Homework #4

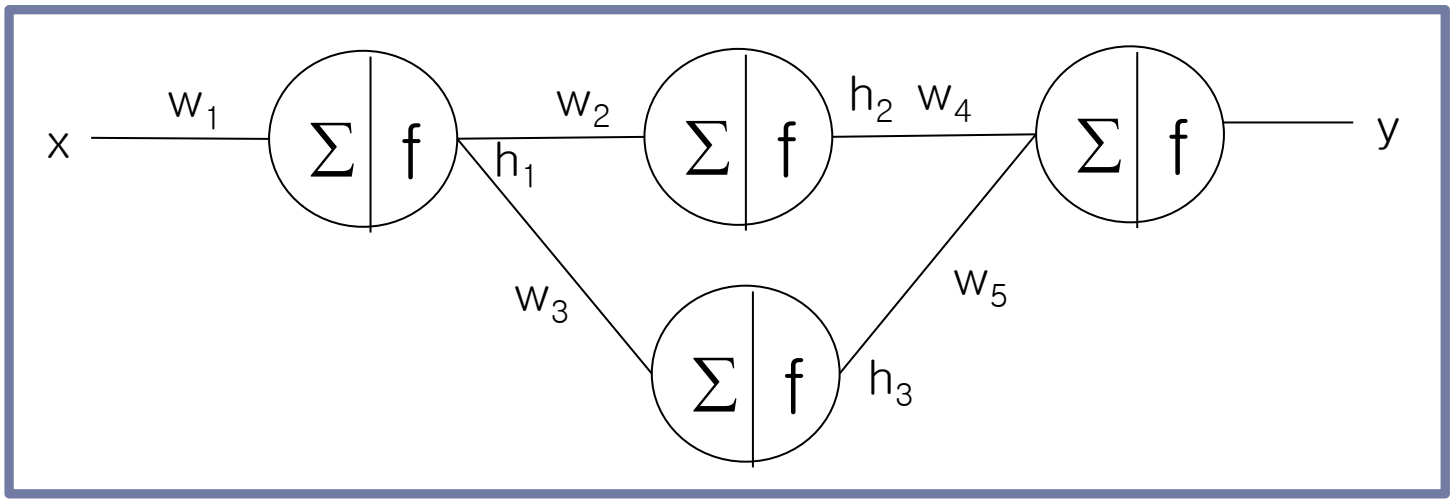
[reference] https://github.com/deeplearningzerotoall/PyTorch

설명

- Part 0 – 문제 풀이 제출
- Part 1 - 예제를 활용하여 공부하고 결과 스크린샷 제출
  - Python
  - Numpy
  - Pytorch(tensor)
- Part 2 – Exercise를 해결할 수 있는 코드 제출
  - Q1 ~ Q8
- Part 0~2의 결과물을 하나의 PDF 파일로 취합하여 하나의 PDF 파일 제출

Part 0

- 하나의 학습데이터 $(x, t) = (1,1)$가 주어지고, 모든 weight들의 초기 값이 모두 1이고, 학습률이 0.1 일때, EBP로 한번 update한 후의 모든 weight 값들을 구하시오. 과정과 값을 보이시오.



Part 1 – Python

1-1. Python: List

다음 코드의 결과를 확인하여 List에 대하여 공부하시오.

```
list1 = [1, 2, 3]

#인덱스가 음수일 경우 리스트의 끝에서부터 셈
type(list1)
list1[0], list1[-1]

#리스트는 자료형이 다른 요소도 저장 가능
list1[1] = "str1"

#range Type출력
list2 = range(10)
type(list2)

#List Type으로 변환
list2 = list(list2)
type(list2)
```

Part 1 – Python

1-2. Python: List

다음 코드의 결과를 확인하여 List에 대하여 공부하시오.

```
# 인덱스 2에서 4(제외)까지 슬라이싱
list2[2:4]

# 인덱스 2에서 끝까지 슬라이싱
list2[2:]
# 처음부터 인덱스 2(제외)까지 슬라이싱
list2[:2]

# 전체 리스트 슬라이싱
list2[:]

# 2씩 증가하면서 리스트 출력
list2[::2]

# 슬라이싱 인덱스는 음수도 가능
list2[:-1]

# 슬라이스된 리스트에 새로운 리스트 할당
list2[2:4] = [8, 9]

# 결과를 예측해 보시오
list2[2:5] = [8, 9]
```

Part 1 – Python

1-3. Python: For Loop with List

다음 코드의 결과를 확인하여 For Loop에 대하여 공부하시오.

```
# List는 순서를 갖기 때문에 for loop 시 index 0 부터 순서대로 접근
animals = ['cat', 'dog', 'monkey']

# 실행 결과를 확인하시오
for animal in animals:
    print(animal)

nums = [0, 1, 2, 3, 4]
squares = []

# 실행 결과를 확인하시오
for x in nums:
    squares.append(x ** 2)
print(squares)
```

Part 1 – Python

1-4. Python: Function

다음 코드의 결과를 확인하여 Function에 대하여 공부하시오.

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

# 실행 결과를 확인하시오.
for x in [-1, 0, 1]:
    print(sign(x))
```

Part 1 – Python

1-5. Python: Function

다음 코드의 결과를 확인하여 Function에 대하여 공부하시오.

```
def hello(name, loud=False):
    if loud:
        print ('HELLO, %s!' % name.upper())
    else:
        print ('Hello, %s' % name)

# 실행 결과를 확인하시오
hello('Bob')

# 실행 결과를 확인하시오
hello('Fred', loud=True)
```

Part 1 – Python

- ## 1-6. Python: Zip
  - ### 다음 코드의 결과를 확인하여 zip에 대하여 공부하시오.

```
# 1 4 7 / 2 5 8 / 3 6 9 출력
for x, y, z in zip([1, 2, 3], [4, 5, 6], [7, 8, 9]):
    print(x, y, z)

# 에러가 나는 이유는 ?
for x, y in zip([1, 2, 3], [4, 5, 6], [7, 8, 9]):
    print(x, y, z)
```

Part 1 – Python

- ## 1-7. Python: Magic method __init__()
  - 다음 코드의 결과를 확인하여 _init_ 함수에 대하여 공부하시오.

```
class HelloWorld():
    def __init__(self):
        print("init")

# 실행 결과를 확인하시오
helloworld = HelloWorld()
```

Part 1 – Python

- 1-8. Python: Magic method __call__()
  - 다음 코드의 결과를 확인하여 __call__ 함수에 대하여 공부하시오.

```
class HelloWorld():
    def __init__(self):
        print("init")

helloworld = HelloWorld()

# 실행 결과를 확인하시오
callable(helloworld) # False
helloworld() # Error

class HelloWorld():
    def __init__(self):
        print("init")

    def __call__(self):
        print("Hello world")

helloworld = HelloWorld()

# 실행 결과를 확인하시오
helloworld()
```

__init__ 함수와 __call__ 함수의 차이점을 서술하여 해당 예제 스크린샷과 함께 첨부하시오.

Part 1 – Numpy

1-9. Numpy: array

다음 코드의 결과를 확인하여 numpy의 array에 대하여 공부하시오.

```
import numpy as np

# rank가 1인 배열 생성
a = np.array([1, 2, 3])

# 각 라인 별 실행 결과를 확인하시오
type(a)
a.shape
a.ndim
a[0], a[1], a[2]

# 요소 변경 후 실행 결과를 확인하시오.
a[0] = 5
a

# rank가 2인 배열 생성
b = np.array([[1,2,3],[4,5,6]])

# 각 라인 별 실행 결과를 확인하시오
b.shape
b.ndim
b[0, 0], b[0, 1], b[1, 0]
```

Part 1 – Numpy

1-10. Numpy: zeros, ones, full, eye, random

- 다음 코드의 결과를 확인하여 numpy의 zeros, ones, full, eye, random 에 대하여 공부하시오.

```
import numpy as np

# 모든 값이 0인 배열 생성 후 실행 결과를 확인하시오.
a = np.zeros((2,2))
a

# 모든 값이 1인 배열 생성 후 실행 결과를 확인하시오.
b = np.ones((1,2))
b

# 모든 값이 특정 상수인 배열 생성 후 실행 결과를 확인하시오.
c = np.full((2,2), 7)
C

# 2x2 단위행렬 생성 후 실행 결과를 확인하시오.
d = np.eye(2)
d

# 임의의 값으로 채워진 배열 생성 후 실행 결과를 확인하시오.
e = np.random.random((2,2))
e
```

Part 1 – Numpy

- ## 1-11. Numpy: where
  - ### 다음 코드의 결과를 확인하여 numpy의 where 에 대하여 공부하시오.

```
import numpy as np

# 0~9를 갖는 배열 생성
a = np.arange(10)


# 실행 결과를 확인하시오.
np.where(a<5)

# 실행 결과를 확인하시오.
np.where(a<5, a, 10*a)

# 실행 결과를 확인하시오.
np.where(a<4, a, -1)
```

Part 1 – Numpy

1-12. Numpy: array slice

- 다음 코드의 결과를 확인하여 numpy의 array slice 에 대하여 공부하시오.

```
import numpy as np

# 3x4 배열 생성
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 슬라이싱을 이용하여 첫 두 행과 1열, 2열로 이루어진 부분배열 생성
# b는 shape가 (2,2)인 배열이 됨
# 실행 결과를 확인하시오.
b = a[:2, 1:3]

# 슬라이싱된 배열은 원본 배열과 같은 데이터를 참조, 즉 슬라이싱 된 배열을 수정하면 원본 배열 역시 수정됨
# 실행 결과를 확인하시오.
a[0, 1]

# b[0, 0]은 a[0, 1]과 같은 데이터
b[0, 0] = 77

# 실행 결과를 확인하시오.
a[0, 1]
```

Part 1 – Numpy

1-13. Numpy: array reshape

- 다음 코드의 결과를 확인하여 numpy의 array reshape 에 대하여 공부하시오.

```
import numpy as np

# 3x4 배열 생성
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 3x4 배열을 2x6으로 변형
b = np.reshape(a, (2, 6))

# 실행 결과를 확인하시오.
b

# 슬라이싱된 배열과 마찬가지고 reshape된 배열도 원본 배열과 같은 데이터를 참조, 즉 슬라이싱 된 배열을 수정하면 원본 배열 역시 수정됨
# 실행 결과를 확인하시오.
a[0, 1]

# b[0, 0]은 a[0, 1]과 같은 데이터
b[0, 0] = 77

# 실행 결과를 확인하시오.
a[0, 1]
```

Part 1 – Numpy

- ## 1-14. Numpy: array reshape
  - 다음 코드의 결과를 확인하여 numpy의 array reshape 에 대하여 공부하시오.

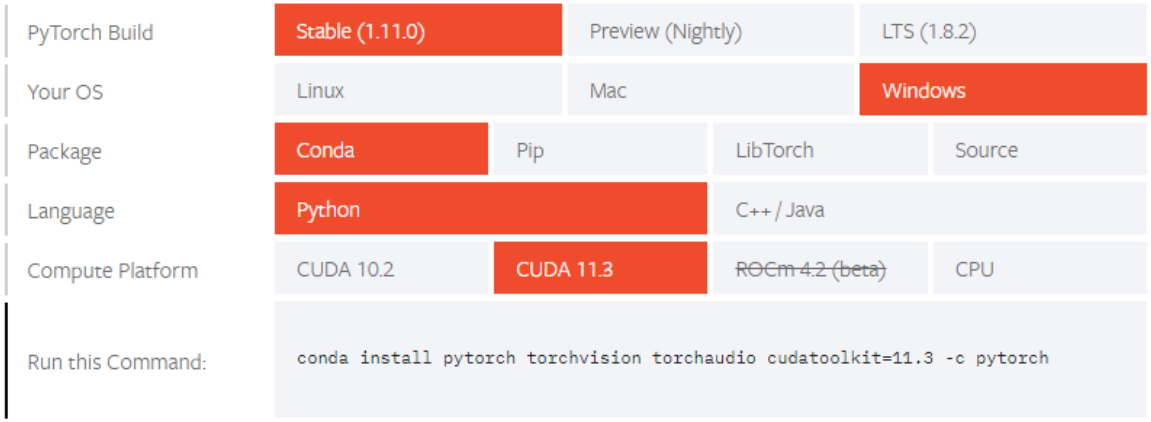
```
import numpy as np

# 2x2 행렬 두 개 생성
a = np.array([[1, 0], [0, 1]])
b = np.array([[4, 1], [2, 2]])

# 2x2 행렬 간 dot product, 실행 결과를 확인하시오.
np.dot(a, b)
```

Part 1 – Pytorch

- **Pytorch: 설치**
  - **Pytorch가 설치되어 있다면 다음 페이지로 이동 (해당 페이지는 스크린샷 필요 없음)**

```
import torch
import numpy as np
```

torch 라이브러리가 없는 경우 아래 사이트를 참고하여 본인의 환경에 맞게 설치하여 진행
https://pytorch.org/get-started/locally/



Part 1 – Pytorch

- ## 1-15. Pytorch: Tensor
  - 다음 코드의 결과를 확인하여 pytorch의 tensor에 대하여 공부하시오.

```
import torch
import numpy as np

# Float 형을 갖는 tensor 생성
t1 = torch.FloatTensor([0, 1, 2, 3, 4, 5, 6])
t2 = torch.tensor(np.arange(7)) # t1과 동일

# 각 라인 별 실행 결과를 확인하시오
t1.shape
t2.shape
t1.dim()
t1.size()
t1[:2]
t1[3:]
```

Part 1 – Pytorch

- 1-16. Pytorch: numpy의 array <-> pytorch의 tensor
  - 다음 코드의 결과를 확인하여 numpy의 array와 pytorch의 tensor간의 상호 변환에 대하여 공부하시오.

```
import torch
import numpy as np

# 각 라인 별 실행 결과를 확인하시오
b = np.arange(7)
t1 = torch.FloatTensor([0, 1, 2, 3, 4, 5, 6])
type(b)
type(t1)

# numpy의 array인 b를 torch의 tensor로 변환
tt = torch.tensor(b)
t_from =torch.from_numpy(b)

# 각 라인 별 실행 결과를 확인하시오
type(tt)
type(t_from)
tt
t_from
```

Part 1 – Pytorch

- **1-17. Pytorch: numpy의 array <-> pytorch의 tensor**
  - 다음 코드의 결과를 확인하여 numpy의 array와 pytorch의 tensor간의 상호 변환에 대하여 공부하시오.

```
# 이 전 슬라이드와 이어서 작성하시오.

# 각 라인 별 실행 결과를 확인하시오.
b[0]= -10
tt
t_from

# torch의 tensor를 numpy의 array로 바꿈 / 실행 결과를 확인하시오.
t_to_np = t_from.numpy()
type(t_to_np)
```

Part 1 – Pytorch

1-18. Pytorch: Broadcasting

- 다음 코드의 결과를 확인하여 broadcasting에 대하여 공부하시오.

```
import torch
import numpy as np

m1 = torch.FloatTensor([[3, 3]])
m2 = torch.FloatTensor([[2, 2]])

# 실행 결과를 확인하시오
m1 + m2

# Vector + scalar
m1 = torch.FloatTensor([[1, 2]])
m2 = torch.FloatTensor([3])

# 실행 결과를 확인하시오
m1 + m2

# 2 x 1 Vector + 1 x 2 Vector
m1 = torch.FloatTensor([[1, 2]])
m2 = torch.FloatTensor([[3], [4]])

# 실행 결과를 확인하시오
m1 + m2
```

Part 1 – Pytorch

1-19. Pytorch: Mul vs MatMul

- 다음 코드의 결과를 확인하여 torch의 Mul과 MatMul에 대하여 공부하시오.

```
m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])

# 각 라인 별 실행 결과를 확인하시오.
m1 * m2
m1.mul(m2)

m1 = torch.FloatTensor([[1, 2], [3, 4]])
m2 = torch.FloatTensor([[1], [2]])

# 실행 결과를 확인하시오.
m1.matmul(m2)
```

Part 1 – Pytorch

- ## 1-20. Pytorch: View (Reshape in Numpy)
  - 다음 코드의 결과를 확인하여 torch의 view에 대하여 공부하시오.

```
import torch
import numpy as np

t = np.arange(12).reshape(-1, 2, 3)
floatT = torch.FloatTensor(t)

# 각 라인 별 실행 결과를 확인하시오.
floatT.shape
floatT.view([-1, 3])
```

Part 2- Exercise

- **2-1. numpy를 활용하여 10부터 49까지를 값으로 갖는 배열을 생성하시오 (ex. 10, 11, ... 49)**

- **2-2. numpy를 활용하여 49부터 10까지를 값으로 갖는 배열을 생성하시오 (ex. 49, 48, ... 10)**

- **2-3. 아래 예제와 같이 [입력]이 주어졌을 때, 'A'와 'B'의 요소가 같은 위치(index)를 갖는 배열을 구하시오 (Hint: np.where)**

  [입력]

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

  B = np.array([7,2,10,2,7,4,9,4,9,8])

  [출력]

  Array([???])

Part 2- Exercise

- **2-4. 아래 예제와 같이 [입력]이 주어졌을 때, Broadcasting 성질을 활용하여 행렬 'a'에 1을 더한 결과를 구하시오**

  [입력]

  a = np.arange(9).reshape(3, 3)

  [출력]

  [[? ? ?] [? ? ?] [? ? ?]]

- **2-5. 아래 예제와 같이 [입력]이 주어졌을 때 [출력]이 나오도록, 1열과 2열을 바꾸는 코드를 작성하시오**

  [입력]

  a = np.arange(9).reshape(3, 3)

  [출력]

  [[3 4 5] [0 1 2] [6 7 8]]

Part 2- Exercise

- **2-6. 아래와 같은 결과가 나오기 위한 변수 'a'를 선언하시오 (Hint: np.arange(n))**

```
a = ??? #Problem

print("Shape: ", a.shape)
print("Value: ", a)


#Output
Shape:  (2, 3, 4)
Value:  [[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

Part 2- Exercise

- 2-7. 문제2-6에서 선언한 변수 'a'를 활용하여 다음과 같은 결과를 만드는 slice 부분을 작성하시오.

```
print(a[???]) #Problem


#Output
[[ 6  7]
 [10 11]]
```

Part 2- Exercise

- **2-8. 아래와 같은 Output을 출력하도록 torch의 view 함수를 작성하시오.**

```
import torch
import numpy as np

t = np.arange(24)
print(t.shape)

floatT = torch.FloatTensor(t)
print(floatT.view([???])) #Problem
```

Output

```
(24,)

tensor([[13., 16.], [19., 22.]])
```