파이썬 입문

한국폴리텍대학

2023.06.02

넘파이 (NumPy)

NumPy (Numerical Python)

- 수치해석, 과학계산을 위한 파이썬 모듈
- 행렬, 배열 연산

```
import numpy as np

a = np.array( [ 1, 2, 3 ] )
b = np.array( [ [ 1. , 2 ] , [ 3, 4 ] ] )
c = np.array( [ [ [ 1, 2 ] , [ 3, 4 ] ] , [ [ 5, 6 ] , [ 7, 8 ] ] ] )
d = np.array( [ [1], [2], [3] ] , float )   # int, float ……

print(type(a), a.shape, a.dtype)
print(type(b), b.shape, b.dtype)
print(type(c), c.shape, c.dtype)
print(type(d), d.shape, d.dtype)
```

```
a = [1, 2]              # list
a = {'1':1, '2':2}      # dict
a = {1, 2}              # set
a = (1, 2)              # tuple
a = np.array([1, 2])    # numpy.ndarray
```

1. list와 numpy.ndarray는 [ ]

2. numpy.ndarray는 모든 요소가 동일한 데이터 타입

넘파이 (NumPy)

```
import numpy as np

a = [[1, 2, 3], [4, 5, 6]]
# a = [[1, 2, '3'], [4, 5, 6]]
# a = [[1, 2, 3. ], [4, 5, 6]]
b = np.array(a)          # list를 numpy.ndarray로 변환
# b = np.array(a, int)
print(b)                 # array는 모든 요소가 동일한 데이터타입
print(b.shape)           # (2, 3)
print(b[0,0])            # 인덱싱

print(np.arange(10))           # 0, 1, 2, ..... , 8, 9
print(np.arange(5, 10))        # 5, 6, 7, 8, 9
print(np.arange(5, 15, 3))     # 5, 8, 11, 14
print(np.random.rand(2, 3)
print(np.empty((2, 3)))
print(np.zeros((2, 3)))        # [ [ 0.  0.  0. ]  [ 0.  0.  0. ] ]
print(np.ones((2, 3)))         # [ [ 1.  1.  1. ]  [ 1.  1.  1. ] ]
print(np.full((2, 3), 5))      # [ [ 5 5 5 ] [ 5 5 5 ] ]
print(np.eye(3))               # [ [1. 0 0] [0 1. 0] [0 0 1.] ]   단위행렬
```

```
a = np.arange(20)
b = a.reshape((4, 5))

    [ [ 0 1 2 3 4 ] [ 5 6 7 8 9 ] [ 10 11 12 13 14 ] [ 15 16 17 18 19 ] ]

data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
arr = np.array(data)   # list를 numpy.ndarray로 변환
a = arr[0:2, 0:2]          # 슬라이싱
print(a)    # [[1 2] [4 5]]
a = arr[1:, 1:]
print(a)    # [[5 6] [8 9]]

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

c = a + b
c = np.add(a, b)

c = a – b
c = np.subtract(a, b)

c = a * b
c = np.multiply(a, b)

c = a / b
c = np.divide(a, b)
```

```
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.sum(a)
c = a.sum()
d = a.mean()
print(b)
print(c)
print(d)
```

파이토치 (PyTorch)

PyTorch는 Python을 위한 오픈소스 머신러닝 라이브러리

torch 패키지 기본 구성

1. torch
   : 텐서 등의 다양한 수학 함수가 포함

2. torch.autograd
   : 자동미분

3. torch.nn # neural network
   : 신경망을 구축하기 위한 다양한 데이터 구조나 레이어 등이 정의
   레이어, 활성화함수, 손실함수 등

4. torch.optim
   : 최적화 알고리즘

5. torch.utils.data
   : 미니 배치용 유틸리티 함수. 데이터로드 등

6. torch.onnx # open neural network exchange
   : onnx 포맷으로 서로 다른 딥러닝 프레임워크간 모델 공유시 사용

사용 (예)

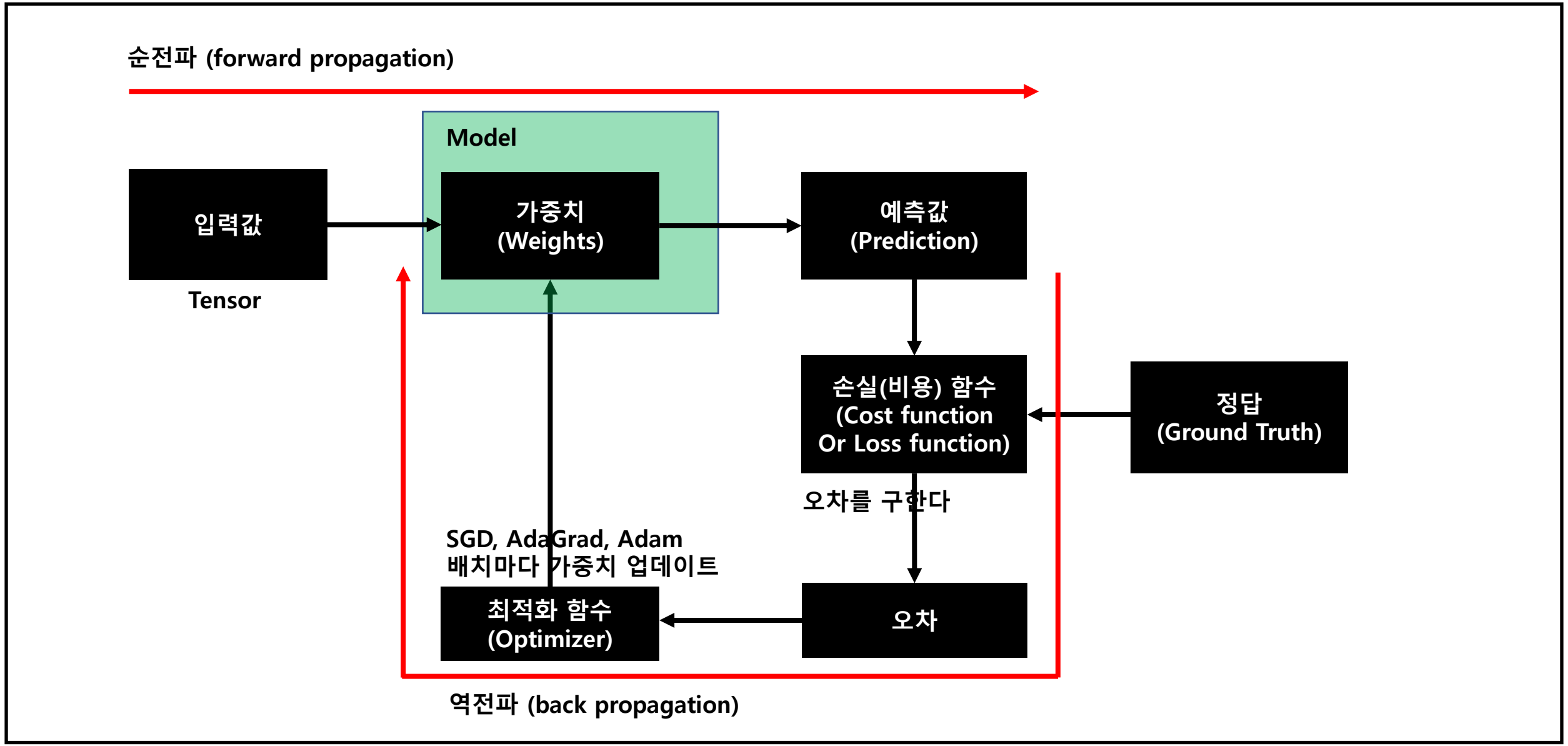
1. 예측 (Prediction)

   - 선형 회귀 (Linear Regression)

2. 분류 (Classification)

   - 합성곱 신경망 (Convolutional Neural Network, CNN)

PyTorch - Model



PyTorch - Tensor

Tensor

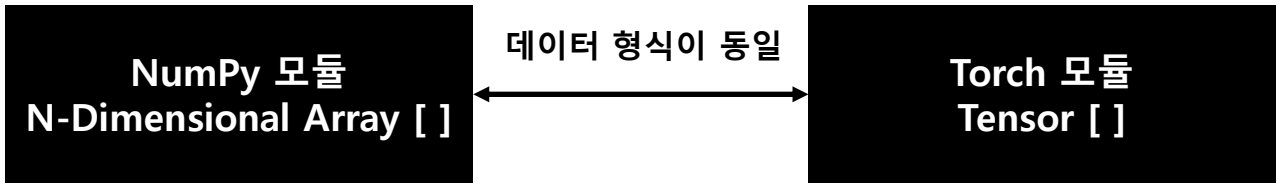
```
import torch

a = torch.Tensor([[1, 2, 3], [4, 5, 6]])  # float32 (class)
print(a, a.shape)
b = a.numpy()          # tensor를 numpy로 변환
a = torch.Tensor(a)    # numpy를 tensor로 변환
```

```
a = torch.IntTensor([[1, 2, 3], [4, 5, 6]])        # int32   (class)
a = torch.ShortTensor([[1, 2, 3], [4, 5, 6]])      # int16   (class)
a = torch.LongTensor([[1, 2, 3], [4, 5, 6]])       # int64   (class)
a = torch.FloatTensor([[1, 2, 3], [4, 5, 6]])      # float32 (class)
a = torch.DoubleTensor([[1, 2, 3], [4, 5, 6]])     # float64 (class)
a = torch.tensor([[1, 2, 3], [4, 5, 6]])           # int   (function)
```

```
a = torch.arange(10)
a = torch.arange(5, 10)
a = torch.arrange(5, 15, 3)
a = torch.rand(2, 3)
a = torch.empty(2, 3)
a = torch.zeros(2, 3)
a = torch.ones(2, 3)
a = torch.full(2, 3)
a = torch.eye(2, 3)
```

```
# NumPy
np.arange(10)
np.arange(5, 10)
np.arange(5, 15, 3)
np.random.rand(2, 3)
np.empty((2, 3))
np.zeros((2, 3))
np.ones((2, 3))
np.full((2, 3), 5)
np.eye(3)
```

데이터 형식



Tensor [ ] 는 GPU 사용이 가능한 데이터 형식

PyTorch - Tensor

Tensor

.squeeze() : tensor의 차원을 줄여줌

```
squeeze_tensor = torch.rand(size=(2,1,2,1,2))
print(squeeze_tensor.squeeze().shape)    # (2, 2, 2)
print(squeeze_tensor.squeeze(3).shape)  # (2, 1, 2, 2)
```

.unsqueeze() : tensor의 차원을 늘려줌

```
unsqueeze_tensor = torch.rand(size=(2,2))
print(unsqueeze_tensor.unsqueeze(0).shape) # [1,2,2]
print(unsqueeze_tensor.unsqueeze(1).shape) # [2,1,2]
print(unsqueeze_tensor.unsqueeze(2).shape) # [2,2,1]
```

속성(attribute)

.shape

.dtype

.device

.size()

```
a.size()
```

.ndimension()

```
a.ndimension()
```

.reshape() : tensor의 shape을 변경

```
b = a.reshape(3, 1, 1)
```

.flatten()

```
c = a.flatten()
```

.item() : 단일 요소 텐서인 경우, 요소를 숫자값으로 변환

_접미사가 붙는 함수 : 바꿔치기(in-place) 연산

.add_(), .abs_(), squeeze_() ……

PyTorch – Model

모델 : Linear, Conv1D, Conv2D, Conv3D, LSTM, RNN 등

활성화함수 : ReLU, LeakyReLU, SeLU, SiLU, tanh, Mish, Sigmoid, Softmax 등

기타 : Pooling, BatchNorm, DropOut, Flatten 등

PyTorch – Cost(Loss) function

비용(손실) 함수 Cost(Loss) Function

```
cost = torch.mean((prediction - y_train) ** 2)
print(cost.item())

cost2 = f.mse_loss(prediction, y_train)        # 예측값 – 정답
print(cost2.item())
```

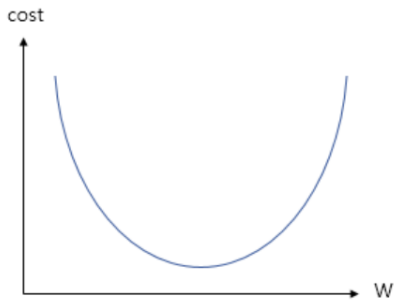
$$cost(W,b) = \frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)} - H(x^{(i)})\right]^2$$

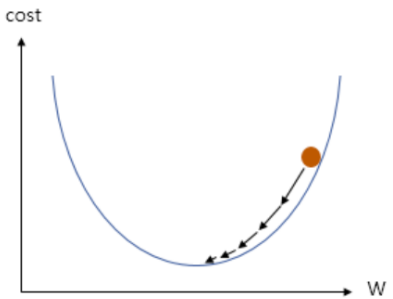
최적화 (Optimizer)

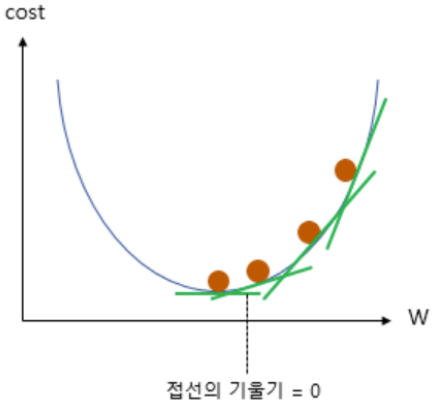
경사하강법 (GD, Gradient Descent), 확률적 경사하강법 (SGD, Stochastic Gradient Descent)

기울기 w가 커지면, cost가 커짐

cost가 최소가 되는 지점: 접선의 기울기가 0이 되는 지점 또는 미분값이 0이 되는 지점

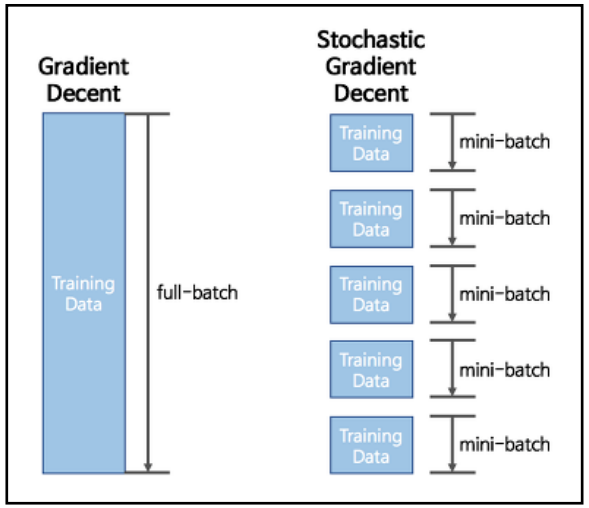






	1	2	3	4
실제	2	4	6	8
예측	0.7	1.4	2.1	2.8
오차	1.3	2.8	3.9	5.2

y = w * x , w 초기값 0.7
cost = 51.78 / 4 = 12.945
w(t+1) = w(t) – lr * (dcost(w)/dw)



PyTorch – (Simple) Linear Regression

y = 2x --> w : 2 , b : 0

w (weight, 가중치), b (bias, 편향)
y = H (Hyperthesis, 가설)

가중치와 편향 구하기

```
import torch        # 설치: pip install torch
                    # 확인: pip list
import torch.nn as nn
import torch.nn.functional as f

torch.manual_seed(1)    # 생성하는 random number 고정

# 데이터셋 정의
# (입력값, 정답(Ground Truth))
# (1, 2), (2, 4), (3, 6)
x_train = torch.Tensor([[1], [2], [3]])
y_train = torch.Tensor([[2], [4], [6]])
print(type(x_train), x_train)
print(type(y_train), y_train)

# 모델 정의
model = nn.Linear(1, 1)    # 모델 선언 및 초기화
                           # input_dim = 1, output_dim = 1
print(list(model.parameters()))    # 첫번째값이 w, 두번째값이 b  (랜덤 초기화)
                                   # requires_grad=True : 학습의 대상

# 최적화 알고리즘 정의. 경사하강법 SGD, learning rate = 0.01
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

# 학습. Cost가 0에 가까울수록 학습이 잘 된 것
epochs = 2000  # 1 에폭: 데이터셋을 1회 학습
for epoch in range(epochs + 1):
    prediction = model(x_train)

    cost = f.mse_loss(prediction, y_train)       # 예측값 – 정답
    optimizer.zero_grad()
    cost.backward()    # 기울기 구함
    optimizer.step()    # 가중치 업데이트 (w – 학습률 x 기울기)
    if epoch % 100 == 0:
        print('Epoch {:4d}/{} Cost: {:.6f}'.format(epoch, epochs, cost.item()))

# 예측(Prediction)
while True:
    a=input()
    new_x = torch.Tensor([[float(a)]])
    pred_y = model(new_x)
    print(pred_y)
```

PyTorch – Multivariate Linear Regression

y = w1x1 + w2x2 + w3x3 + b : 다중 선형회귀

w (weight, 가중치), b (bias, 편향)
y = H (Hyperthesis, 가설)

가중치와 편향 구하기

```
import torch        # 설치: pip install torch
                    # 확인: pip list
import torch.nn as nn
import torch.nn.functional as f

torch.manual_seed(1)    # 생성하는 random number 고정

# 데이터셋 정의
x_train = torch.Tensor([[73, 80, 75],
                        [93, 88, 93],
                        [89, 91, 90],
                        [96, 98, 100],
                        [73, 66, 70]])
y_train = torch.Tensor([[152], [185], [180], [196], [142]])

# 모델 정의
model = nn.Linear(3, 1)    # 모델 선언 및 초기화
                           # input_dim = 3, output_dim = 1
print(list(model.parameters()))    # 첫번째값이 w, 두번째값이 b  (랜덤 초기화)
                                   # requires_grad=True : 학습의 대상

# 최적화 알고리즘 정의. 경사하강법 SGD, learning rate = 0.00001 (1e-5)
optimizer = torch.optim.SGD(model.parameters(), lr=0.00001)

# 학습. Cost가 0에 가까울수록 학습이 잘 된 것
epochs = 2000  # 1 에폭: 데이터셋을 1회 학습
for epoch in range(epochs + 1):
    prediction = model(x_train)

    cost = f.mse_loss(prediction, y_train)       # 예측값 – 정답
    optimizer.zero_grad()
    cost.backward()    # 기울기 구함
    optimizer.step()
    if epoch % 100 == 0:
        print('Epoch {:4d}/{} Cost: {:.6f}'.format(epoch, epochs, cost.item()))

# 예측(Prediction)
new_x = torch.Tensor([[73, 80, 75]])
pred_y = model(new_x)
print(pred_y)
print(list(model.parameters()))
```