



Day55; 20221123

날짜	@2022년 11월 23일
유형	@2022년 11월 23일
태그	

GitHub - u8yes/AI

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/u8yes/ai>

u8yes/AI



1 Contributor 0 Issues 0 Stars 0 Forks

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ed534ed2-044c-48d6-a2f6-1b659411c2a9/01_linear_regression_20221122-1123.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bd6ae83c-15e5-4682-a498-1e2c24d768d0/03_%EC%84%A0%ED%98%95_%ED%9A%8C%EA%B7%80%EB%B6%84%EC%84%9D.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/417b929b-22d6-4e02-a636-7fd8a13bde7f/04_%EB%A1%9C%EC%A7%80%EC%8A%A4%ED%8B%B1_%ED%9A%8C%EA%B7%80-Classification.pdf



Q 구문 오류(Syntax Error)가 뭔가요?

A 프로그래밍을 처음 하는 사용자들이 많이 만나게 되는 오류인데요. 이는 작성한 코드에 뭔가 문제가 있어서 아예 실행 조차 되지 않는다는 의미입니다. 대표적인 오류가 괄호를 열고, 닫지 않을 때입니다. 그러므로 Syntax Error를 만나면 작성한 코드에 잘못 입력한 것은 없는지 살펴보기 바랍니다. 오류와 관련된 내용은 6장에서 자세하게 살펴보겠습니다.

```
>>> print("""\\n  
동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세\\n  
""")  
동해물과 백두산이 마르고 닳도록  
하느님이 보우하사 우리나라 만세  
무궁화 삼천리 화려강산 대한사람  
대한으로 길이 보전하세
```

note \ 기호는 여러 줄 문자열을 사용할 때뿐만 아니라 다양한 상황에 활용되는 기호이므로 기억해 두기 바랍니다.

concatenate

미국식[kənkatəneɪt] ◻) 영국식[kɔn-] ◻)

(타동사)

1 사슬같이 있다; 연쇄시키다; <사건 등을> 결부[연결]시키다, 연관시키다

(형용사)

1 연쇄된, 이어진, 연결된

[영어사전](#) [격과](#) [더보기](#)

지금까지 문자열에서 원하는 위치를 지정해 문자를 분리해 보았습니다. 이처럼 [] 연산자를 이용해 문자열의 특정 위치에 있는 문자를 참조하는 것을 인덱싱indexing이라 하고, [:] 연산자를 이용해 문자열의 일부를 추출하는 것을 슬라이싱slicing이라 합니다.

IndexError 예외는 리스트/문자열의 수를 넘는 요소/글자를 선택할 때 발생합니다.

▣ 오류 → 파일 IDLE 에디터에서 실행했을 때 나타나는 내용으로 에디터마다 다르게 나타납니다.

```
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print("안녕하세요"[10])
IndexError: string index out of range → IndexError 예외가 발생했어요.
```

코드를 작성하다가 이런 예외가 발생하면 ‘리스트/문자열의 수를 넘는 부분을 선택했구나’라고 바로 인지할 수 있게 기억해 주세요. 중요한 예외이므로 4장에서 리스트를 배울 때 한 번 더 짚고 넘어가겠습니다.

out of range (of something) (out of range)

(~이) 달는[보이는/들리는] 거리[범위] 밖의

부동 소수점

[Floating Point  , 浮動小數點]

浮 뜰 부

법으로, 소수점의 위치

이며 지수는 소수점의

이 곱셈 형태로 표현하

호비트(1비트). 지수부

부수  총획 10획

1. (물에)뜨다

2. 떠다니다

3. 떠서 움직이다

떠서 움직이는 소수점.

와 floating point(실수 또는 부동 소수점)입니다. ‘부동 소수점’이라는 단어가 생소할 텐데, 실수는 52.273을 0.52273×10^2 와 같이 소수점의 위치를 바꿔도 결국 같은 숫자이므로, ‘소수점이 움직이는 숫자’라는 의미로 **부동 소수점**이라고 표현하기도 합니다. 이 단어를 쉽게 기억하기 위해서 ‘소수점이 동동 부유하며 움직인다’ 정도로 이해하겠습니다.

+ 여기서 잠깐

파이썬에서의 지수 표현

파이썬으로 만들어진 수학 연산, 인공지능 알고리즘을 보면 $0.52273e2$ 혹은 $0.52273E2$ 등의 특이한 숫자 표현을 볼 수 있습니다. 이는 파이썬에서 부동 소수점을 지수승으로 표현하는 방법입니다. 파이썬에서는 0.52273×10^2 를 $0.52273e2$ 혹은 $0.52273E2$ 로 표현합니다. 예를 들어 다음과 같습니다.

```
>>> 0.52273e2  
52.273  
>>> 0.52273e-2  
0.0052273
```

이 책에서는 크게 활용하지 않는 내용이지만, 이후에 필요할 수도 있는 내용이므로 “이러한 숫자 표현도 있구나”라고 기억하면 좋습니다.

회귀분석은 머신러닝의 시작이다. 딥러닝만 회귀분석 아니다.

정수 나누기 연산자: //

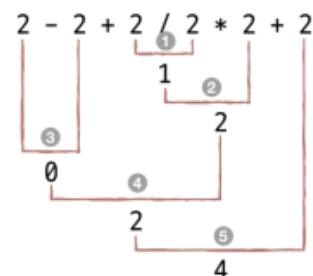
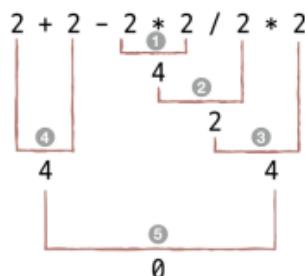
너무 쉬웠다고요? 그렇다면 생소한 연산자 하나만 알고 갈까요? 파이썬에는 // 형태의 연산자가 있습니다. 연산자 기호는 생소하지만, 이 또한 쉬운 개념의 연산자입니다. 이것은 숫자를 나누고 소수점 이하의 자릿수를 빼어 버린 후 정수 부분만 남기는 **정수 나누기 연산자**입니다. 간단하게 실행 결과를 살펴보겠습니다.

```
>>> print("3 / 2 =", 3 / 2)
3 / 2 = 1.5
>>> print("3 // 2 =", 3 // 2)
3 // 2 = 1
```

3/2는 1.5를 계산하지만 3//2는 1.5에서 소수점을 뗀 1만 결과로 나타내고 있습니다. // 연산자를 사용한 수식의 결과는 소수점 아래를 빼어 버린 값이 출력되는 것을 확인할 수 있습니다.

파이썬도 마찬가지입니다. 곱셈과 나눗셈이 덧셈과 뺄셈보다 우선합니다. 또한 곱셈/나눗셈과 덧셈/뺄셈처럼 같은 우선순위를 가지는 연산자는 왼쪽에서 오른쪽 순서로 계산합니다. 직접 코드를 입력하면서 살펴보겠습니다. 코드를 입력하고 실행하기 전에 실행 결과를 예측해 보세요.

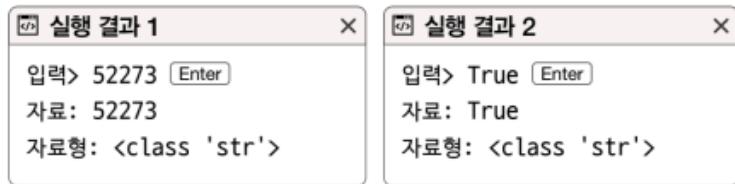
```
>>> print(2 + 2 - 2 * 2 / 2 * 2)
0.0
>>> print(2 - 2 + 2 / 2 * 2 + 2)
4.0
```



직접 해보는 손코딩

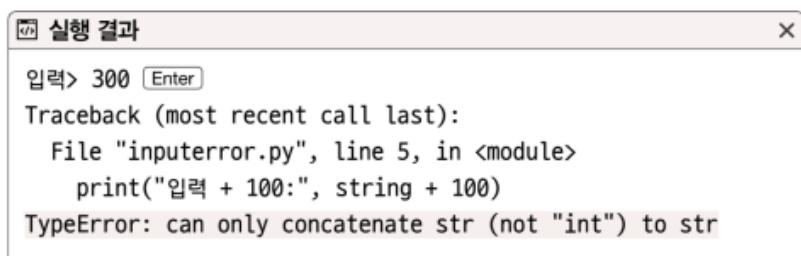
입력 자료형 확인하기 [소스 코드 input.py](#)

```
01 # 입력을 받습니다.  
02 string = input("입력> ")  
03  
04 # 출력합니다.  
05 print("자료:", string)  
06 print("자료형:", type(string))
```



입력받고 더하기 [소스 코드 input_error.py](#)

```
01 # 입력을 받습니다.  
02 string = input("입력> ")  
03  
04 # 출력합니다.  
05 print("입력 + 100:", string + 100)
```



입력받은 값 300과 100을 더하고자 한 것이었으나 input() 함수로 입력받은 자료는 모두 문자열로 저장되므로 "300"+100이 되어 문자열과 숫자는 더할 수 없어 발생한 오류입니다. 입력받은 문자열을 숫자로 변환해야 숫자 연산에 활용할 수 있습니다.

feature가 1개인 것이 데이터가 1개인 것이다. 예) [1,2,3,4]

(TABLE을 기준으로) 다중은 feature가 여러 개인 것.

convolutional

[형용사](#)

나선형의.

[영어사전 결과 더보기](#)

신경망(CNN) → 딥러닝(파이썬에서 tensorflow 사용)

layer(=층) = 퍼셉트론

input - 퍼셉트론 - output

deep - layer층을 깊게 가져가겠다는 뜻.

오래전에는 단층으로만 생각하고 있어서 딥러닝의 발전이 없었다.

int() 함수 활용하기 [소스 코드 int_convert.py](#)

```
01 string_a = input("입력A> ")
02 int_a = int(string_a)
03
04 string_b = input("입력B> ")
05 int_b = int(string_b)
06
07 print("문자열 자료:", string_a + string_b)
08 print("숫자 자료:", int_a + int_b)
```



int() 함수와 float() 함수 조합하기 | [소스 코드 int_float02.py](#)

```
01 input_a = float(input("첫 번째 숫자> "))
02 input_b = float(input("두 번째 숫자> "))
03
04 print("덧셈 결과:", input_a + input_b)
05 print("뺄셈 결과:", input_a - input_b)
06 print("곱셈 결과:", input_a * input_b)
07 print("나눗셈 결과:", input_a / input_b)
```

```
실행 결과
첫 번째 숫자> 273 [Enter]
두 번째 숫자> 52 [Enter]
덧셈 결과: 325.0
뺄셈 결과: 221.0
곱셈 결과: 14196.0
나눗셈 결과: 5.25
```

첫째, 숫자가 아닌 것을 숫자로 변환하려고 할 때

```
int("안녕하세요")
float("안녕하세요")
```

오류

```
Traceback (most recent call last):
  File "int_convert.py", line 2, in <module>
    int_a = int(string_a)
ValueError: invalid literal for int() with base 10: '안녕하세요'
```

```

1: # 텐서플로우 설치(command 창에서)
# conda install tensorflow

# 텐서플로우 설치 확인
import tensorflow as tf # 사이킷런은 딥러닝 지원 안 함.
import numpy as np

print(tf.__version__) # 2.9.1버전부터는 keras로직에 접근 가능해졌음.

```

2.9.1

```

1: # 가설 설정은  $H(x) = w(\text{기울기})x + b(\text{절편})$ 
x_train = [1,2,3,4] # 하나의 특성은 단순회귀분석이다.
y_train = [0,-1,-2,-3] #  $w$ 는 -1,  $b$ 는 1

1: tf.model = tf.keras.Sequential() # 모델 생성하기 위해서 Sequential 자료형을 통해서 모형을 만들 객체 생성.
# 회귀분석을 할 것인지 어떤 알고리즘을 할 것인지 tf.model에 변수로 저장해놓고 보려하는 것임. only 툴을 만들어줌.

1: # input_dim : input shape, units : output shape
tf.model.add(tf.keras.layers.Dense(input_dim=1, units=1)) # 구미는 작업을 add # 1차원 단층 회귀
# 마지막 선형자 학습처럼 결의 돼있음. # input_dim은 입력 차원으로 알려주면 됨. # units은 출력 차원

1: # SGD(Stochastic Gradient Descent) - 속도적 경사하강법, lr(learning_rate) - 이동의 간격을 지정해줌.
# 미분해서 0이 되는 값을 찾겠다는 것.
sgd = tf.keras.optimizers.SGD(learning_rate=0.1) # learning_rate * 기울기
# 기본적으로는 lr=0.1의 값으로 시작. 이 간격으로 옮겨가겠다는 것.
# 변수로 넣는 것은 또 다른 알고리즘이 있기 때문이다. 하나밖에 없었으면 변수로 안 만들고 작업했을 것이다.

1: # 목표는 minimize cost이다.
# mse(mean_squared_error) - 잔차를 제곱해서 평균을 내라,  $1/m * \text{sig}(y' - y)^2$ 
# 핵심의 기울기를 미분하면서 최적으로 학습하면서 핵심의 기울기가 0으로 되는 최적의 w값을 찾아라

tf.model.compile(loss='mse', optimizer='sgd')
# mean square error = cost function을 적용하라. # optimizer='sgd' 알고리즘을 이용해서 찾아라.

1: # 학습 전, 최초 설정된 w 값 조회
weights = tf.model.layers[0].get_weights()
w = weights[0][0][0] # 면 행 열로 접근해서 w값을 꺼낼 수 있다.
print('initial w is : ' + str(w)) # 랜덤으로 w값을 잡음.

initial w is : -1.1021825

```

deprecated 웬수집

중요도가 떨어져 더 이상 사용되지 않고 앞으로는 사라지게 될 (컴퓨터 시스템 기능 등)

The `lr` argument is deprecated, use `learning_rate` instead.

```

# SGD(Stochastic Gradient Descent) - 속도적 경사하강법, lr(learning_rate) - 이동의 간격을 지정해줌.
# 미분해서 0이 되는 값을 찾겠다는 것.
sgd = tf.keras.optimizers.SGD(lr=0.1) # learning_rate * 기울기
# 기본적으로는 lr=0.1의 값으로 시작. 이 간격으로 옮겨가겠다는 것.
# 변수로 넣는 것은 또 다른 알고리즘이 있기 때문이다. 하나밖에 없었으면 변수로 안 만들고 작업했을 것이다.

C:\ProgramData\Anaconda3\envs\tf_cpu\lib\site-packages\keras\optimizers\optimizer_v2\gradient_descent.py:108: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
    super(SGD, self).__init__(name, **kwargs)

```

```
tf.model.summary() # 신경망을 어떻게 구성해주고 있는지 알려줄.
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1)	2
<hr/>		
Total params: 2		
Trainable params: 2		
Non-trainable params: 0		
<hr/>		

Output Shape - None 행의 갯수는 제한을 두지 않겠다는 것.

```
tf.model.fit(x_train, y_train, epochs=200)  
# 대표적인 지도학습 # epochs=200 번 반복하면서 학습하라고 명령 # 너무 적으면 과소적합
```

```
Epoch 152/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0029  
Epoch 193/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0029  
Epoch 194/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0029  
Epoch 195/200  
1/1 [=====] - 0s 5ms/step - loss: 0.0028  
Epoch 196/200  
1/1 [=====] - 0s 3ms/step - loss: 0.0028  
Epoch 197/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0028  
Epoch 198/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0028  
Epoch 199/200  
1/1 [=====] - 0s 4ms/step - loss: 0.0028  
Epoch 200/200  
1/1 [=====] - 0s 3ms/step - loss: 0.0028
```

```
<keras.callbacks.History at 0x1f9db4c0ec8>
```

```
: y_predict = tf.model.predict(np.array([5])) # 넘파이 이용해서 5를 예측해보기  
y_predict
```

```
1/1 [=====] - 0s 74ms/step
```

```
: array([-3.9101572], dtype=float32)
```

무한대의 값을 기계에 설정될 수가 없다. 그래서 오차가 생길 수밖에 없다.

```
array([[-3.9101572]], dtype=float32)
```

02_minimizing_loss_show_graph

```
1]: import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

4]: # 가설 설정은  $H(x) = w(\text{기울기})x + b(\절편)$ 
x_train = [1,2,3,4] # 5번의 특성을 단순회귀분석이다.
y_train = [0,-1,-2,-3] # w는 -1, b는 1

tf.model = tf.keras.Sequential()
tf.model.addtf.keras.layers.Dense(input_dim=1, units=1))

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

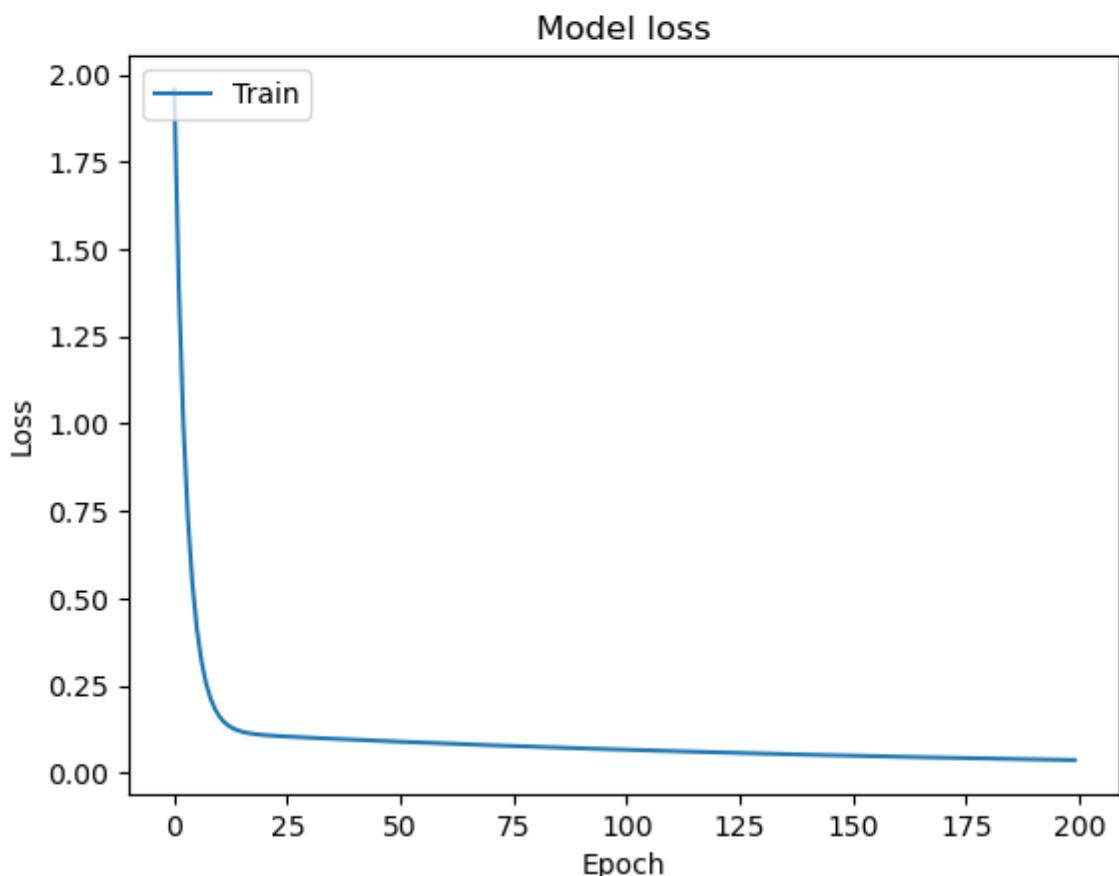
6]: history = tf.model.fit(x_train, y_train, epochs=200)

y_predict = tf.model.predict(np.array([5])) # 넘파이 이용해서 5를 예측해봄 # 결과는 -4에 가까운
y_predict
Epoch 193/200
1/1 [=====] - 0s 4ms/step - loss: 0.0570
Epoch 194/200
1/1 [=====] - 0s 4ms/step - loss: 0.0567
Epoch 195/200
1/1 [=====] - 0s 5ms/step - loss: 0.0563
Epoch 196/200
1/1 [=====] - 0s 3ms/step - loss: 0.0560
Epoch 197/200
1/1 [=====] - 0s 3ms/step - loss: 0.0557
Epoch 198/200
1/1 [=====] - 0s 4ms/step - loss: 0.0553
Epoch 199/200
1/1 [=====] - 0s 5ms/step - loss: 0.0550
Epoch 200/200
1/1 [=====] - 0s 3ms/step - loss: 0.0547
1/1 [=====] - 0s 15ms/step

6]: array([-3.6003642], dtype=float32)

]: # loss 값의 시각화
```

```
# loss 값의 시각화
plt.plot(loss.history['loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



Gradient descent algorithm

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^m (W x^{(i)} - y^{(i)}) x^{(i)}$$

맨 오른쪽 x는 입력값

다중 선형 회귀 분석

Cost function

$$H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^m (H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

다중 선형 회귀 분석

matrix를 사용한 Hypothesis

x ₁	x ₂	x ₃	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

$$w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

$$H(X) = XW$$

버려집니다. 그래서 아무 문제 없이 실행됩니다. 두 번째는 {}가 매개변수보다 많은 경우로 Index Error라는 예외가 발생합니다.

```
>>> "{} {}".format(1, 2, 3, 4, 5)
'1 2'
>>> "{} {} {}".format(1, 2)
Traceback (most recent call last):
File "<pyshell#1>", line 1, in <module>
    "{} {} {}".format(1, 2)
IndexError: tuple index out of range
```

다중 선형 회귀 분석

matrix를 사용한 Hypothesis

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[5, 3]

[3, 1]

[5, 1]

$$H(X) = XW$$

의미 없는 소수점 제거하기

파이썬은 0과 0.0을 출력했을 때 내부적으로 자료형이 다르므로 서로 다른 값으로 출력합니다. 그런데 의미 없는 0을 제거한 후 출력하고 싶을 때가 있습니다. 이때는 { :g}를 사용합니다.

직접 해보는 손코딩

의미 없는 소수점 제거하기 [소스 코드 format07.py](#)

```
01 output_a = 52.0
02 output_b = "{:g}".format(output_a)
03 print(output_a)
04 print(output_b)
```

실행 결과
52.0
52

03_multi_variable_linear_regression

```
: import tensorflow as tf
: import numpy as np

: x_data =
:   [ [73., 80., 75.],
:     [93., 88., 93.],
:     [89., 91., 90.],
:     [96., 98., 100.],
:     [73., 66., 70.] ]
:
: y_data =
:   [ [152],
:     [185],
:     [180],
:     [196],
:     [142] ]
```

```
: tf.model = tf.keras.Sequential()
: tf.model.add(tf.keras.layers.Dense(input_dim=3, units=1)) # input_dim=3는 feature 수를 얘기하기 때문에 차원으로 이해하지 말자.
```

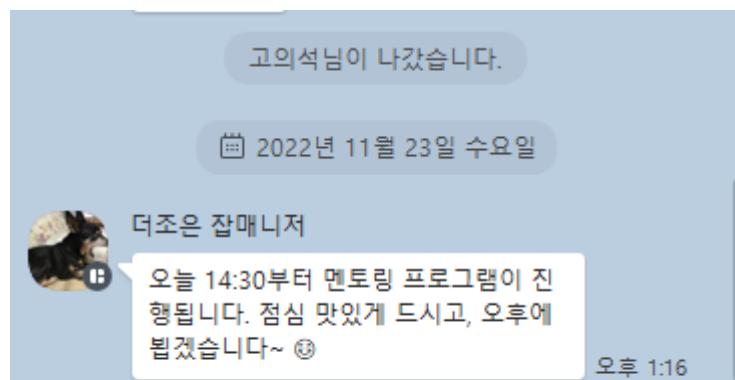
```
: tf.model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(learning_rate=1e-5))
: tf.model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_2 (Dense)	(None, 1)	4

=====
Total params: 4
Trainable params: 4
Non-trainable params: 0

```
: loss = tf.model.fit(x_data, y_data, epochs=100) # 'infinity'
1/1 [=====] - 0s 5ms/step - loss: 3.1150
Epoch 69/100
1/1 [=====] - 0s 4ms/step - loss: 3.1140
Epoch 70/100
1/1 [=====] - 0s 4ms/step - loss: 3.1129
Epoch 71/100
1/1 [=====] - 0s 5ms/step - loss: 3.1118
Epoch 72/100
1/1 [=====] - 0s 4ms/step - loss: 3.1107
Epoch 73/100
1/1 [=====] - 0s 3ms/step - loss: 3.1097
Epoch 74/100
1/1 [=====] - 0s 4ms/step - loss: 3.1086
Epoch 75/100
1/1 [=====] - 0s 5ms/step - loss: 3.1076
Epoch 76/100
1/1 [=====] - 0s 3ms/step - loss: 3.1065
Epoch 77/100
1/1 [=====] - 0s 4ms/step - loss: 3.1055
Epoch 78/100
```



현업 실무자 멘토링 - (주) 메디칼 스탠다드

PACS - 의료영상, 저장, 전송, 시스템 관련 업무.

SaMD - software as Medical Demand (기기)

본사 - 문정동(서울)

빅쿼리 big query

빅쿼리 (BigQuery)는 페타바이트급 이상의 데이터에 대해 스케일링 분석(필요시 실시간으로 컴퓨팅 자원을 동적으로 확장)을 가능하게 하는 완전 관리형(fully-managed) 서비스 컴퓨팅 데이터 웨어하우스이다. 빅쿼리는 2010년 5월 발표되었으며 2011년 11월 일반에 공개되었다. 설계 빅쿼리는 구글의 Dremel 기술에 대한 외부 접...
으키면 그

[GCP]GCP 기초_BigQuery란?

1. Big Query란? 확장성이 뛰어난 구글의 기업용 서버리스 기반의 데이터 웨어하우스, 표준 SQL을 지원하기 때문에 기존 SQL을 알고 있는 사용자도 손쉽게 사용 가능, 매월 무료로 최대 1TB 상당의 데이터

☞ <https://spacek82.tistory.com/68>



시사상식사전

피보팅

요약 트렌드나 바이러스 등 급속도로 변하는 외부 환경에 따라 기존 사업 아이템을 바탕으로 사업의 방향을 다른 쪽으로 전환하는 것을 일컫는다.

외국어 표기

pivoting(영어)

기존 사업 아이템이나 모델을 바탕으로 사업의 방향을 다른 쪽으로 전환하는 것을 말한다. 피봇(Pivot)은 본래 체육 용어로 몸의 중심축을 한쪽 발에서 다른 쪽 발로 이동시키는 것을 가리킨다. 산업상에서 피보팅은 완전히 새로운 사업을 시작하거나 모델을 개발하는 것이 아니라 기존의 비전은 유지하면서 전략만을 수정하여 사업 방향을 전환하는 것을 뜻한다. 대리점 중심으로 여행 상품을 판매하던 여행사가 온라인 판매로 사업을 전환하는 것을 예로 들 수 있다.

피보팅은 시장 상황이 예상과 다르거나 성과가 예상보다 저조할 때 또는 개발 일정이 지연될 때 주로 행해진다. 피보팅을 할 경우 기존의 사업 방향을 포기하고 새롭게 도전해야 하므로 위험도가 크지만 큰 성과를 얻을 수도 있다.

최근에는 트렌드에 민감한 사회가 도래하면서 피보팅을 얼마나 빠르게 하는지가 기업의 생존을 결정지을 수 있다는 주장도 나온다. 특히 김난도 서울대 소비자학과 교수는 소띠 해인 2021년 신축년(辛丑年) 한국 사회의 소비 흐름을 전망한 키워드 카우보이 히어로(COWBOY HERO)에서 「Best we pivot(거침없이 피보팅)」를 거론한 바 있다.

AWS???

의사는 유방암 검사에 70% 확률이 나오지만, AI는 83%가 넘어감.

다시 수업

input → layer: $x \cdot w$ 활성화(activation) 함수를 거쳐서 → output 출력

type의 오류

```
'builtin_function_or_method' object is unsubscriptable
```

다음과 같은 에러가 났다. TypeError: 'builtin_function_or_method' object is unsubscriptable 환경 : python 2.6.6 on Windows7. 전체 traceback 은 다음과 같다. Traceback (most recent call last): File

 <https://adnoctum.tistory.com/461>



```
[8]: x_predict = tf.model.predict(np.array([[72., 93., 90.]]))  
x_predict  
1/1 [=====] - 0s 69ms/step  
[8]: array([[165.5858]], dtype=float32)
```

04_file_input_linear_regression

```
] : import tensorflow as tf  
      import numpy as np  
  
[] : xy = np.loadtxt('data/data-01-test-score.csv', delimiter=',', dtype=np.float32)  
    # csv의 csv의 delimiter=','는 구분자를 말함 # float 4byte = 32bit  
    x_data = xy[:, 0:-1] # 슬라이싱이기 때문에 생략 가능 # ,로 접근하려면 반드시 []로 감싸줘야 한다.  
    y_data = xy[:, [-1]]
```

```
: print(x_data, '#nx_data.shape:', x_data.shape)
print(y_data, '#ny_data.shape:', y_data.shape)
```

```
[[ 73.  80.  75.]
 [ 93.  88.  93.]
 [ 89.  91.  90.]
 [ 96.  98. 100.]
 [ 73.  66.  70.]
 [ 53.  46.  55.]
 [ 69.  74.  77.]
 [ 47.  56.  60.]
 [ 87.  79.  90.]
 [ 79.  70.  88.]
 [ 69.  70.  73.]
 [ 70.  65.  74.]
 [ 93.  95.  91.]
 [ 79.  80.  73.]
 [ 70.  73.  78.]
 [ 93.  89.  96.]
 [ 78.  75.  68.]
 [ 81.  90.  93.]
 [ 88.  92.  86.]
 [ 70.  88.  77.]
```

```
[7]: print(x_data, '#nx_data.shape:', x_data.shape)
print(y_data, '#ny_data.shape:', y_data.shape)
```

```
[ 96.  93.  95.])
x_data.shape: (25, 3)
[[152.]
 [185.]
 [180.]
 [196.]
 [142.]
 [101.]
 [149.]
 [115.]
 [175.]
 [164.]
 [141.]
 [141.]
 [184.]
 [152.]
 [148.]
 [192.]
 [147.]
 [183.]]
```

```
|: print(x_data, '\nx_data.shape:', x_data.shape)
|: print(y_data, '\ny_data.shape:', y_data.shape)
|: [115.]
|: [175.]
|: [164.]
|: [141.]
|: [141.]
|: [184.]
|: [152.]
|: [148.]
|: [192.]
|: [147.]
|: [183.]
|: [177.]
|: [159.]
|: [177.]
|: [175.]
|: [175.]
|: [149.]
|: [192.]]
y_data.shape: (25, 1)
```

learning_rate를 너무 크게 하면 발산해버린다. 넓게 움직이기 때문에.

```
|: tf.model = tf.keras.Sequential()
|: tf.model.add(tf.keras.layers.Dense(input_dim=3, units=1, activation='linear')) # activation 생략해놓고 변수 선언을 해서 헷갈리지 않도록.
|: tf.model.summary()
Model: "sequential"
+-----+
Layer (type)        Output Shape       Param #
+-----+
dense (Dense)      (None, 1)           4
+-----+
Total params: 4
Trainable params: 4
Non-trainable params: 0
```

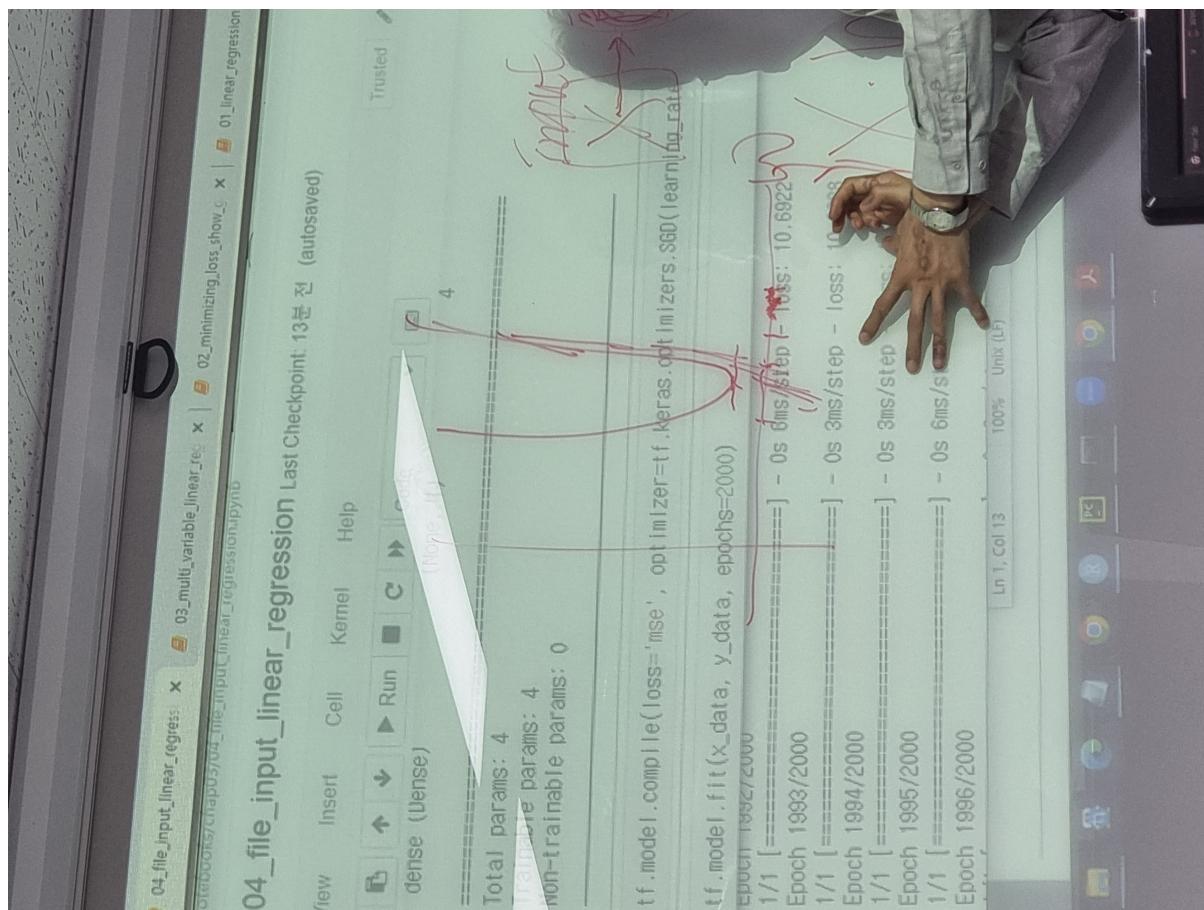
```

: tf.model.compile(loss='mse', optimizer=tf.keras.optimizers.SGD(learning_rate=1e-5))
# learning_rate가 너무 커서 loss값이 크게 나오는 것이다.

: tf.model.fit(x_data, y_data, epochs=2000)
Epoch 1992/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7497
Epoch 1993/2000
1/1 [=====] - 0s 4ms/step - loss: 6.7492
Epoch 1994/2000
1/1 [=====] - 0s 5ms/step - loss: 6.7487
Epoch 1995/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7482
Epoch 1996/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7477
Epoch 1997/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7472
Epoch 1998/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7467
Epoch 1999/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7462
Epoch 2000/2000
1/1 [=====] - 0s 3ms/step - loss: 6.7457

: <keras.callbacks.History at 0x20489c19ec8>

```



데이터가 너무 적으면 epochs값을 크게 해야한다.

```
In [12]: print(tf.model.predict([[100., 70., 98.]]))  
1/1 [=====] - 0s 74ms/step  
[[178.72472]]
```
