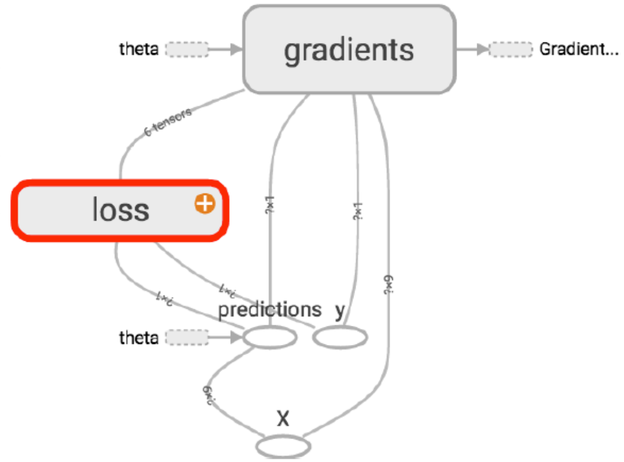
...

```
print(error.op.name)
```

```
loss/sub
```

```
print(mse.op.name)
```

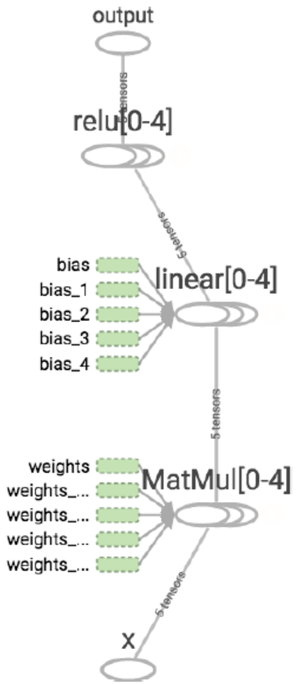
```
loss/mse
```




```
a1 = tf.Variable(0, name="a")        # name == "a"
a2 = tf.Variable(0, name="a")        # name == "a_1"

with tf.name_scope("param"):         # name == "param"
    a3 = tf.Variable(0, name="a")    # name == "param/a"

with tf.name_scope("param"):         # name == "param_1"
    a4 = tf.Variable(0, name="a")    # name == "param_1/a"
```

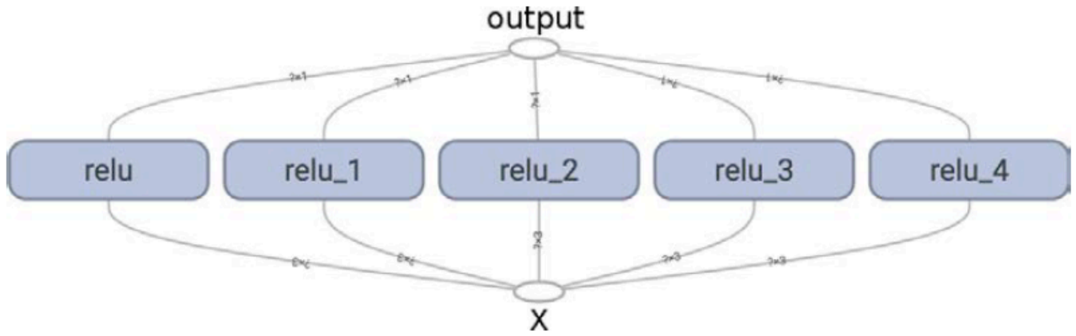
모듈화

```python
def relu(X):
    w_shape = (int(X.get_shape()[1]), 1)
    w = tf.Variable(tf.random_normal(w_shape), name="weights")
    b = tf.Variable(0.0, name="bias")
    z = tf.add(tf.matmul(X, w), b, name="z")
    return tf.maximum(z, 0., name="relu")

n_features = 3
X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = [relu(X) for i in range(5)]
output = tf.add_n(relus, name="output")
```




```python
def relu(X):
    with tf.name_scope("relu"):
        w_shape = (int(X.get_shape()[1]), 1)
        w = tf.Variable(tf.random_normal(w_shape), name="weights")
        b = tf.Variable(0.0, name="bias")
        z = tf.add(tf.matmul(X, w), b, name="z")
        return tf.maximum(z, 0., name="max")
```

```python
n_features = 3
X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = [relu(X) for i in range(5)]
output = tf.add_n(relus, name="output")
```



단순한 변수 공유

```
def relu(X, threshold):
    with tf.name_scope("relu"):
        w_shape = (int(X.get_shape()[1]), 1)
        w = tf.Variable(tf.random_normal(w_shape), name="weights")
        b = tf.Variable(0.0, name="bias")
        z = tf.add(tf.matmul(X, w), b, name="z")
        return tf.maximum(z, threshold, name="max")

threshold = tf.Variable(0.0, name="threshold")
X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = [relu(X, threshold) for i in range(5)]
output = tf.add_n(relus, name="output")
```

```
def relu(X):
    with tf.name_scope("relu"):
        if not hasattr(relu, "threshold"):
            relu.threshold = tf.Variable(0.0, name="threshold")
        w_shape = int(X.get_shape()[1]), 1
        w = tf.Variable(tf.random_normal(w_shape), name="weights")
        b = tf.Variable(0.0, name="bias")
        z = tf.add(tf.matmul(X, w), b, name="z")
        return tf.maximum(z, relu.threshold, name="max")
```

```
X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
relus = [relu(X) for i in range(5)]
output = tf.add_n(relus, name="output")
```

tf.variable_scope()

```
with tf.variable_scope("relu"):
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
```

```
with tf.variable_scope("relu", reuse=True):
    threshold = tf.get_variable("threshold")
```

```
with tf.variable_scope("relu") as scope:
    scope.reuse_variables()
    threshold = tf.get_variable("threshold")
```

```
def relu(X):
    with tf.variable_scope("relu", reuse=True):
        threshold = tf.get_variable("threshold")
        w_shape = int(X.get_shape()[1]), 1                          # 책에는 없습니다.
        w = tf.Variable(tf.random_normal(w_shape), name="weights")  # 책에는 없습니다.
        b = tf.Variable(0.0, name="bias")                           # 책에는 없습니다.
        z = tf.add(tf.matmul(X, w), b, name="z")                    # 책에는 없습니다.
        return tf.maximum(z, threshold, name="max")

X = tf.placeholder(tf.float32, shape=(None, n_features), name="X")
with tf.variable_scope("relu"):
    threshold = tf.get_variable("threshold", shape=(),
                                initializer=tf.constant_initializer(0.0))
relus = [relu(X) for relu_index in range(5)]
output = tf.add_n(relus, name="output")
```

examples

```
reset_graph()

with tf.variable_scope("my_scope"):
    x0 = tf.get_variable("x", shape=(), initializer=tf.constant_initializer(0.))
    x1 = tf.Variable(0., name="x")
    x2 = tf.Variable(0., name="x")

with tf.variable_scope("my_scope", reuse=True):
    x3 = tf.get_variable("x")
    x4 = tf.Variable(0., name="x")

with tf.variable_scope("", default_name="", reuse=True):
    x5 = tf.get_variable("my_scope/x")

print("x0:", x0.op.name)
print("x1:", x1.op.name)
print("x2:", x2.op.name)
print("x3:", x3.op.name)
print("x4:", x4.op.name)
print("x5:", x5.op.name)
print(x0 is x3 and x3 is x5)
```

```
x0: my_scope/x
x1: my_scope/x_1
x2: my_scope/x_2
x3: my_scope/x
x4: my_scope_1/x
x5: my_scope/x
True
```

수치 미분

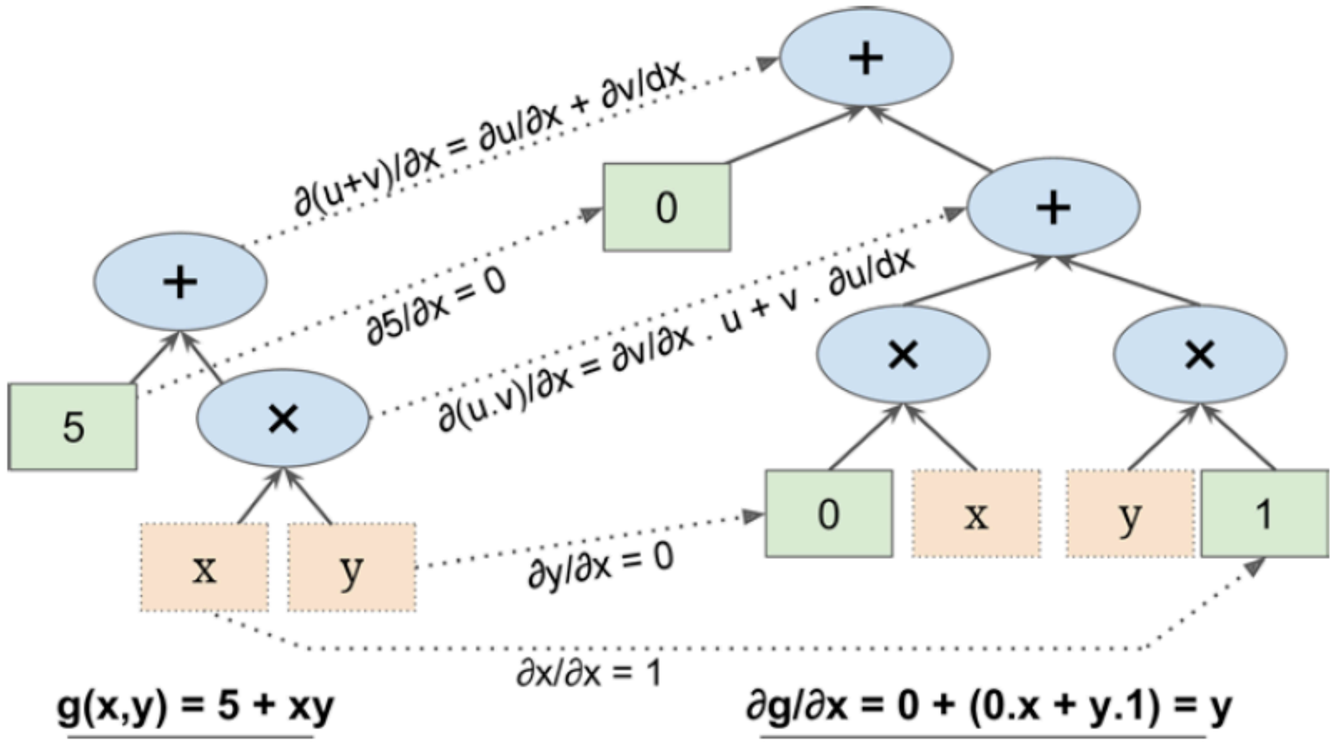
- 정확도가 낮음
- 입력 개수 + 1 만큼 실행

$$h'(x_0) = \lim_{x \to x_0} \frac{h(x) - h(x_0)}{x - x_0}$$

$$= \lim_{\varepsilon \to 0} \frac{h(x_0 + \varepsilon) - h(x_0)}{\varepsilon}$$

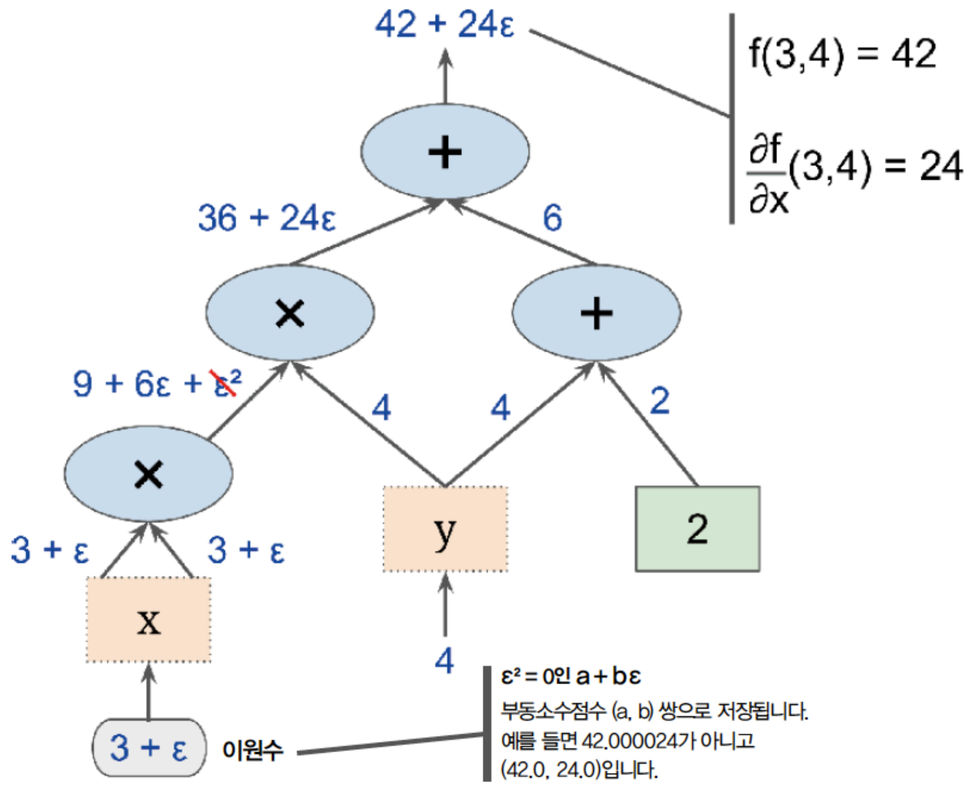
기호 미분

- 임의의 함수에 대해 적용하기 어려움



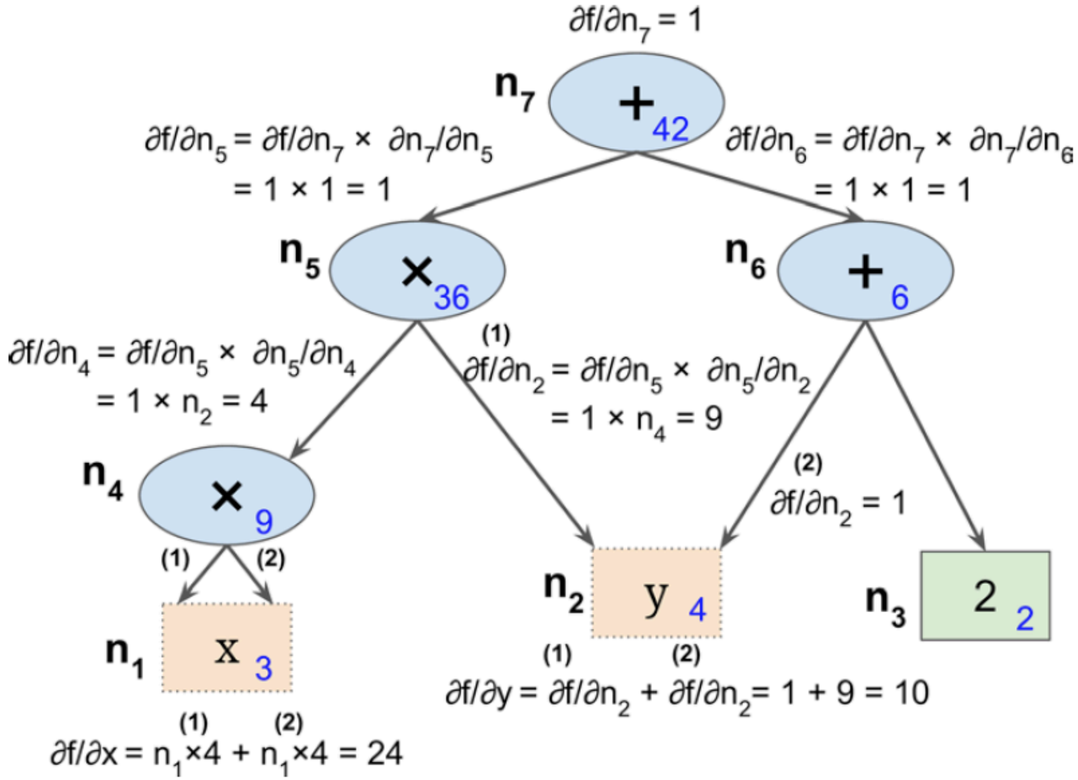
전진 모드 자동 미분

- 이원수 사용 $h(a+b\varepsilon) = h(a) + b \times h'(a)\varepsilon$.
- 입력 개수만큼 실행



후진 모드 자동 미분

- 미분의 연쇄 법칙(chain rule) 사용
- 출력 개수 + 1 만큼 실행



감사합니다