NeuralNet 101

3. Linear Regression

What is Linear Regression?

Ans. Estimating the mean in multi variable data with Linear function.

Then, what is the mean?

There have lots of way to calculate mean

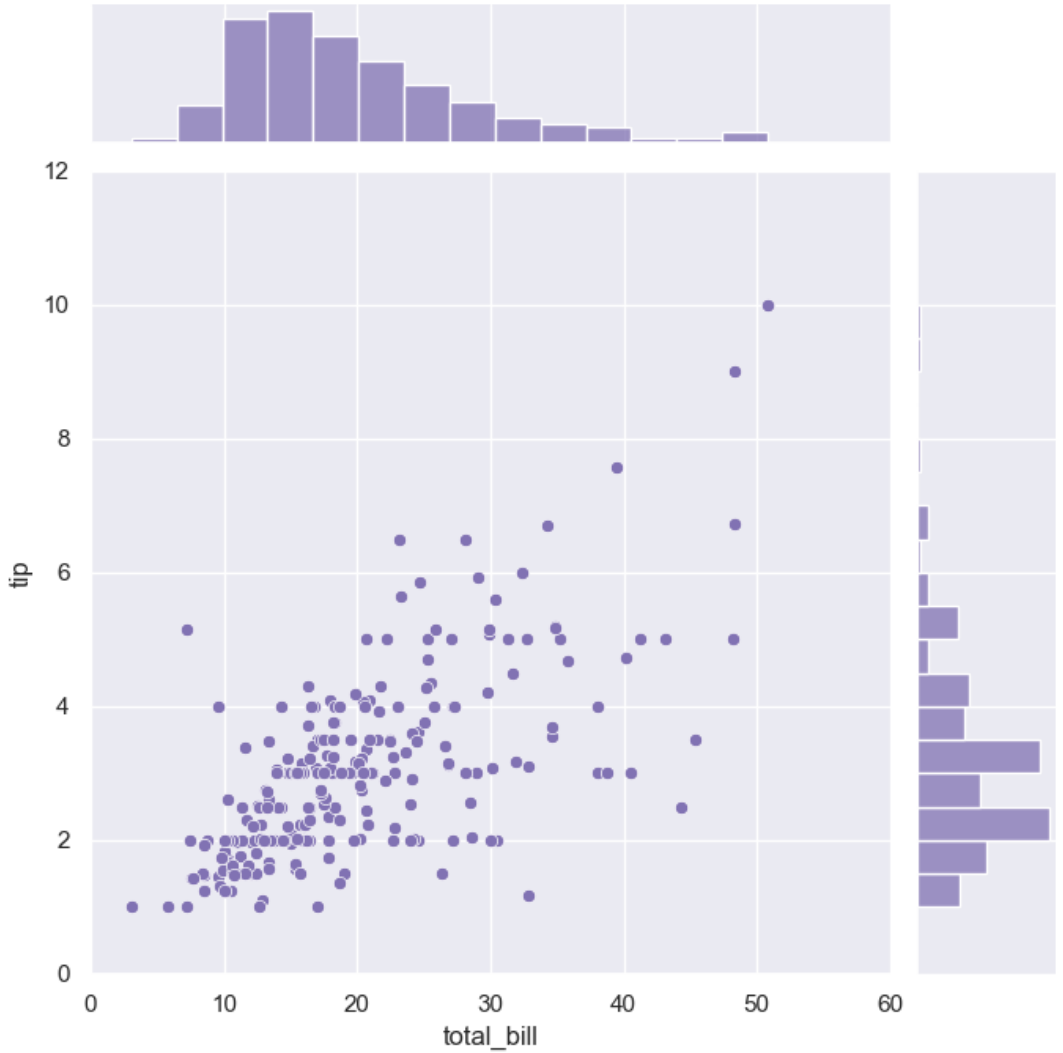
Arithmetical mean

$$\frac{\sum_{i=1}^{n} A_i}{n}$$

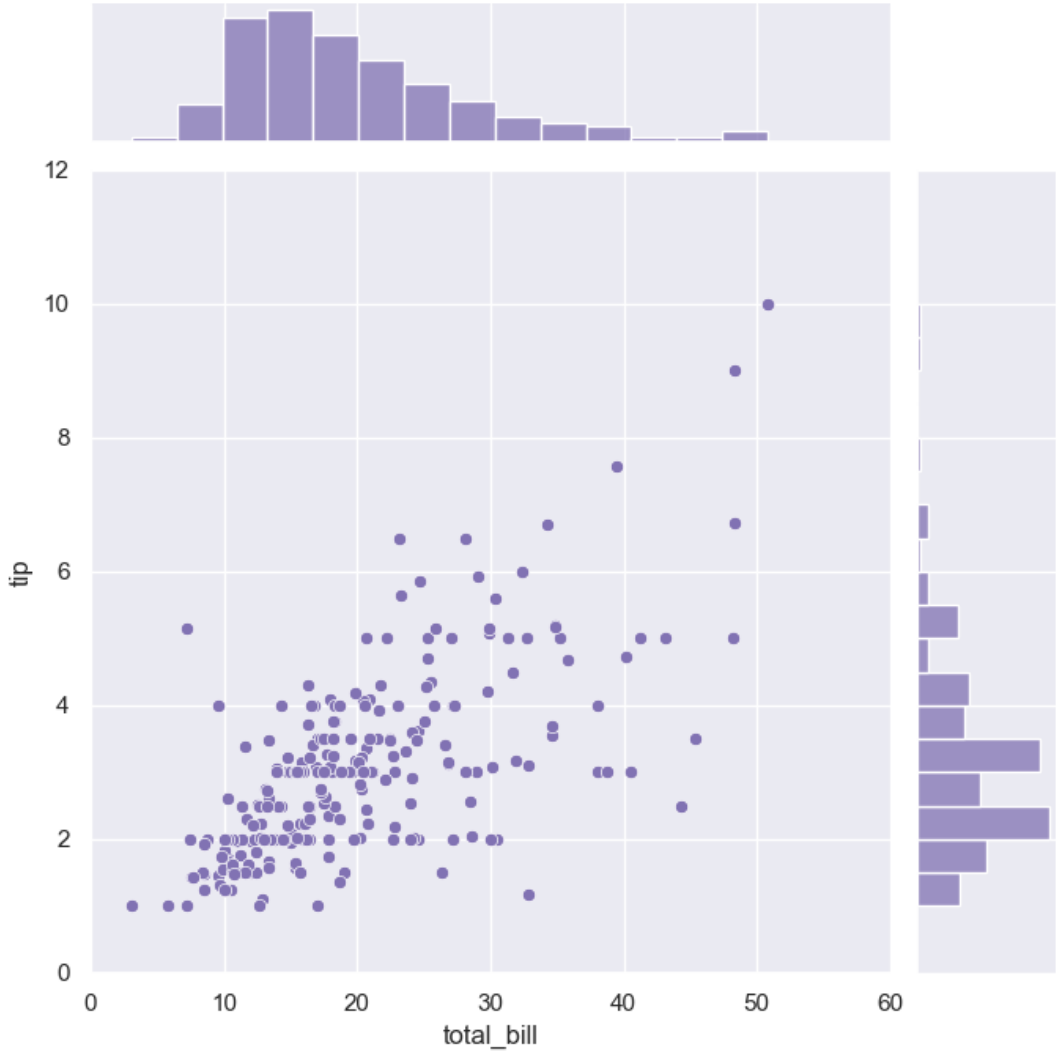
Geometric mean

$$\sqrt[n]{\prod_{i=1}^{n} A_i}$$

But how can we define the mean in this picture?



We can set it like on 0 -> x, on 1 -> 1 … but it is not efficient at all.



So I will make a linear function to get mean value in anywhere of domain.

Let’s define a linear function

$$f(x) = wx + b$$

And let’s think…

- Can we set the parameter (w and b) of linear function to estimate the mean value of data?

- Does Linear function can accurately give exact mean value?

First question : True

- Let’s think…
- We define the mean as average in x at certain point p.
- Then we can define as average not only just arithmetical mean, but with loss function L

Defining Loss function L

- Let distance D as (f(x_i) - y_i)^2 (I squared it because loss is diminished if the f(x_i) is smaller than y_i

- And make the sum of D and make average. (the number of D is m)

- Then, we can define the Loss function as $\frac{1}{2m}\sum D$ (I set not m but 2m because we need to use derivative in next slide)

Let me set Loss function based on f(x) and y…

- Then I can define the loss function as…

- $L(w, b) = \frac{1}{2m}\sum_{i=o}^{m}(f(x_i) - y_i)$^2

- We set loss function, and we have learned the optimization based on gradient. Therefore, we can find the optimal point(w,b) by getting gradient of function L

Some mathematical stuff…

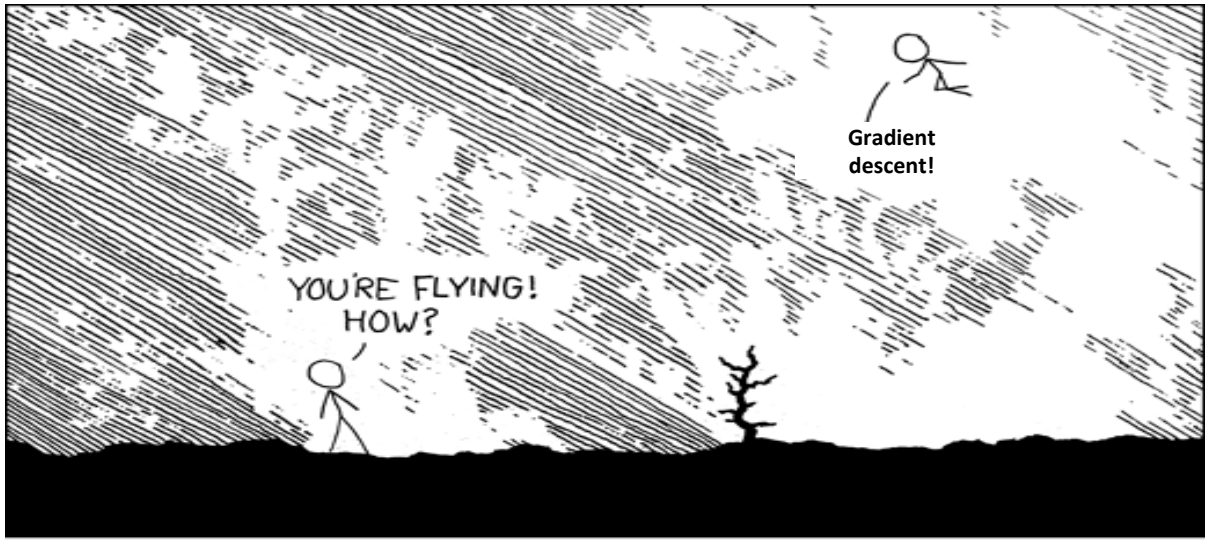
$$w := w - \alpha \frac{\delta}{\delta W} \frac{1}{2m} \sum_{i=1}^{m} 2(Wx - y)x$$

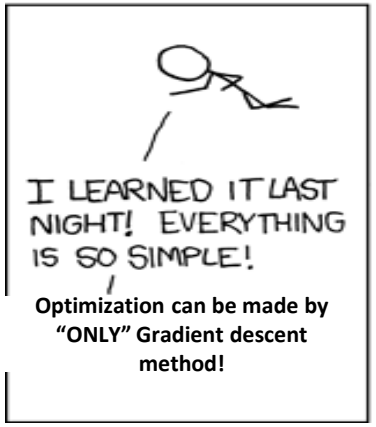
$$w := w - \alpha \frac{\delta}{\delta W} \frac{1}{m} \sum_{i=1}^{m} (Wx - y)x$$

And also we can apply it on multi-variable

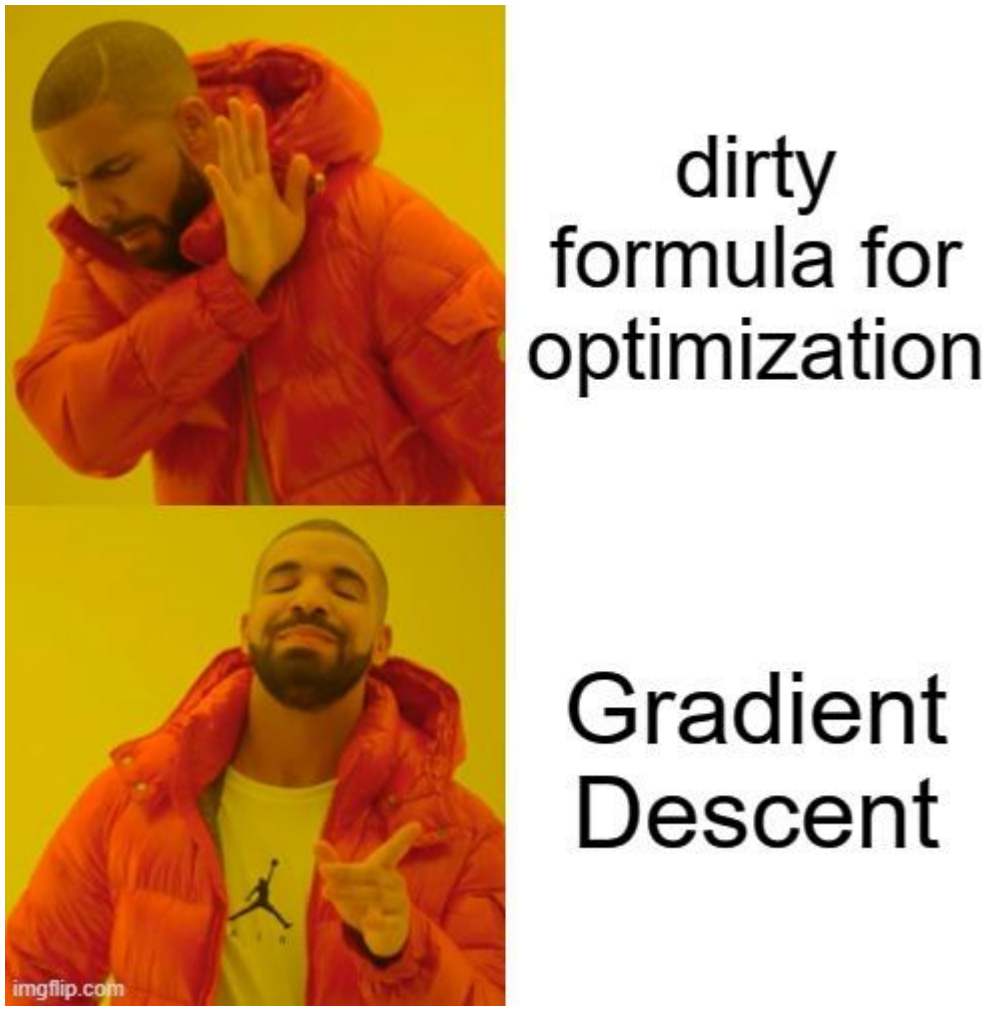
- Let's define f(x) as f($\vec{x}$).
- Then we can write the function f as: $\vec{x} \cdot \vec{w} + \vec{b}$
- And also expand L function to R^m

Just think that we can optimize it by gradient-descent method





Just think that we can optimize it by gradient-descent method



What can we do with Linear Regression?

- Predicting delivery time based on time and distance

- Predicting housing prices based on many features(crime rate, distance between subway station, rooms … etc)

References.

- 김성훈, 모두를 위한 딥러닝 Linear regression cost 함수 최소화, https://hunkim.github.io/ml/lec3.pdf

Lab session

In Lab session…

- Linear regression with mathematical operation
- Linear regression with Optimizer in PyTorch
- Pandas – how to read excel files in Python
- PyTorch – how to manipulate tensor

Linear regression with mathematical operation

$$H(x) = W \cdot x$$

(no bias to simplify the formula)

Example)

x	Y
1	3.3
2	7.3
3	10.3
4	14

Linear regression with mathematical operation

```
1    import torch
2    lr = 0.01
```

lr = learning rate

Linear regression with mathematical operation

x	Y
1	3.3
2	7.3
3	10.3
4	14

$$X = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

Size : 4×1

$$Y = \begin{pmatrix} 3.3 \\ 7.3 \\ 10.3 \\ 14 \end{pmatrix}$$

Size : 4×1

```
4    x_train = torch.tensor([[1], [2], [3], [4]], dtype=torch.float32)
5    y_train = torch.tensor([[3.3], [7.3], [10.3], [14]], dtype=torch.float32)
```

Linear regression with mathematical operation

```
W = torch.zeros(1, dtype=torch.float32)
```

$$W = (0)$$

Size : 1×1
(scalar)

Linear regression with mathematical operation

$$H(x) = W$$

Cost(Loss function) -> Gradient Descent

$$\frac{\partial L(W)}{\partial W} = \frac{1}{m}\sum_{i=o}^{m}(W \cdot x_i - y_i) \cdot x_i \qquad W -= \alpha \frac{\partial L(W)}{\partial W}$$

Linear regression with mathematical operation

```
 9    nb_epochs = 20
10    for epoch in range(nb_epochs):
11        hypothesis = x_train * W
12        cost = torch.mean((hypothesis - y_train) ** 2)
13        gradient = torch.sum((x_train * W - y_train) * x_train)
14        W -= lr * gradient
```

Hypothesis : $\mathrm{X} \cdot W = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \cdot (a) = \begin{pmatrix} a \\ 2a \\ 3a \\ 4a \end{pmatrix}$

Linear regression with mathematical operation

```
16    x_predict = torch.FloatTensor([[10], [17], [20], [23]])
17    y_predict = x_predict * W
18    print(y_predict)
```

Result :

```
tensor([[34.9055],
        [59.3393],
        [69.8109],
        [80.2826]])
```

Linear regression with mathematical operation – Full code

```
import torch
lr = 0.01

x_train = torch.tensor([[1], [2], [3], [4]], dtype=torch.float32)
y_train = torch.tensor([[3.3], [7.3], [10.3], [14]], dtype=torch.float32)

W = torch.zeros(1, dtype=torch.float32)

nb_epochs = 20
for epoch in range(nb_epochs):
    hypothesis = x_train * W
    cost = torch.mean((hypothesis - y_train) ** 2)
    gradient = torch.sum((x_train * W - y_train) * x_train)
    W -= lr * gradient

x_predict = torch.FloatTensor([[10], [17], [20], [23]])
y_predict = x_predict * W
print(y_predict)
```

Linear regression with Optimizer in PyTorch

With same example, but

$$H(x) = W \cdot x + b$$

```
6    W = torch.zeros(1, requires_grad=True, dtype=torch.float32)
7    b = torch.zeros(1, requires_grad=True, dtype=torch.float32)
```

Linear regression with Optimizer in PyTorch

```
9    optimizer = torch.optim.SGD([W, b], lr=0.01)
```

Stochastic Gradient Descent

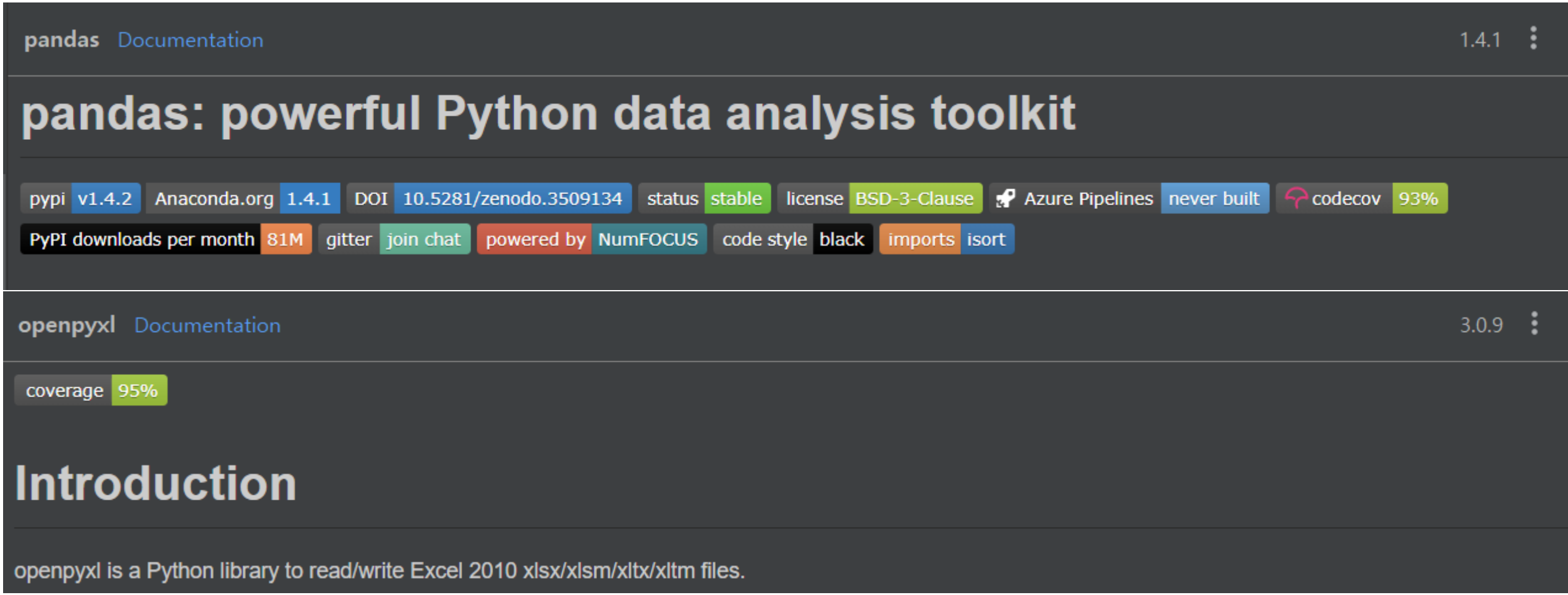
```
11   nb_epochs = 5000
12   for epoch in range(nb_epochs):
13       hypothesis = x_train * W + b
14       cost = torch.mean((hypothesis - y_train) ** 2)
15
16       optimizer.zero_grad() #initialize gradient
17       cost.backward() #calculate gradient
18       optimizer.step() #change W, b
```

Linear regression with Optimizer in PyTorch – Full code

```python
import torch

x_train = torch.tensor([[1], [2], [3], [4]], dtype=torch.float32)
y_train = torch.tensor([[3.3], [7.3], [10.3], [14]], dtype=torch.float32)

W = torch.zeros(1, requires_grad=True, dtype=torch.float32)
b = torch.zeros(1, requires_grad=True, dtype=torch.float32)

optimizer = torch.optim.SGD([W, b], lr=0.01)

nb_epochs = 5000
for epoch in range(nb_epochs):
    hypothesis = x_train * W + b
    cost = torch.mean((hypothesis - y_train) ** 2)

    optimizer.zero_grad() #initialize gradient
    cost.backward() #calculate gradient
    optimizer.step() #change W, b

x_predict = torch.FloatTensor([[10], [17], [20], [23]])
y_predict = x_predict * W + b
print(y_predict)
```

Pandas – how to read excel files in Python

Install ‘pandas’ and ‘openpyxl’



Pandas – how to read excel files in Python

```
1    import pandas as pd
```

Import ‘pandas’.

```
6    df = pd.read_excel(filename)
```

Use pd.read_excel(file address).
It gives you pandas dataframe.

Pandas – how to read excel files in Python

	A	B	C
1	wavelength	intensity	
2	350.4	0.016524	
3	351.1	0.018386	
4	351.7	0.016921	
5	352.3	0.016081	
6	353	0.017333	
7	353.6	0.01454	

Pandas – how to read excel files in Python

```
a = torch.tensor(df['wavelength'].values.tolist(), dtype=torch.float32)
```

Results : print(a)

```
tensor([350.4000, 351.1000, 351.7000, 352.3000, 353.6000])
```

Pandas – how to read excel files in Python

```
a = torch.tensor(df['wavelength'].values.tolist(), dtype=torch.float32)
```

Results : print(a)

```
tensor([350.4000, 351.1000, 351.7000, 352.3000, 353.0000, 353.6000])
```

Pandas – how to read excel files in Python

	A	B	C
1	wavelength	intensity	
2	350.4	0.016524	
3	351.1	0.018386	
4	351.7	0.016921	
5	352.3	0.016081	
6	353	0.017333	
7	353.6	0.01454	

Pandas – how to read excel files in Python

Make list with multiple columns with df.loc[]

```
a = torch.tensor(df.loc[:, ['wavelength','intensity']].values.tolist(), dtype=torch.float32)
```

Results : print(a)

```
tensor([[3.5040e+02, 1.6524e-02],
        [3.5110e+02, 1.8386e-02],
        [3.5170e+02, 1.6921e-02],
        [3.5230e+02, 1.6081e-02],
        [3.5300e+02, 1.7333e-02],
        [3.5360e+02, 1.4540e-02]])
```

PyTorch – how to manipulate tensor

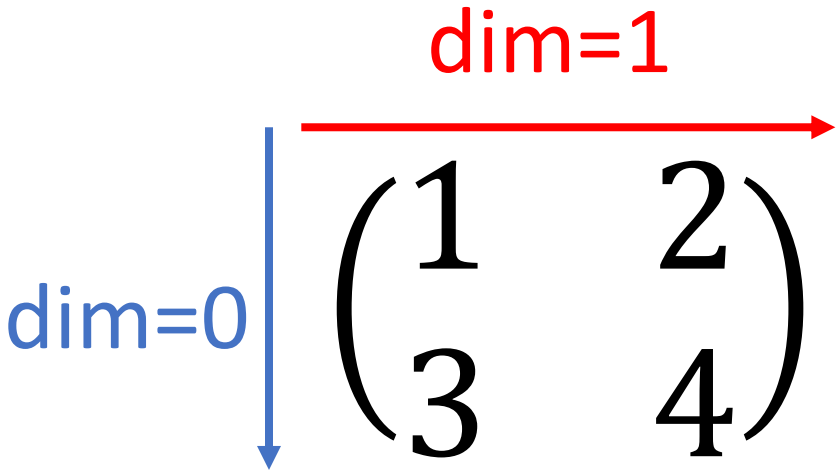
m1.matmul(m2) #matrix multiplication
m1.mul(m2) #multiplication

$$\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}(matmul)\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\cdot\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}7 & 10\\ 15 & 22\end{pmatrix}$$

$$\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}(mul)\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}=\begin{pmatrix}1 & 4\\ 9 & 16\end{pmatrix}$$

PyTorch – how to manipulate tensor

```
t.mean()        #mean(average)
t.sum()         #sum
t.max()         #find max value&index
```

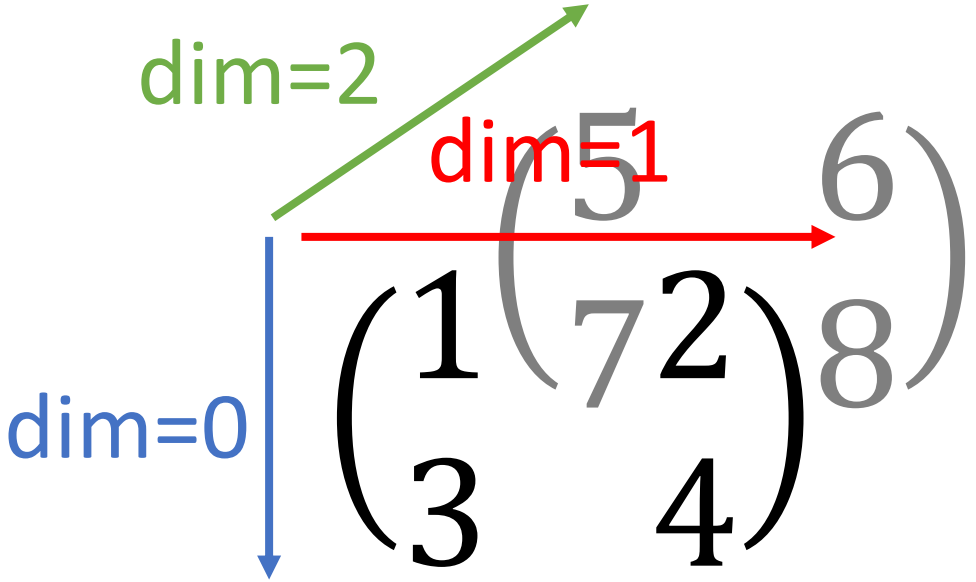


t.mean(dim=0)

$$\begin{pmatrix} 2 & 3 \end{pmatrix}$$

PyTorch – how to manipulate tensor

```
t = np.array([[[1,5],[2,6]],[[3,7],[4,8]]])
ft = torch.tensor(t, dtype=torch.float32)
ft.shape                    #[2,2,2]
```



$$\begin{pmatrix} 1 & 5 \\ 2 & 6 \end{pmatrix} \begin{pmatrix} 3 & 7 \\ 4 & 8 \end{pmatrix}$$

```
ft.view([-1,2])     #[4,2]
[[1,5],[2,6],[3,7],[4,8]]
```

$$\begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

PyTorch – how to manipulate tensor

ft.squeeze()
Shape : [n, m, l, 1] -> [n, m, l]

ft.unsqueeze(dim) #insert ‘1’ in dimension
Shape : [n, m, l] -> [1, n, m, l] #if dim is 0

PyTorch – how to manipulate tensor

torch.cat([x,y], dim=0)
torch.stack([x,y,z])

torch.zeros_like(x)
torch.ones_like(x)

In-place operators (x.mul(y) vs x.mul_(y))

Lab02 Problems

- Check github vlab-kaist
- https://github.com/vlab-kaist/NeuralNet101
- Problems>Lab02

- If you solved all of problems, please make issue.
- Title : ‘Section_Name_lab_week2’
  (ex. ‘B_장유진_lab_week2’)

References.

- 모두를 위한 딥러닝 시즌2 – PyTorch https://www.youtube.com/playlist?list=PLQ28Nx3M4JrhkqBVIXg-i5_CVVoS1UzAv, Lab01~04