텐서플로 (TensorFlow)

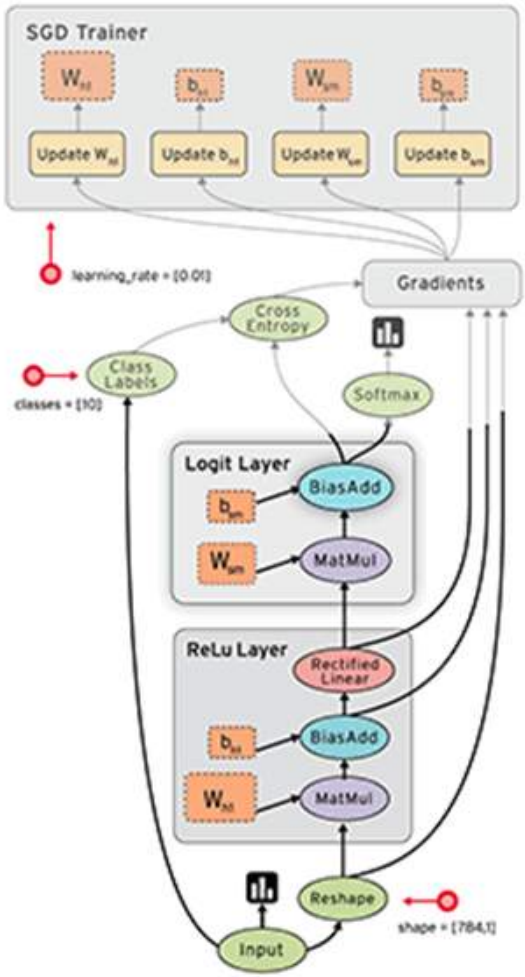
v1.x 기본 문법

김 대 환

2022.

텐서플로우 특징

- **텐서(Tensor)**를 활용한 그래프 수치 연산 도구
- 그래프
  - 노드(Node)와 엣지(Edge)로 구성되는 기하학적 모양
  - 노드는 연산과 데이터를 정의
  - 엣지는 노드들을 연결하여 데이터의 흐름을 표현
  - 텐서는 다차원 배열의 형태를 띤 데이터
    - ✓ 노드에서 텐서를 연산하고 엣지를 통해 텐서가 이동
- 텐서 플로우 버전
  - 1.x대 버전과 2.x대 버전이 사용 됨
  - 문법적 내용이 상이하여 호환성이 없음
  - 현재까지는 1.x 대 버전의 참고 자료가 많음



텐서플로우 문법 (v1.x)

- 노드로 연산 및 데이터를 정의하고, 데이터 흐름을 위해 엣지를 구성하여 그래프 생성
- **세션(Session)**
  - 세션 실행으로 그래프의 시작점부터 모든 연산과 동작이 수행
  - 엣지를 통해 데이터 출력
  - 세션 실행 방법:

```
sess = tf.Session()
sess.run(실행할 코드)

혹은,
with tf.Session() as sess
  sess.run(실행할 코드)
```

  - 첫 번째 방식으로 세션을 실행할 경우에는 `**sess.close()**` 라는 세션 종료 코드를 삽입해야 함

텐서플로 문법 (v1.x)

- 세션 실행 예

```
import tensorflow as tf

# 상수 선언
hello = tf.constant("Hellow World")
print(hello)

a = tf.constant(10)
b = tf.constant(32)
c = a+b
print(c)


Output:
Tensor("Const:0", shape=(), dtype=string)
Tensor("add:0", shape=(), dtype=int32))


# 텐서플로의 자료형 정보만 출력된다
```

```
# 그래프를 실행할 세션 생성
sess = tf.Session()

# sess.run(): 텐서 그래프 (변수, 수식)을 실행
# 문자열을 unicode가 기본이므로,
# str() 함수로 encoding 처리를 해야 한다.
# bytes type이 아닌 unicode type으로 변환된다.

print(sess.run(hello))
print(str(sess.run(hello), encoding = "utf-8"))
print(sess.run([a, b, c]))

sess.close()    # 세션 종료

Output:
b'Hellow World'
Hellow World
[10, 32, 42]

# Session 객체를 만든고, run() 매서드를 호출해야 비로소 처리된다.
```

텐서플로 문법 (v1.x)

- 세션 실행 예

```
import tensorflow as tf
node1 = tf.constant(3.0)
node2 = tf.constant(4.0)
node3 = tf.add(node1, node2)

# with 문은 sess.close() 가 자동 호출
with tf.Session() as sess:
    print("node3: ", node3)
    print("sess.run(node3): ", sess.run(node3))

# sess.run을 실행하지 않으면 노드 구조만 출력
# sess.run을 실행해야 실제 결과를 출력

Output:
node3: Tensor("Add_1:0", shape=(), dtype=float32)
sess.run(node3): 7.0
```

텐서플로 문법 (v1.x)

- 패치(**Fetch**)
  - 피드(**feed**)와 플레이스홀드(**placehold**)를 이용
    - ✓ 플레이스홀드를 통해 데이터 구조를 만든 다음, 피드는 `**feed_dict**` 구문을 이용하여 데이터 전달
  - 플레이스홀더는 노드에 들어갈 데이터 형식을 미리 정의 함
  - 다음 예는 플레이스홀더를 이용하여 float32 형 자료를 input_01 이라는 이름으로 정의한 다음, 세션이 실행될 때 `**feed_dict**`를 통해 만들어진 자료 구조로 데이터를 전달하고 있다

텐서플로 문법 (v1.x)

- 패치(**Fetch**) 사용 예

```
import tensorflow as tf

# placeholder: 그래프의 입력 변수
input01 = tf.placeholder(tf.float32)
input02 = tf.placeholder(tf.float32)
output = tf.multiply(input01, input02)

print("input01:", input01)
print("input02:", input02)

with tf.Session() as sess:
    print(sess.run(output, feed_dict={input01:3.0, input02:5.0}))
    print(sess.run(output, feed_dict={input01:0.0, input02:6.0}))
    print(sess.run(output, feed_dict={input01:[2.0], input02:[6.0]}))

# 같은 그래프 구조에 feed_dict= {} 구문을 활용하여 데이터를 입력한다.
# 데이터에 따른 실행 결과를 확인할 수 있다.
```

```
#출력결과:

input01: Tensor("Placeholder_2:0",
dtype=float32) input02:
Tensor("Placeholder_3:0", dtype=float32)
15.0 0.0 [12.]
```

텐서플로 문법 (v1.x)

- **변수(Variable)**
  - 학습 과정을 통해 지속적으로 변경되는 값을 저장하는 곳을 의미
  - 초기화는 노드에 값을 입력한다는 의미
  - 변수를 사용할 때는 미리 초기화가 반드시 필요

```
# 변수 선언
tf.Variable(초기값, 타입)

# 변수 초기화
tf.global_variables_initializer()
```

텐서플로 문법 (v1.x)

- 변수**(Variable)** 사용 예

```
import tensorflow as tf

# None: 크기가 정해지지 않았음을 의미한다.
X = tf.placeholder(tf.float32, [None, 3])
print(X)

# X placeholder에 넣을 값
# 두 번째 차원의 요소의 개수는 3
x_data = [[1, 2, 3], [4, 5, 6]]

# tf.Variable() 그래프를 계산하면서 최적화할 함수
# tf.random_normal(): 각 변수들의 초기 값을
# 정규분포 랜덤 값으로 초기화
W = tf.Variable(tf.random_normal([3, 2]))
b = tf.Variable(tf.random_normal([2, 1]))

# 입력 값과 변수들을 계산할 수식 작성
# tf.matmul 처럼 mat로 시작하는 함수로 행렬계산을 수행
# 행렬곱셈
expr = tf.matmul(X,W) + b
sess = tf.Session()
```

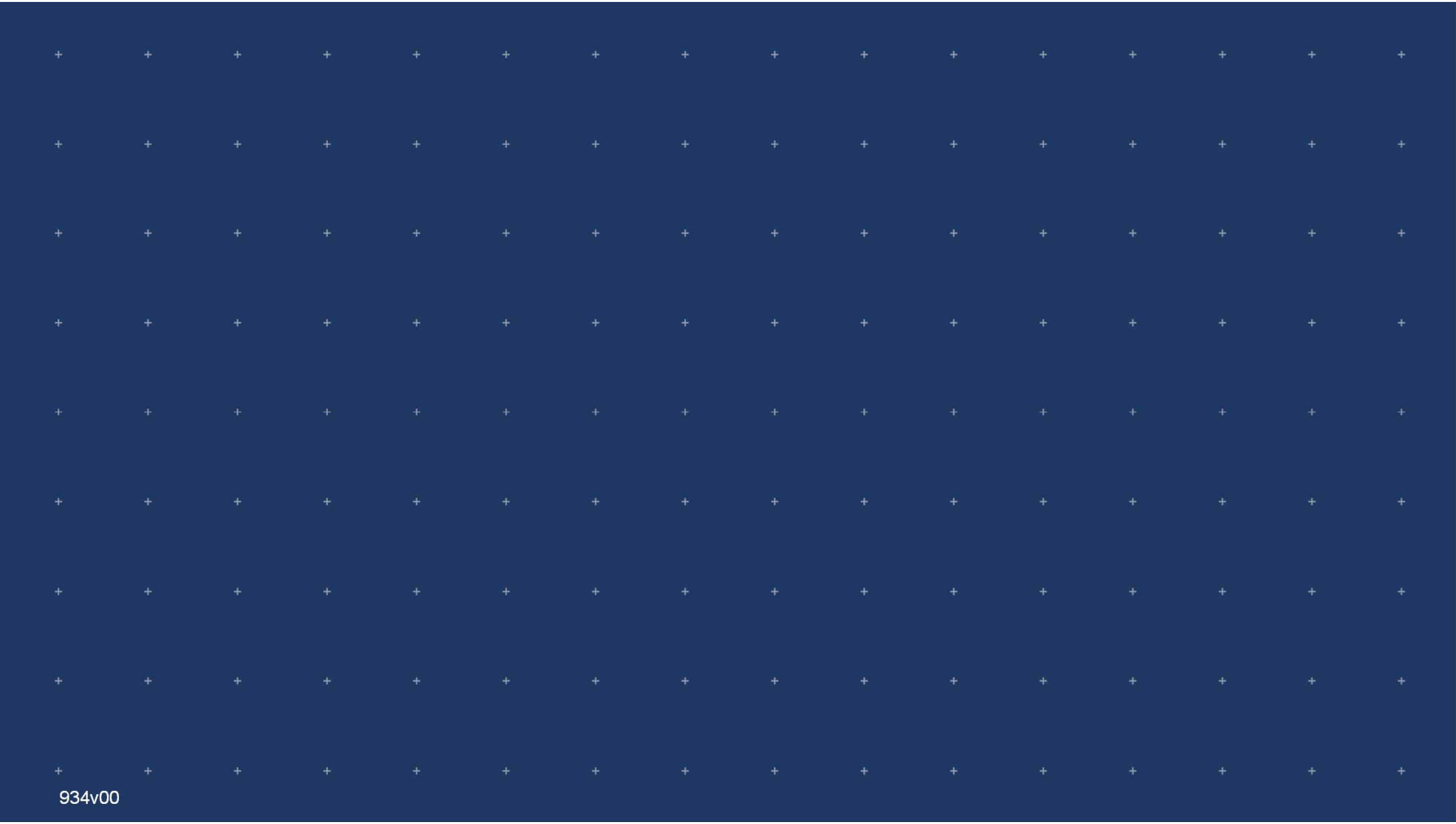
```
# 변수 초기화
sess.run(tf.global_variables_initializer())
print(x_data)
print(sess.run(W))
print(sess.run(b))

# 수식에 값을 전달하는 방법
# sess.run (수식, feed_dict={변수: 값})
print(sess.run(expr, feed_dict={X: x_data}))

# 세션 종료
sess.close()

#출력결과:

[[1, 2, 3], [4, 5, 6]]
[[ 1.7435963 0.5700547 ] [ 1.9454948 0.32510278]
[-0.9426243 -0.14288588]]
...
...
```


934v00