TensorFlow Abstractions and Simplification

Chapter Overview

프로그래밍의 맥락에서 추상화(abstraction)라는 용어는 원래 코드를 특정한 목적으로 일반화하여 기존 코드의 ‘위에 올라가는’ 코드의 계층을 이르는 말이다. 관련 있는 몇몇 고차원 기능을 묶는 방식의 재구성을 통해 코드를 묶고 감싸서 추상화한다. 다음 코드를 보자.

def weight\_variable(shape):

initial = tf.truncated\_normal(shape, stddev=0.1)

return tf.Variable(initial)

def bias\_variable(shape):

initial = tf.constant(0.1, shape=shape)

return tf.Variable(initial)

def conv2d(x, W):

return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def conv\_layer(input, shape):

W = weight\_variable(shape)

b = bias\_variable([shape[3]])

h = tf.nn.relu(conv2d(input, W) + b)

hp = max\_pool\_2x2(h)

return hp

def max\_pool\_2x2(x):

return tf.nn.max\_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

x = tf.placeholder(tf.float32, shape=[None, 784])

x\_image = tf.reshape(x, [-1, 28, 28, 1])

h1 = conv\_layer(x\_image, shape=[5, 5, 1, 32])

h2 = conv\_layer(h1, shape=[5, 5, 32, 64])

h3 = conv\_layer(h2, shape=[5, 5, 64, 32])

기본 tensorflow에서 합성곱 계층을 생성하려면 입력과 원하는 출력의 형태에 따라 가중치 및 편향값을 정의하고 초기화한 후 정의된 strider와 padding을 사용해 합성곱 연산을 적용하고 마지막으로 활성화 함수 연산을 추가해야 한다. 이러한 코드는 실수를 범하기 쉽고 이런 프로세스를 여러 번 반복하는 것은 지루한 일이 될 수도 있고 좀더 효율적으로 했으면 하는 생각이 들 것이다.

위의 코드는 중복 작업을 일부 제거하는 함수를 사용해 추상화를 시도해 본 결과물이다. 다음 코드와 가독성을 비교해보자.

x = tf.placeholder(tf.float32, shape=[None, 784])

x\_image = tf.reshape(x, [-1, 28, 28, 1])

W1 = tf.truncated\_normal([5, 5, 1, 32], stddev=0.1)

b1 = tf.constant(0.1, shape=[32])

h1 = tf.nn.relu(tf.nn.conv2d(x\_image, W1, strides=[1, 1, 1, 1], padding='SAME') + b1)

hp1 = tf.nn.max\_pool(h1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W2 = tf.truncated\_normal([5, 5, 32, 64], stddev=0.1)

b2 = tf.constant(0.1, shape=[64])

h2 = tf.nn.relu(tf.nn.conv2d(hp1, W2, strides=[1, 1, 1, 1], padding='SAME') + b2)

hp2 = tf.nn.max\_pool(h2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W3 = tf.truncated\_normal([5, 5, 64, 32], stddev=0.1)

b3 = tf.constant(0.1, shape=[32])

h3 = h1 = tf.nn.relu(tf.nn.conv2d(hp2, W3, strides=[1, 1, 1, 1], padding='SAME') + b3)

hp3 = tf.nn.max\_pool(h3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

고작 3개의 계층밖에 없는데도 불구하고 만들어진 코드는 보기에 꽤 복잡하고 혼란스럽다. 전형적인 중간 크기의 코드를 넘어서는 더 길고 복잡한 코드는 보통 추상화 라이브러리로 ‘감싸서’ 제공하는 것이 좋다. tensorflow에서 사용 가능한 추상화 라이브러리 중 contrib.learn을 사용하면 선형회귀모형을 정의하고 학습하는 핵심 기능을 다음과 같이 단 두 줄로 구현할 수 있다.

regressor = learn.LinearRegressor(feature\_columns=feature\_columns, optimizer=optimizer)

regressor.fit(X, Y, steps=200, batch\_size=506)