

실습 I [텐서플로우의 변수 텐서 생성 코드]

텐서플로우에서 상수 텐서를 생성하는 방법은 매우 간단합니다. `tf.constant()` 함수를 사용하면 됩니다.

이 함수는 입력으로 리스트나 넘파이 배열을 받아 상수 텐서를 생성합니다. 다음 코드는 3x4 크기의 상수 텐서를 생성하는 예시입니다.

`tf.constant()` 함수에 리스트를 입력으로 전달하면 리스트의 요소로 채워진 텐서가 생성됩니다. **->값 변경 불가능**

```
import tensorflow as tf
x = tf.constant([[1, 2, 3, 4],
                 [5, 6, 7, 8],
                 [9, 10, 11, 12]])

print(x)

a = tf.constant(2)
b = tf.constant(3)
c = a + b
print(c)
```

```
tf.Tensor(
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]], shape=(3, 4), dtype=int32)

tf.Tensor(5, shape=(), dtype=int32)
```

실습 2[텐서플로우의 변수 텐서 생성 코드]

텐서플로우에서 변수 텐서를 생성하려면 `tf.Variable()` 함수를 사용합니다. 이 함수는 입력으로 초기 값을 가지는 텐서를 받아 변수 텐서를 생성합니다. 변수 텐서는 값을 변경할 수 있습니다. -> 값 변경 가능

```
import tensorflow as tf
x = tf.Variable(initial_value=[[1, 2, 3, 4],
                              [5, 6, 7, 8],
                              [9, 10, 11, 12]], name='hh')
print(x)

a = tf.Variable(initial_value=2)
b = tf.Variable(initial_value=3)
c = a + b
print()
print(c)
```

```
<tf.Variable 'hh:0' shape=(3, 4) dtype=int32, numpy=
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])>

tf.Tensor(5, shape=(), dtype=int32)
```

실습 3[텐서 연산자 생성 코드]

```
import tensorflow as tf
x = tf.constant([[1, 2]])
y = tf.constant([[1, 2]])

negMatrix = tf.negative(x) → 텐서의 값을 음수로 바꾼 텐서를 반환
print(negMatrix)
negMatrix.numpy()

z=tf.add(x, y)
tf.subtract(x, y)
tf.multiply(x, y)
tf.pow(x, y)
tf.exp(1.1) #float지정
tf.sqrt(0.3) #float지정
tf.divide(x, y)
tf.truediv(x, y)
tf.math.floordiv(x, y)
tf.math.mod(x, y)
print(z.numpy())

tensor1 = tf.constant([1, 2, 3], dtype=tf.int32)
tensor2 = tf.ones([3], dtype=tf.int32) → 모든 요소가 1로 채워진 텐서를 생성하는 함수
result = tensor1 + tensor2
print(result)
```

pow: 거듭 제곱 (x의 요소를 y의 대응 되는 요소 만큼 거듭 제곱)
exp: 지수 함수 결과 반환
sqrt: 제곱근
divide: 나누기
truediv: 나누기
math.floordiv: 소수점 이하를 버린 몫을 반환
math.mod: 나눈 나머지를 반환

```
tf.Tensor([[-1 -2]], shape=(1, 2), dtype=int32)
[[2 4]]
tf.Tensor([2 3 4], shape=(3,), dtype=int32)
```

```
tf.Tensor([1 1 1], shape=(3,), dtype=int32)
```


실습4[텐서 변환 I]

텐서플로우는 다양한 텐서 변환 함수를 제공합니다. 텐서의 형태, 데이터 타입, 값을 변환할 수 있습니다.

- `tf.reshape()` 함수를 사용하면 텐서의 형태를 변경할 수 있습니다.
- `tf.cast()` 함수를 사용하면 텐서의 데이터 타입을 변경할 수 있습니다.
- `tf.convert_to_tensor()` 함수를 사용하면 다양한 데이터 소스(배열, 리스트 등)를 텐서의 형태로 변경할 수 있습니다.

```
import tensorflow as tf
import numpy as np

w1=[1.0, 2.0], [3.0, 4.0]
w2=np.array([[1.0, 2.0],
             [3.0, 4.0]], dtype=np.float32)
t1=tf.convert_to_tensor(w1, dtype=tf.float32)
t2=tf.convert_to_tensor(w2, dtype=tf.float32)
print(t1)
print(t2)

# 부울형 텐서 생성
bool_tensor = tf.constant([True, False, True], dtype=tf.bool)
# 부울형 텐서를 정수형으로 변환 (True -> 1, False -> 0)
int_tensor = tf.cast(bool_tensor, dtype=tf.int32)
print(int_tensor)

# 정수형 텐서 생성
x = tf.constant([1, 2, 3, 4], dtype=tf.int32)
# 정수형 텐서를 실수형으로 변환
x_float = tf.cast(x, dtype=tf.float32)
print(x_float)

# 1차원 텐서 생성
tensor_1d = tf.constant([1, 2, 3, 4, 5, 6], dtype=tf.int32)
# 1차원 텐서를 2x3의 2차원 텐서로 변환
tensor_2d = tf.reshape(tensor_1d, shape=(2, 3))
print(tensor_2d)
# 2차원 텐서 생성
tensor_2d = tf.constant([[1, 2, 3], [4, 5, 6]], dtype=tf.int32)
# 2차원 텐서를 3차원 텐서로 변환
tensor_3d = tf.reshape(tensor_2d, shape=(2, 1, 3))
print(tensor_3d)
# 예시로, 8개의 샘플이 있고 각 샘플은 3개의 피치를 가짐
tensor_batch = tf.constant([[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12],
                             [13, 14, 15], [16, 17, 18], [19, 20, 21], [22, 23, 24]]],
                             dtype=tf.int32)
# 배치 크기를 4로 변경 (2x4 배치로 변경)
reshaped_batch = tf.reshape(tensor_batch, shape=(4, 2, 3))
print(reshaped_batch)
```

```
tf.Tensor(
[[1. 2.]
 [3. 4.]], shape=(2, 2), dtype=float32)
tf.Tensor(
[[1. 2.]
 [3. 4.]], shape=(2, 2), dtype=float32)
-----
tf.Tensor([1 0 1], shape=(3,), dtype=int32)
-----
tf.Tensor([1. 2. 3. 4.], shape=(4,), dtype=float32)
-----
```

```
tf.Tensor(
[[1 2 3]
 [4 5 6]], shape=(2, 3), dtype=int32)
-----
tf.Tensor(
[[[1 2 3]]], shape=(2, 1, 3), dtype=int32)
-----
tf.Tensor(
[[[ 1  2  3]
   [ 4  5  6]]], shape=(2, 1, 3), dtype=int32)
-----
[[[ 7  8  9]
   [10 11 12]]]
-----
[[[13 14 15]
   [16 17 18]]]
-----
[[[19 20 21]
   [22 23 24]]], shape=(4, 2, 3), dtype=int32)
-----
```

Shape 예시	분류	설명	Sample
(8,)	1차원 텐서	배열 형태로 8개의 요소로 구성되어 있습니다.	[1, 2, 3, 4, 5, 6, 7, 8]
(2, 4)	2차원 텐서	두 개의 그룹으로 나누고, 각 그룹은 4개의 요소를 가지고 있는 구조입니다.	[[1, 2, 3, 4], [5, 6, 7, 8]]
(2, 2, 2)	3차원 텐서	두 개의 그룹으로 나누고, 각 그룹별로 4개의 요소를 2개 그룹으로 분할하여 가지고 있습니다.	[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]

실습5[스파이크 감지]

```
import tensorflow as tf
raw_data = [1., 2., 8., -1., 0., 5.5, 6., 13]
spikes = tf.Variable([False] * len(raw_data), name='spikes') → 초기값 모두 False로 지정
spikes.numpy()

for i in range(1, len(raw_data)):
    if raw_data[i] - raw_data[i-1] > 5:
        spikes_val = spikes.numpy()
        spikes_val[i] = True
        spikes.assign(spikes_val) → For 루프를 통해 raw_data 의 차이가 5보다
                                   크면 spikes = True로 반환

print(spikes.numpy())
```

↓

i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=7
False	False	True	False	False	True	False	True
	2.-1./	8.-2./	-1-8/	0-(-1)/	5.5-0/	6-5.5/	13-6

```
import tensorflow as tf
raw_data = [1.,2.,8.,-1.,0.,5.5,6.,13]
spikes = tf.Variable([False] * len(raw_data), name='spikes')
spikes.numpy()
```

```
for i in range(1, len(raw_data)):
    if raw_data[i] - raw_data[i-1] > 5:
        spikes_val = spikes.numpy()
        spikes_val[i] = True
        spikes.assign(spikes_val)
spikes.numpy()
```

자신의 경로 입력

```
import os
directory = "C:/Users/dahae/Desktop/machine"
if not os.path.exists(directory):
    os.makedirs(directory)

checkpoint = tf.train.Checkpoint(spikes=spikes)
save_path = checkpoint.save("C:/Users/dahae/Desktop/machine/spikes.ckpt")
```



디렉토리 생성 및 체크 포인트 저장

```
import tensorflow as tf
nspikes = tf.Variable([False]*8, name='spikes')
nspikes.numpy()
new_checkpoint = tf.train.Checkpoint(spikes=nspikes)
new_checkpoint.restore(save_path)
result = nspikes.numpy()
print(result)
```

[False False True False False True False True]

실습[선형 회귀 코드]

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam

x_train = np.linspace(-1, 1, 1000)  → 입력 data (-1~1 까지 1000개의 data)
y_train = 2 * x_train + np.random.randn(*x_train.shape) * 0.33

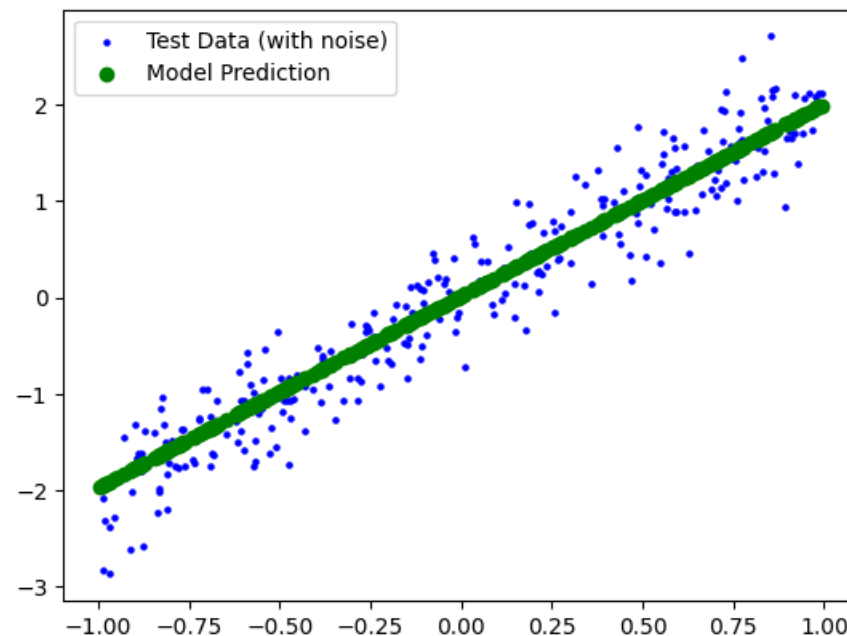
model = models.Sequential([
    layers.Input(shape=(1,)), → Model 구축
    layers.Dense(1)
])

x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.3)
↓
x_train = tf.convert_to_tensor(x_train, dtype=tf.float32)
y_train = tf.convert_to_tensor(y_train, dtype=tf.float32)
x_test = tf.convert_to_tensor(x_test, dtype=tf.float32)
y_test = tf.convert_to_tensor(y_test, dtype=tf.float32)
test 30%
train 70 %
데이터 나눔

# Compile the model
optimizer = Adam(learning_rate=0.001) → 최적화
model.compile(optimizer=optimizer, loss='mse')
model.summary()

history = model.fit(x_train, y_train, epochs=100, batch_size=16) #, validation_data=(x_test, y_test)
y_pred = model.predict(x_test)

plt.scatter(x_test, y_test, color='blue', s=5, label='Test Data (with noise)')
plt.scatter(x_test, y_pred, color='green', linewidth=1, label='Model Prediction')
plt.legend()
plt.show() → 시각화
```



→ 첫 모델 학습, 예측

실습2[비선형 다항 회귀 코드]

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import SGD

learning_rate = 0.0085
training_epochs = 40
x_train = np.linspace(-1, 1, 1001)
iterations = 0
num_coeffs = 6
trY_coeffs = [1, 2, 3, 4, 5, 6]
y_train = 0

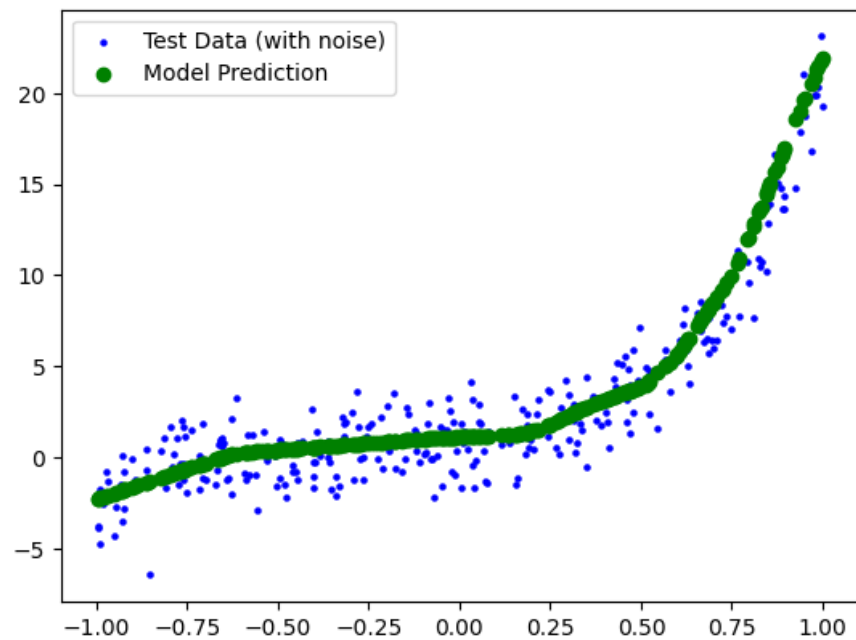
for i in range(num_coeffs):
    y_train += trY_coeffs[i] * np.power(x_train, i)
y_train += np.random.randn(*x_train.shape) * 1.5

model = models.Sequential([
    layers.Input(shape=(1,)),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(1)
])

x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.3)

x_train = tf.convert_to_tensor(x_train, dtype=tf.float32)
y_train = tf.convert_to_tensor(y_train, dtype=tf.float32)
x_test = tf.convert_to_tensor(x_test, dtype=tf.float32)
y_test = tf.convert_to_tensor(y_test, dtype=tf.float32)


# Compile the model
optimizer = SGD(learning_rate=learning_rate, momentum = 0.5)
model.compile(optimizer=optimizer, loss='mse')
model.summary()
history = model.fit(x_train, y_train, epochs=training_epochs, batch_size=16)
y_pred = model.predict(x_test)
plt.scatter(x_test, y_test, color='blue', s=5, label='Test Data (with noise)')
plt.scatter(x_test, y_pred, color='green', linewidth=1, label='Model Prediction')
plt.legend()
plt.show()
```



Google

아나콘다 다운로드

전체 이미지 동영상 쇼핑 뉴스 지도 웹 : 더보기 도구

1)  **Anaconda**
<https://www.anaconda.com> > download

Download Anaconda Distribution

Download Anaconda's open-source Distribution today. Discover the easiest way to perform Python/R data science and machine learning on a single machine.

2) **Distribution**
Free Download*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- ✓ Easily search and install thousands of data science, machine learning, and AI packages
- ✓ Manage packages and environments from a desktop application or work from the command line
- ✓ Deploy across hardware and software platforms
- ✓ Distribution Installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 250 employees requires a Business or Enterprise license. [See Pricing](#)

Provide email to download Distribution

Email Address:

☐ Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent [at any time](#).

By continuing, I agree to Anaconda's [Privacy Policy](#) and [Terms of Service](#).

Submit >


Only registration

3) **Download Now**

For installer assistance, refer to [troubleshooting](#)

Download Distribution by choosing the proper installer for your machine.

Download



1. Anaconda에서 환경 만들기

```
conda create -n machine python=3.7
```

원하는 환경 이름

```
conda activate machine
```

2. Tensorflow , Jupyter notebook 설치

```
(machine) C:\#Users\dahae>pip install tensorflow  
collecting tensorflow
```

```
pip install tensorflow  
pip install jupyter notebook
```

3. Jupyter notebook에 코드 실행

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

```
[9]: print(tf.__version__)
```

```
2.11.0
```