

4주차(3/3)

퍼셉트론 코딩

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

퍼셉트론 코딩

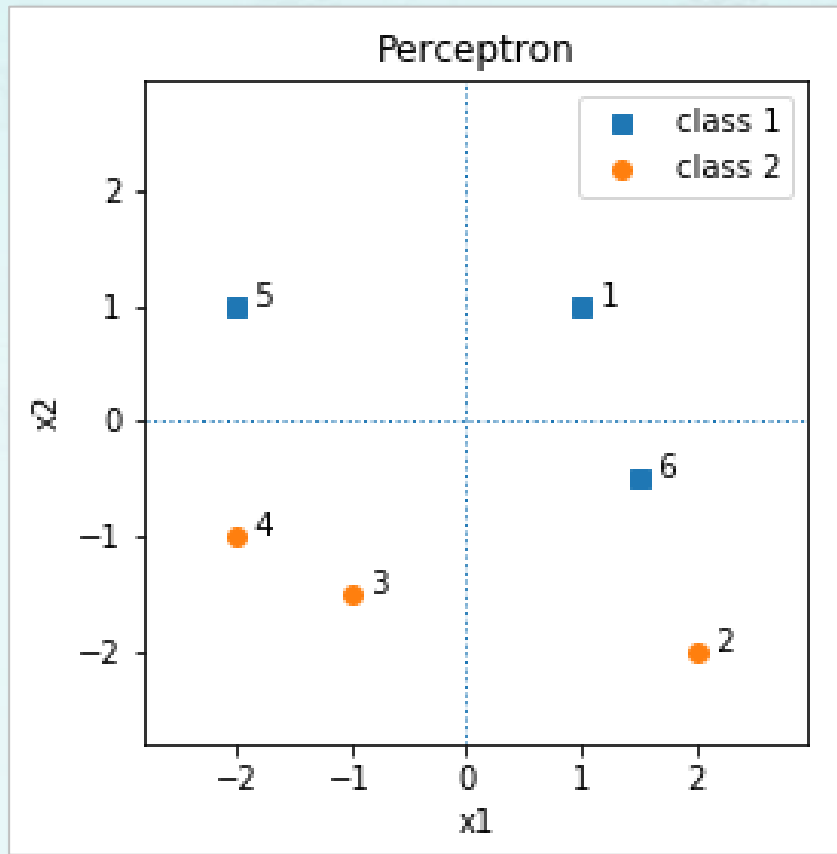
- 학습 목표
 - 퍼셉트론을 파이썬으로 구현한다.
- 학습 내용
 - 학습 자료의 준비
 - 학습 자료로 연산하기
 - 학습률과 에포크
 - 퍼셉트론 함수 구현

1. 퍼셉트론 예제: 전체 개괄

- 퍼셉트론 예제 코딩으로 시작
- 최종적으로 퍼셉트론 함수 완성

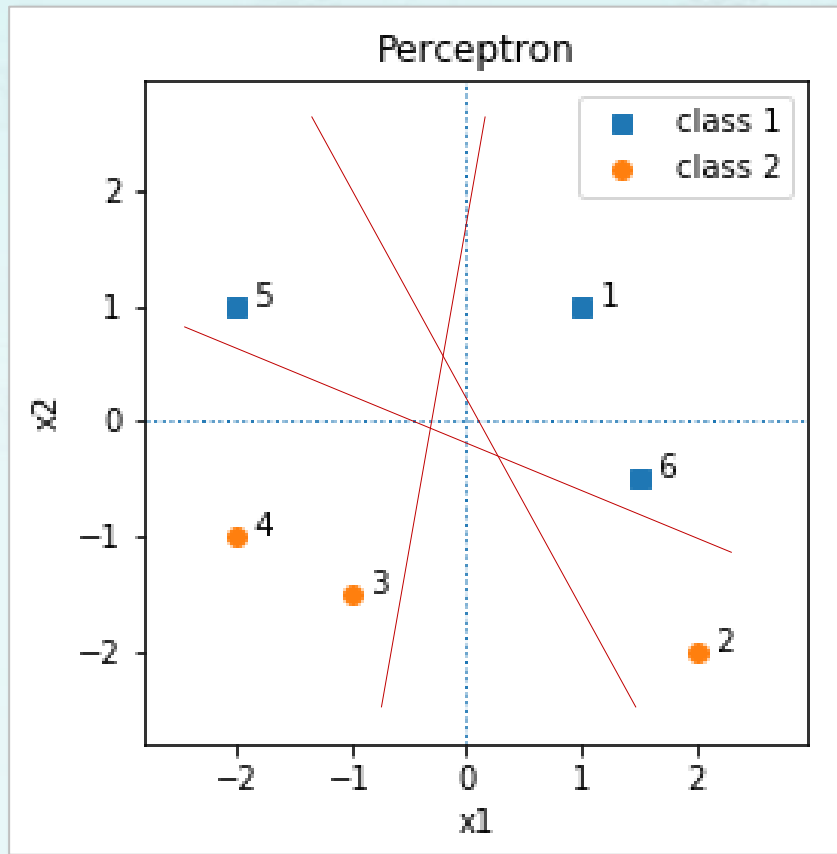
1. 퍼셉트론 예제: 학습자료와 목표

- 6개의 학습자료
- 클래스 레이블 $y = [1, -1, -1, -1, 1, 1]$



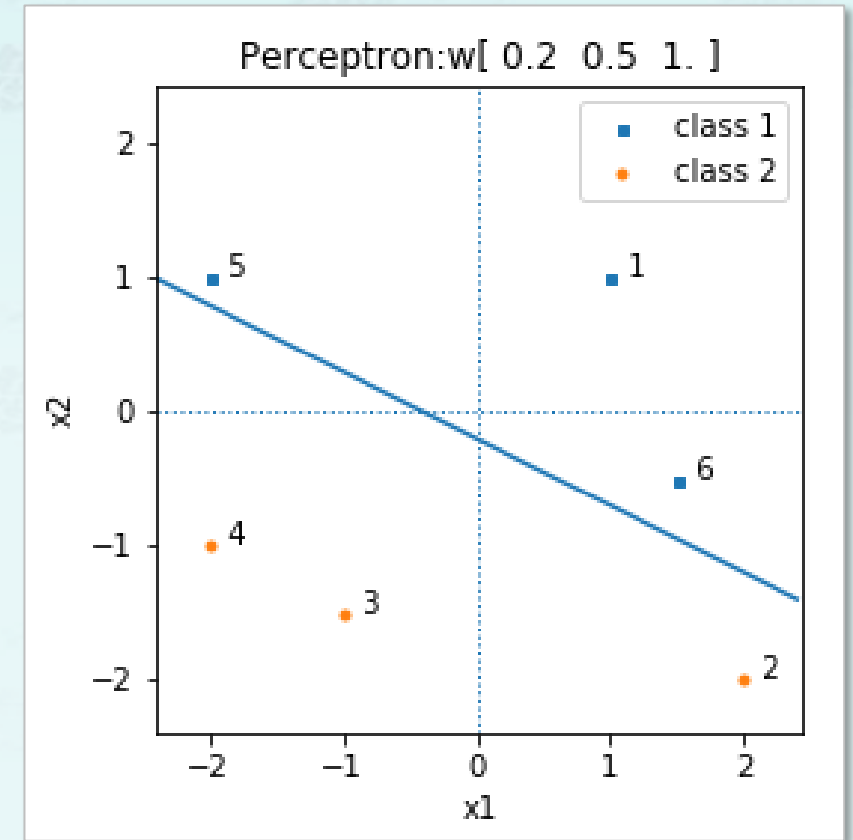
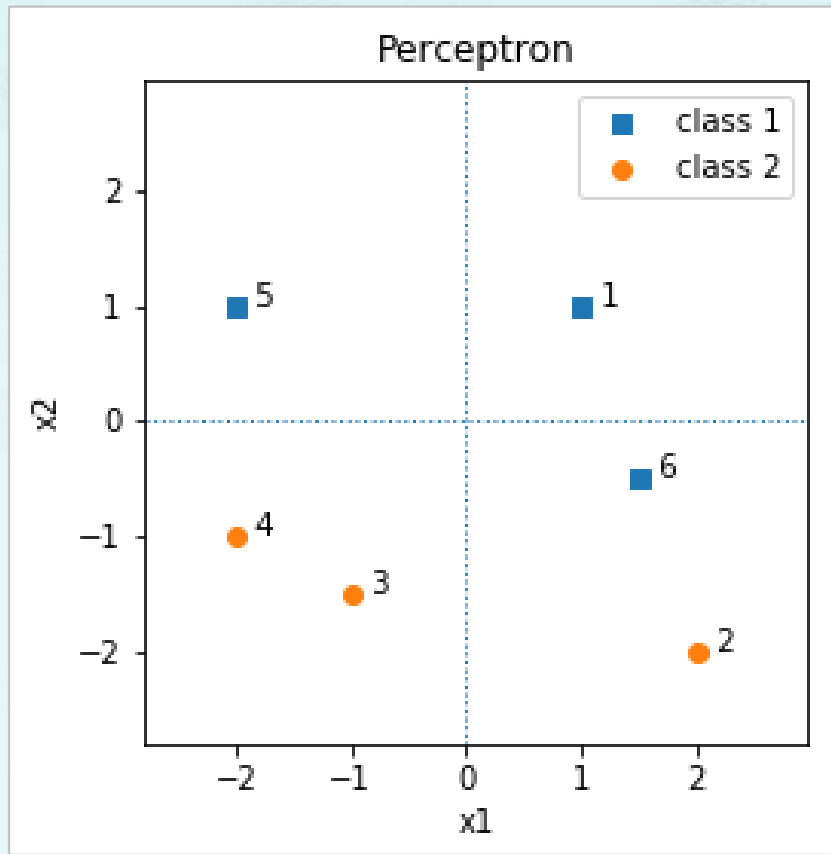
1. 퍼셉트론 예제: 학습자료와 목표

- 6개의 학습자료
- 클래스 레이블 $y = [1, -1, -1, -1, 1, 1]$



1. 퍼셉트론 예제: 학습자료와 목표

- 6개의 학습자료
- 클래스 레이블 $y = [1, -1, -1, -1, 1, 1]$



1. 퍼셉트론 예제: 조건

- 초기 가중치 \mathbf{w} :
 - $\mathbf{w}^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 \mathbf{x} :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 \mathbf{y} :
 - $\mathbf{y} = [1, -1, -1, -1, 1, 1]$

2. 학습자료 준비: 입력 자료

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
```


2. 학습자료 준비: 입력 자료

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
```

```
[[ 1. ,  1. ],
 [ 2. , -2. ],
 [-1. , -1.5],
 [-2. , -1. ],
 [-2. ,  1. ],
 [ 1.5, -0.5]]
```

2. 학습자료 준비: 입력 자료

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
```

```
[[ 1. ,  1. ],
 [ 2. , -2. ],
 [-1. , -1.5],
 [-2. , -1. ],
 [-2. ,  1. ],
 [ 1.5, -0.5]]
```



```
[[ 1. ,  1. ,  1. ],
 [ 1. ,  2. , -2. ],
 [ 1. , -1. , -1.5],
 [ 1. , -2. , -1. ],
 [ 1. , -2. ,  1. ],
 [ 1. ,  1.5, -0.5]]
```

2. 학습자료 준비: 입력 자료

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
```

```
[[ 1. ,  1. ],
 [ 2. , -2. ],
 [-1. , -1.5],
 [-2. , -1. ],
 [-2. ,  1. ],
 [ 1.5, -0.5]]
```



```
[[ 1. ,  1. ,  1. ],
 [ 1. ,  2. , -2. ],
 [ 1. , -1. , -1.5],
 [ 1. , -2. , -1. ],
 [ 1. , -2. ,  1. ],
 [ 1. ,  1.5, -0.5]]
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```

2. 학습자료 준비: 클래스 레이블

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```

2. 학습자료 준비: 초기 가중치 설정

- 초기 가중치 w :
 - $w^T = [0 \ 1 \ 0.5]$
- 학습률 $\eta = 0.1$
- 학습자료 입력 x :
 - $x^{(1)} = [1, 1]$
 - $x^{(2)} = [2, -2]$
 - $x^{(3)} = [-1, -1.5]$
 - $x^{(4)} = [-2, -1.0]$
 - $x^{(5)} = [-2.0, 1.0]$
 - $x^{(6)} = [1.5, -0.5]$
- 클래스레이블 y :
 - $y = [1, -1, -1, -1, 1, 1]$

```
w = np.array([0.0, 1.0, 0.5])
```

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```

2. 학습자료 준비: 초기 가중치 설정

- 보기

1. 3

2. `X.shape[1]`

```
w = np.random.random( )
```

```
w = np.array([0.0, 1.0, 0.5])
```

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```

2. 학습자료 준비: 초기 가중치 설정

- 보기

- ✓ 1. (3, 1)

- ✓✓ 2. (X.shape[1], 1)

```
w = np.random.random( )
```

```
w = np.array([0.0, 1.0, 0.5])
```

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```


2. 학습자료 준비: 초기 가중치 설정

- 보기
 - ✓ 1. (3, 1)
 - ✓✓ 2. (X.shape[1], 1)
- x.shape =
- X.shape =

```
w = np.random.random((X.shape[1], 1))
```

```
w = np.array([0.0, 1.0, 0.5])
```

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```


2. 학습자료 준비: 초기 가중치 설정

- 보기

- ✓ 1. (3, 1)

- ✓✓ 2. (X.shape[1], 1)

- x.shape = (6, 2)

- X.shape = (6, 3)

X.shape[0]
len(X)

X.shape[1]

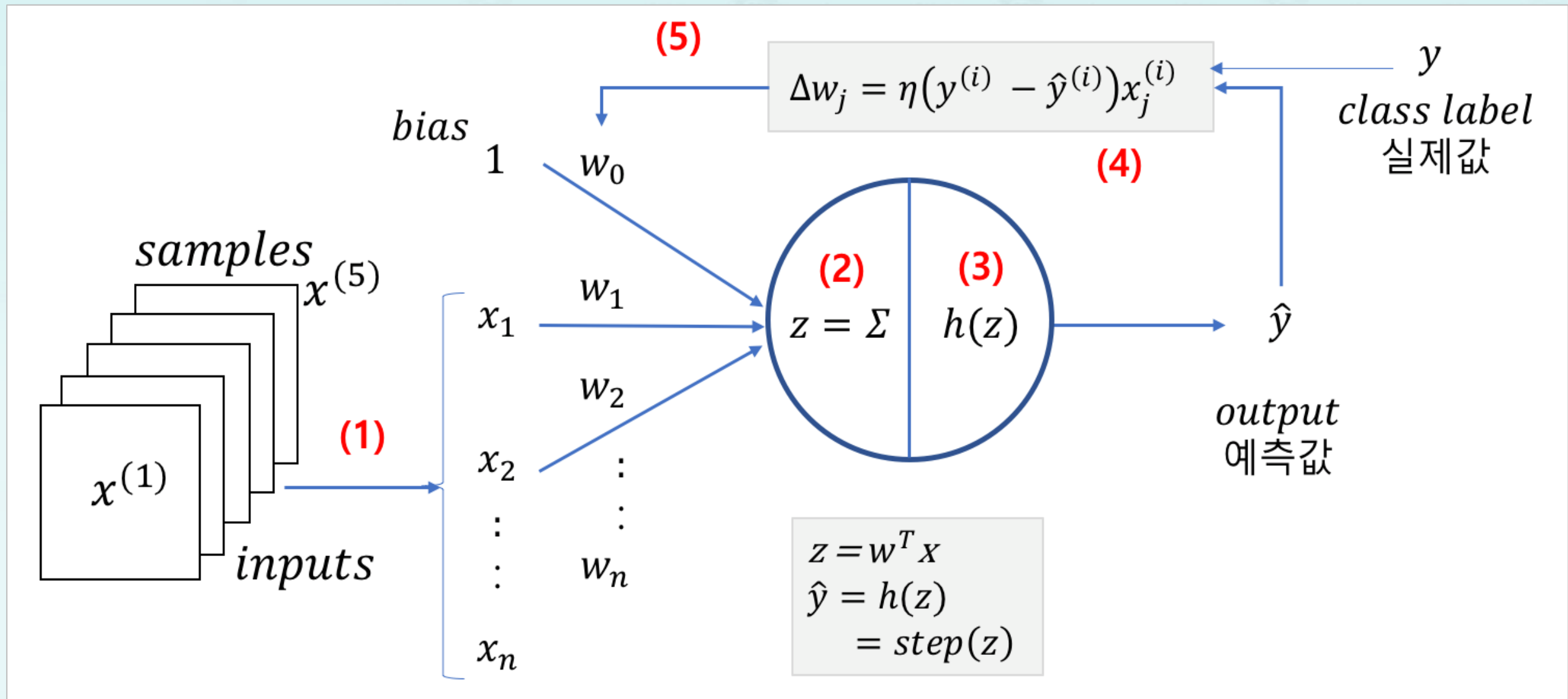
```
w = np.random.random((X.shape[1], 1))
```

```
w = np.array([0.0, 1.0, 0.5])
```

```
y = np.array([1, -1, -1, -1, 1, 1])
```

```
import numpy as np
x = np.array([[1.0, 1.0], [2.0, -2.0],
              [-1.0, -1.5], [-2.0, -1.0],
              [-2.0, 1.0], [1.5, -0.5]])
X = np.c_[ np.ones(len(x)), x ]
```

3. 퍼셉트론 알고리즘 코딩: 전체 과정



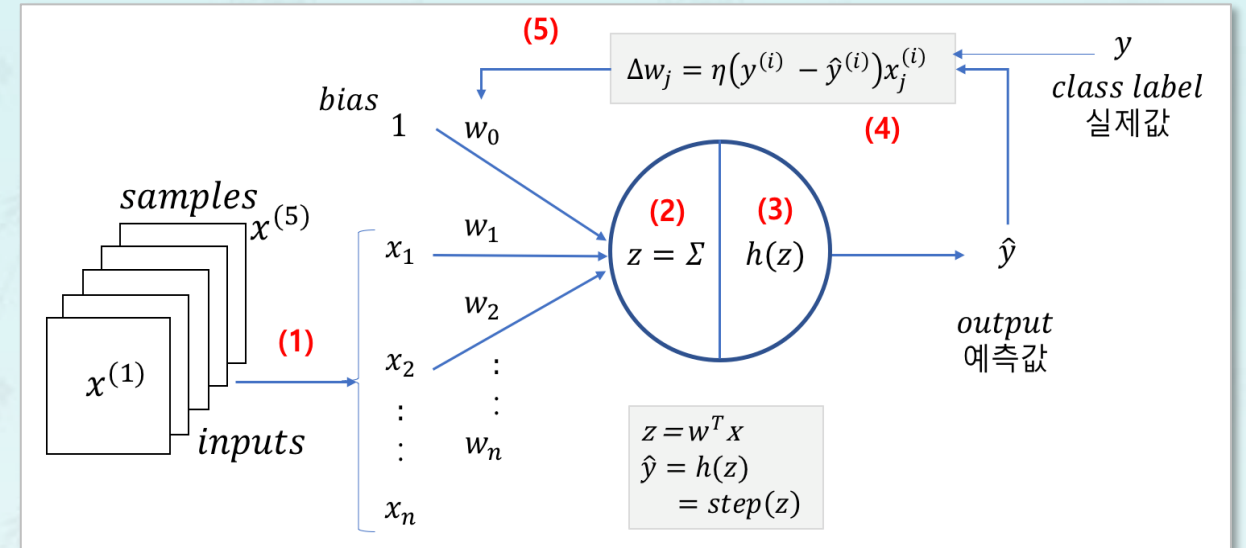
3. 퍼셉트론 알고리즘 코딩: 전체 과정

- 각 학습자료에 대해 $\mathbf{z} = \mathbf{w}^T \mathbf{x}$ 계산
 - $\mathbf{z} = \text{np.dot}(\mathbf{w}, \mathbf{x})$
- 활성화 함수(계단함수)에 \mathbf{z} 적용
 - $\hat{\mathbf{y}} = h(\mathbf{z})$
- 가중치 조정값 계산과 가중치 조정

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (1)$$

$$w_j := w_j + \Delta w_j$$

- 모든 학습자료에 대해 실행



3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 각 학습자료에 대해 $\mathbf{z} = \mathbf{w}^T \mathbf{x}$ 계산
 - `z = np.dot(w.T, x)`
- 활성화 함수(계단함수)에 \mathbf{z} 적용
 - `yhat = h(z)`
- 가중치 조정값 계산과 가중치 조정

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (1)$$

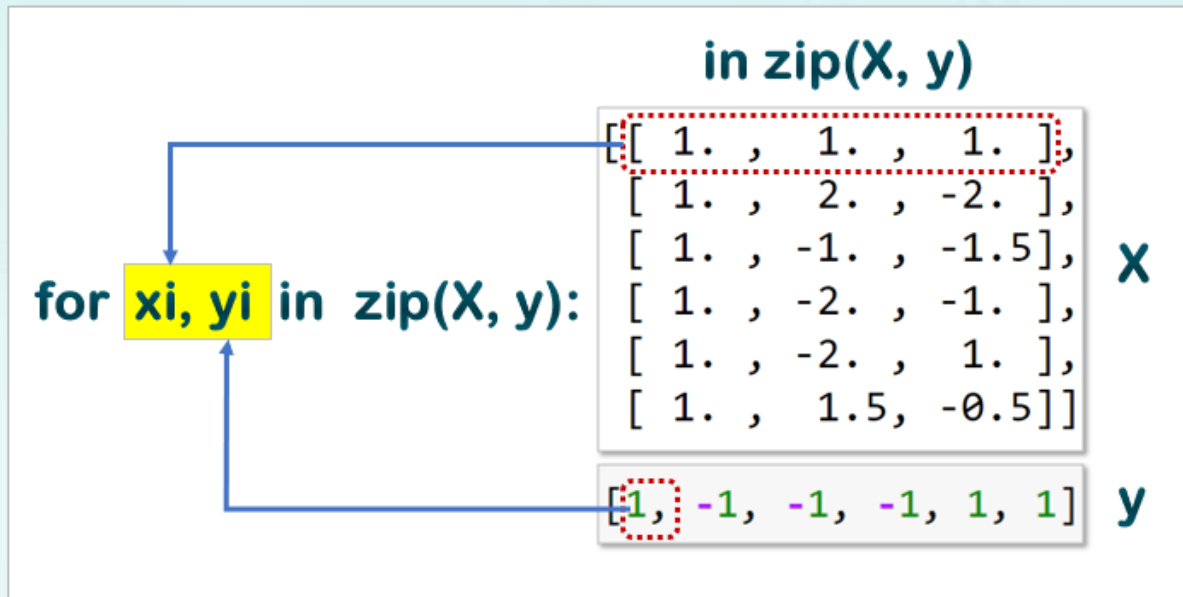
$$w_j := w_j + \Delta w_j$$

- 모든 학습자료에 대해 실행

```
1 eta = 0.1
2 for xi, yi in zip(X, y):
3     xi = xi.reshape(w.shape)
4     z = np.dot(w.T, xi)
5     yhat = np.where(z > 0.0, 1, -1)
6     delta = eta * (yi - yhat) * xi
7     w += delta
8 print(np.round(w, 2))
```

3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 파이썬 **zip()** 함수
 - 동일한 길이의 자료를 묶어줌



```
1 eta = 0.1
2 for xi, yi in zip(X, y):
3     xi = xi.reshape(w.shape)
4     z = np.dot(w.T, xi)
5     yhat = np.where(z > 0.0, 1, -1)
6     delta = eta * (yi - yhat) * xi
7     w += delta
8 print(np.round(w, 2))
```

3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 각 학습자료에 대해 $\mathbf{z} = \mathbf{w}^T \mathbf{x}$ 계산
 - `z = np.dot(w.T, x)`
- 활성화 함수(계단함수)에 \mathbf{z} 적용
 - `yhat = h(z)`
- 가중치 조정값 계산과 가중치 조정

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (1)$$

$$w_j := w_j + \Delta w_j$$

- 모든 학습자료에 대해 실행

```
1 eta = 0.1
2 for xi, yi in zip(X, y):
3     xi = xi.reshape(w.shape) ←
4     z = np.dot(w.T, xi)
5     yhat = np.where(z > 0.0, 1, -1)
6     delta = eta * (yi - yhat) * xi
7     w += delta
8 print(np.round(w, 2))
```

3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 코드 5번줄 - 계단함수

```
if z > 0:  
    yhat = 1  
else:  
    yhat = -1
```

```
1 eta = 0.1  
2 for xi, yi in zip(X, y):  
3     xi = xi.reshape(w.shape)  
4     z = np.dot(w.T, xi)  
5     ➡ yhat = np.where(z > 0.0, 1, -1)  
6     delta = eta * (yi - yhat) * xi  
7     w += delta  
8 print(np.round(w,2))
```

3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 코드 5번줄 - 계단함수

```
if z > 0:  
    yhat = 1  
else:  
    yhat = -1
```



```
yhat = np.where(z > 0, 1, -1)
```

```
1 eta = 0.1  
2 for xi, yi in zip(X, y):  
3     xi = xi.reshape(w.shape)  
4     z = np.dot(w.T, xi)  
5     ➡ yhat = np.where(z > 0.0, 1, -1)  
6     delta = eta * (yi - yhat) * xi  
7     w += delta  
8 print(np.round(w, 2))
```


3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 각 학습자료에 대해 $\mathbf{z} = \mathbf{w}^T \mathbf{x}$ 계산
 - `z = np.dot(w.T, x)`
- 활성화 함수(계단함수)에 \mathbf{z} 적용
 - `yhat = h(z)`
- 가중치 조정값 계산과 가중치 조정

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (1)$$

$$w_j := w_j + \Delta w_j$$

- 모든 학습자료에 대해 실행

```
1 eta = 0.1
2 for xi, yi in zip(X, y):
3     xi = xi.reshape(w.shape)
4     z = np.dot(w.T, xi)
5     yhat = np.where(z > 0.0, 1, -1)
6     → delta = eta * (yi - yhat) * xi
7     w += delta
8     print(np.round(w, 2))
```

3. 퍼셉트론 알고리즘 코딩: 자료 연산

- 각 학습자료에 대해 $\mathbf{z} = \mathbf{w}^T \mathbf{x}$ 계산
 - `z = np.dot(w.T, x)`
- 활성화 함수(계단함수)에 \mathbf{z} 적용
 - `yhat = h(z)`
- 가중치 조정값 계산과 가중치 조정

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (1)$$

$$w_j := w_j + \Delta w_j$$

- 모든 학습자료에 대해 실행

```
1 eta = 0.1
2 for xi, yi in zip(X, y):
3     xi = xi.reshape(w.shape)
4     z = np.dot(w.T, xi)
5     yhat = np.where(z > 0.0, 1, -1)
6     delta = eta * (yi - yhat) * xi
7     w += delta
8 print(np.round(w,2))
```

```
[[0.2]
 [0.5]
 [1.  ]
```

4. 퍼셉트론 전체 코드: 에폭

```
1 x = np.array([[1.0, 1.0], [2.0, -2.0], [-1.0, -1.5],
2               [-2.0, -1.0], [-2.0, 1.0], [1.5, -0.5]])
3 X = np.c_[ np.ones(len(x)), x ]          # samples
4 y = np.array([1, -1, -1, -1, 1, 1])     # class labels
5 w = np.array([0, 1.0, 0.5]).reshape(X.shape[1], 1)
6 #w = np.random.random((X.shape[1],1))   # initial weight
7 maxlabel, minlabel = y.max(), y.min()
8
9 epochs = 1                               ←
10 eta = 0.1
11 for _ in range(epochs):                  ←
12     for xi, yi in zip(X, y):
13         xi = xi.reshape(w.shape)
14         z = np.dot(w.T, xi)
15         yhat = np.where(z > 0.0, maxlabel, minlabel)
16         delta = eta * (yi - yhat) * xi
17         w += delta
18 print(np.round(w,2))
```

4. 퍼셉트론 전체 코드: min/maxlable 변수

```
1 x = np.array([[1.0, 1.0], [2.0, -2.0], [-1.0, -1.5],
2               [-2.0, -1.0], [-2.0, 1.0], [1.5, -0.5]])
3 X = np.c_[ np.ones(len(x)), x ]          # samples
4 y = np.array([1, -1, -1, -1, 1, 1])     # class labels
5 w = np.array([0, 1.0, 0.5]).reshape(X.shape[1], 1)
6 #w = np.random.random((X.shape[1],1))   # initial weight
7 maxlabel, minlabel = y.max(), y.min() ←
8
9 epochs = 1
10 eta = 0.1
11 for _ in range(epochs):
12     for xi, yi in zip(X, y):
13         xi = xi.reshape(w.shape)
14         z = np.dot(w.T, xi)
15         yhat = np.where(z > 0.0, maxlabel, minlabel) ←
16         delta = eta * (yi - yhat) * xi
17         w += delta
18 print(np.round(w,2))
```

5. 퍼셉트론 함수 구현: 함수로 만드는 이유

- 현재 코드
 - 퍼셉트론 기능 가능
 - 간단한 코딩
- 코드 재사용(**Code Reusability**)
 - 함수로 전환

5. 퍼셉트론 함수 구현: 필요한 인자

- 함수 **perceptron** 에 넘겨줄 인자
 - 학습자료 **X**
 - 클래스 레이블 **y**
 - 학습률 **eta**
 - 반복회수 **epochs**
 - 동일난수 위한 시드 **random_seed**


5. 퍼셉트론 함수 구현: 반환값

- 함수 **perceptron** 에 넘겨줄 인자
 - 학습자료 **X**
 - 클래스 레이블 **y**
 - 학습률 **eta**
 - 반복회수 **epochs**
 - 동일난수 위한 시드 **random_seed**
- 함수의 반환 값
 - 가중치


5. 퍼셉트론 함수 구현: 코드

```
1  #%%writefile code/perceptron.py ←
2  def perceptron(X, y, w = None, eta=0.1, epochs=5, random_seed=1):
3      if w is None:
4          np.random.seed(random_seed)
5          w = np.random.random((X.shape[1],1))
6      maxlabel, minlabel = y.max(), y.min()
7      for _ in range(epochs):
8          for xi, yi in zip(X, y):
9              xi = xi.reshape(w.shape)
10             z = np.dot(w.T, xi)
11             yhat = np.where(z >= 0.0, maxlabel, minlabel)
12             delta = eta * (yi - yhat) * xi
13             w += delta
14      return w
```


5. 퍼셉트론 함수 구현: 코드




```
1  #%%writefile code/perceptron.py
2  def perceptron(X, y, w = None, eta=0.1, epochs=5, random_seed=1):
3      if w is None:
4          np.random.seed(random_seed)
5          w = np.random.random((X.shape[1],1))
6      maxlabel, minlabel = y.max(), y.min()
7      for _ in range(epochs):
8          for xi, yi in zip(X, y):
9              xi = xi.reshape(w.shape)
10             z = np.dot(w.T, xi)
11             yhat = np.where(z >= 0.0, maxlabel, minlabel)
12             delta = eta * (yi - yhat) * xi
13             w += delta
14      return w
```



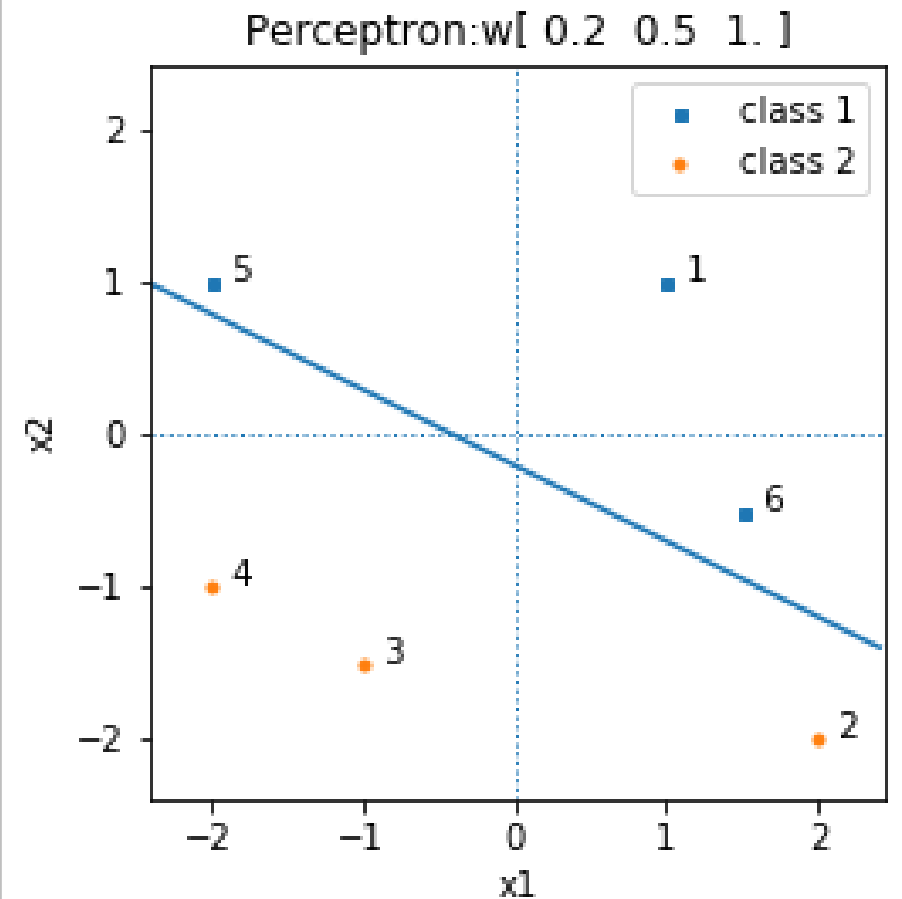
5. 퍼셉트론 함수 구현: 코드

```
1  #%%writefile code/perceptron.py
2  def perceptron(X, y, w = None, eta=0.1, epochs=5, random_seed=1):
3      if w is None:
4          np.random.seed(random_seed)
5          w = np.random.random((X.shape[1],1))
6      maxlabel, minlabel = y.max(), y.min()
7      for _ in range(epochs):
8          for xi, yi in zip(X, y):
9              xi = xi.reshape(w.shape)
10             z = np.dot(w.T, xi)
11             yhat = np.where(z >= 0.0, maxlabel, minlabel)
12             delta = eta * (yi - yhat) * xi
13             w += delta
14      return w
```

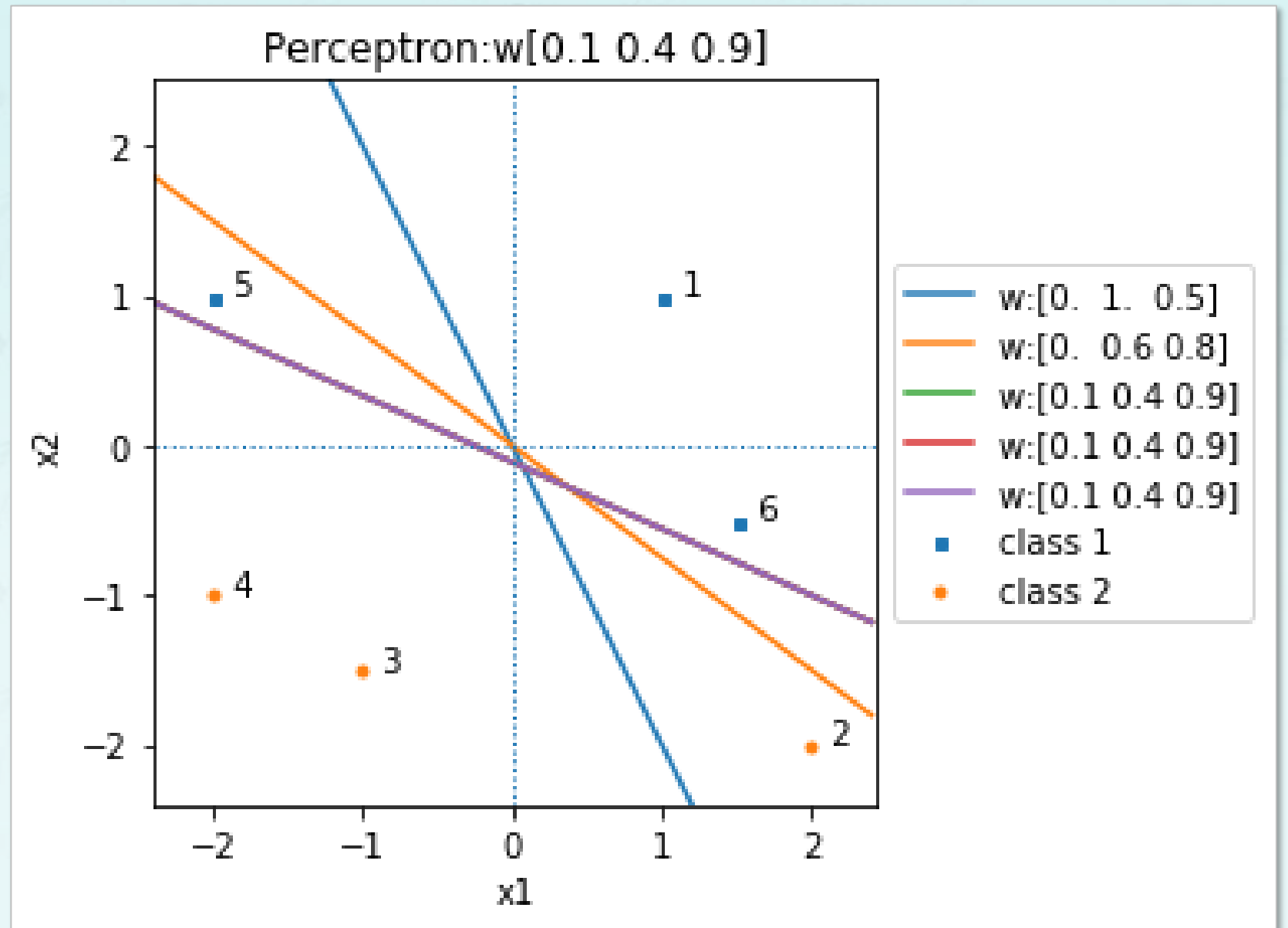


5. 퍼셉트론 함수 구현: 테스트와 시각화

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12 w = perceptron(X, y, w, eta=0.1, epochs=1)
13 plot_xyw(X, y, w, X0=True)
```

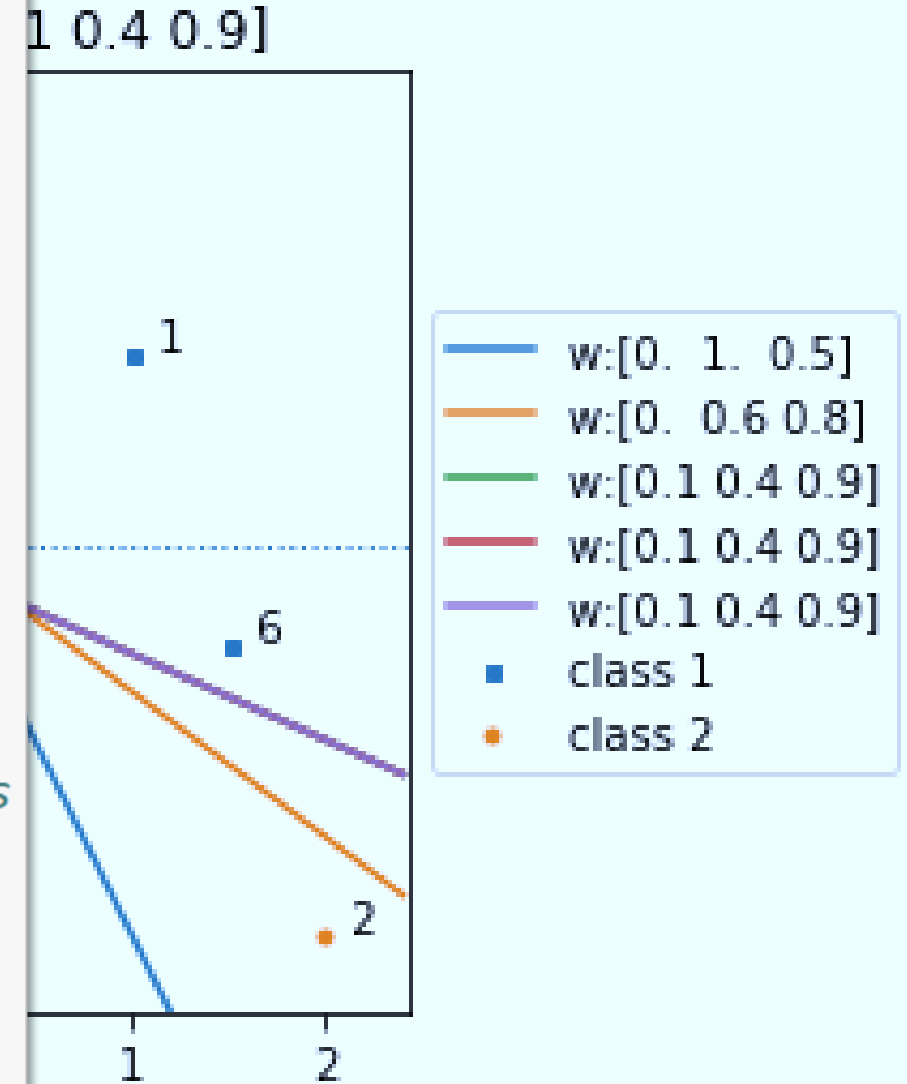


5. 퍼셉트론 함수 구현: 에폭에 따른 판별식 시각화



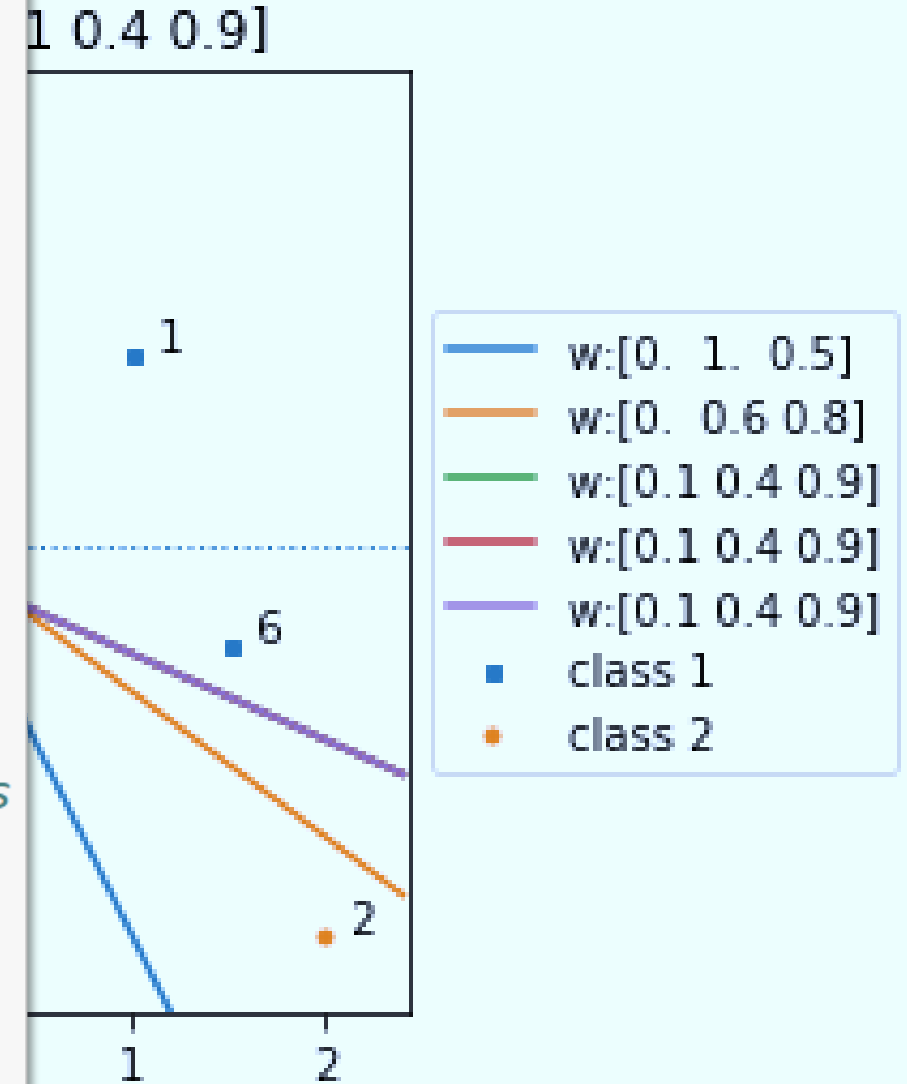
5. 퍼셉트론 함수 구현: 에폭에 따른 판별식 시각화

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12
13 W = np.array([w]) # adding the initial weights
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y, w, eta=0.05, epochs=1)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annotate=True)
```



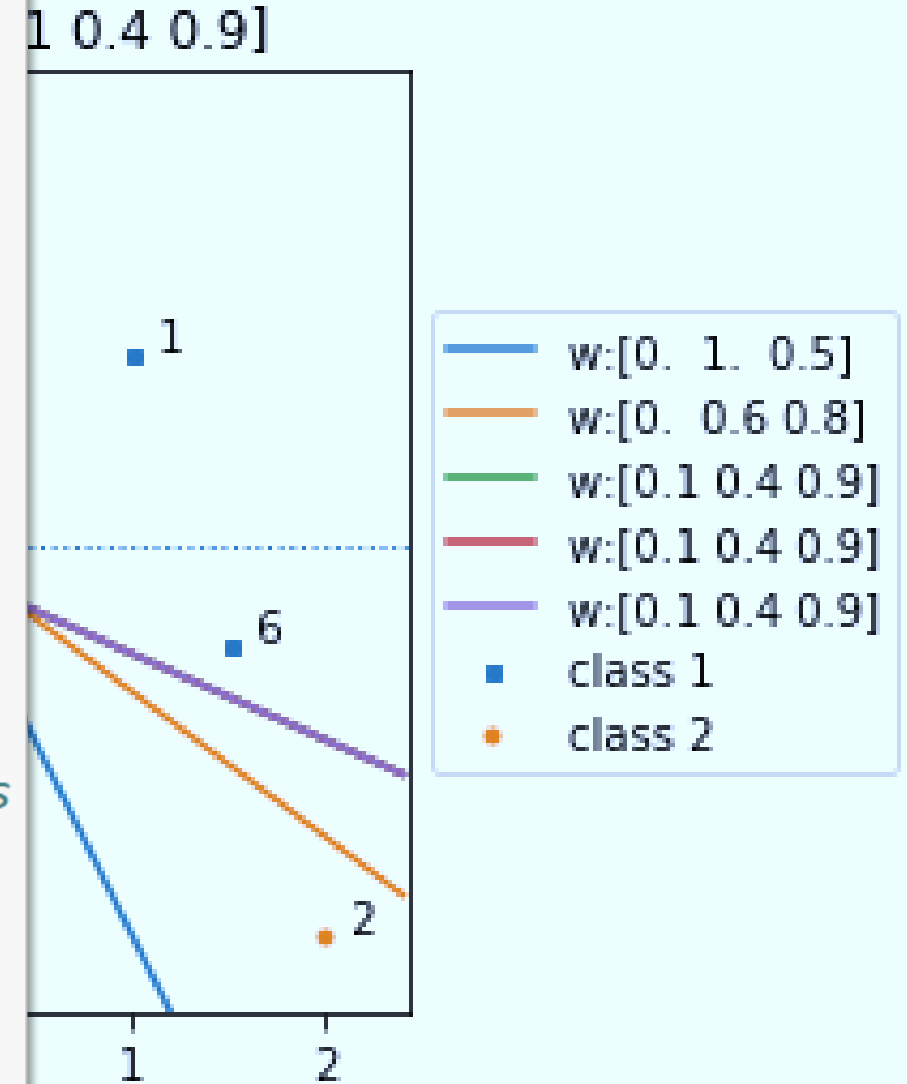
5. 퍼셉트론 함수 구현: 에폭에 따른 판별식 시각화

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12
13 ➡ W = np.array([w]) # adding the initial weights
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y, w, eta=0.05, epochs=1)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annotate=True)
```



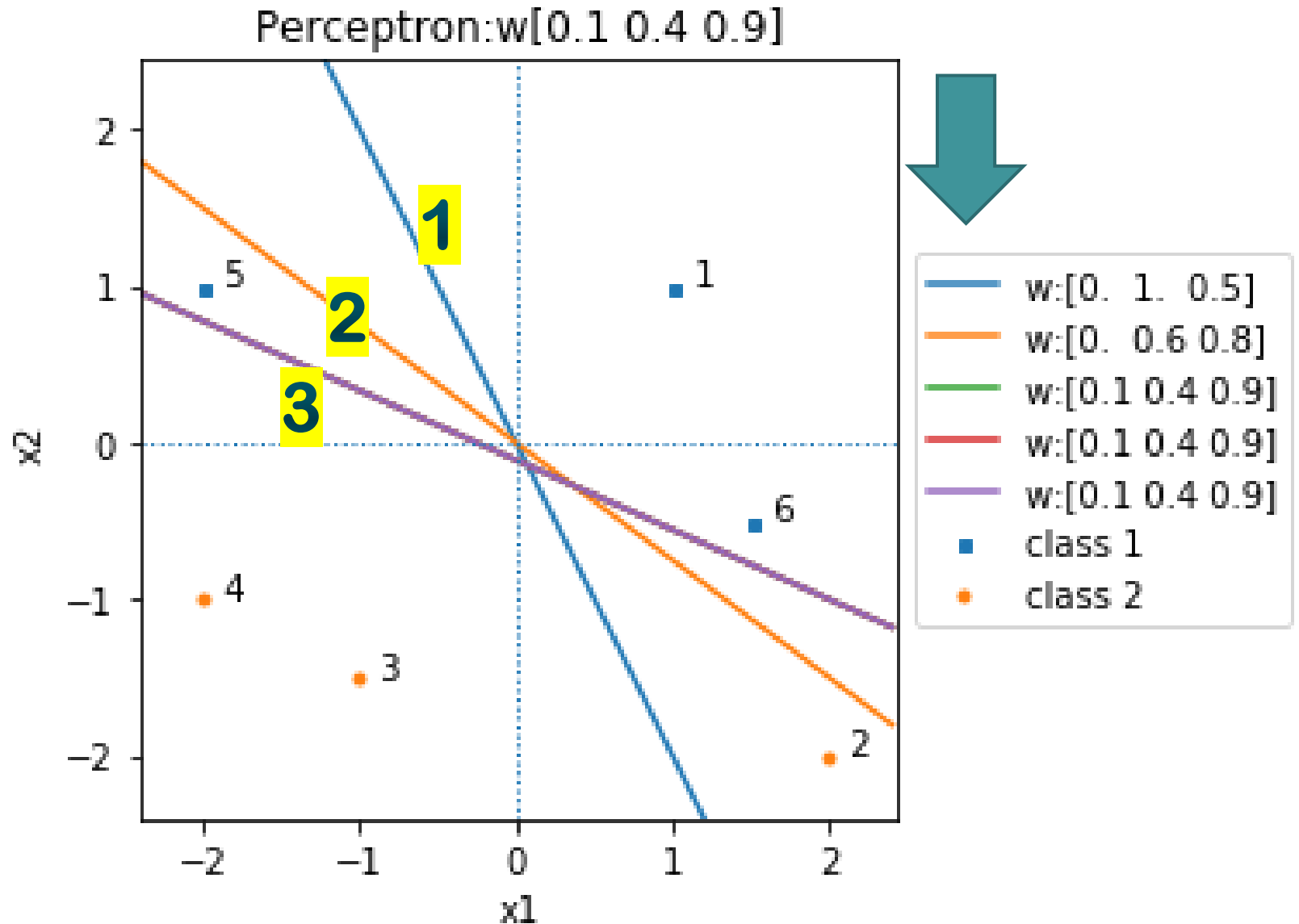
5. 퍼셉트론 함수 구현: 에폭에 따른 판별식 시각화

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12
13 W = np.array([w]) # adding the initial weights
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y, w, eta=0.05, epochs=1)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annotate=True)
```



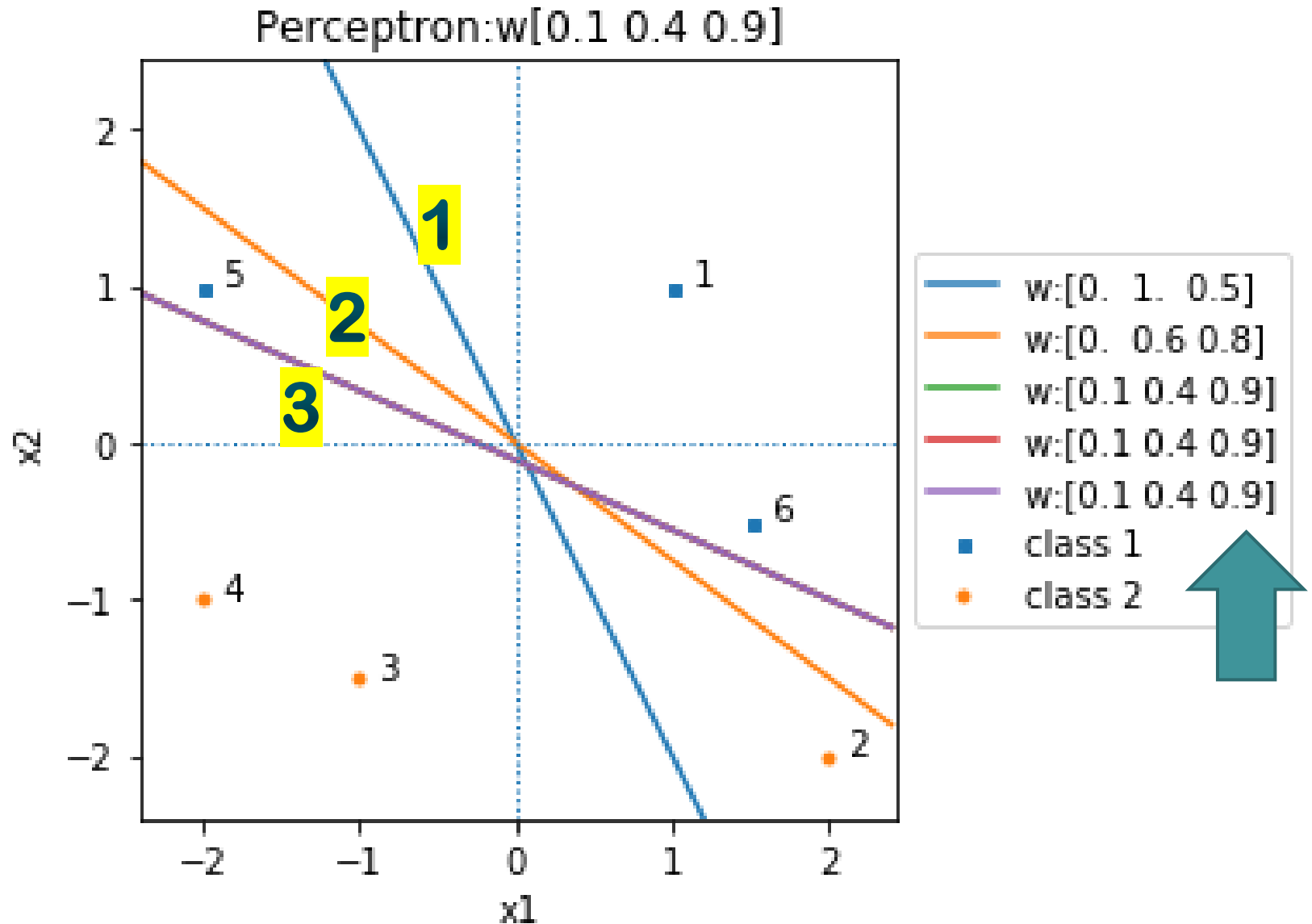
5. 퍼셉트론 함수 구현: 판별식 시각화 결과 분석

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0,
7               [-1.0, -1.0,
8               [-2.0, 1.0]
9 X = np.c_[ np.ones(len(x)), x]
10 y = np.array([1, -1, -1])
11 w = np.array([0, 1.0, 0])
12
13 W = np.array([w]) # a
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annot=True)
```



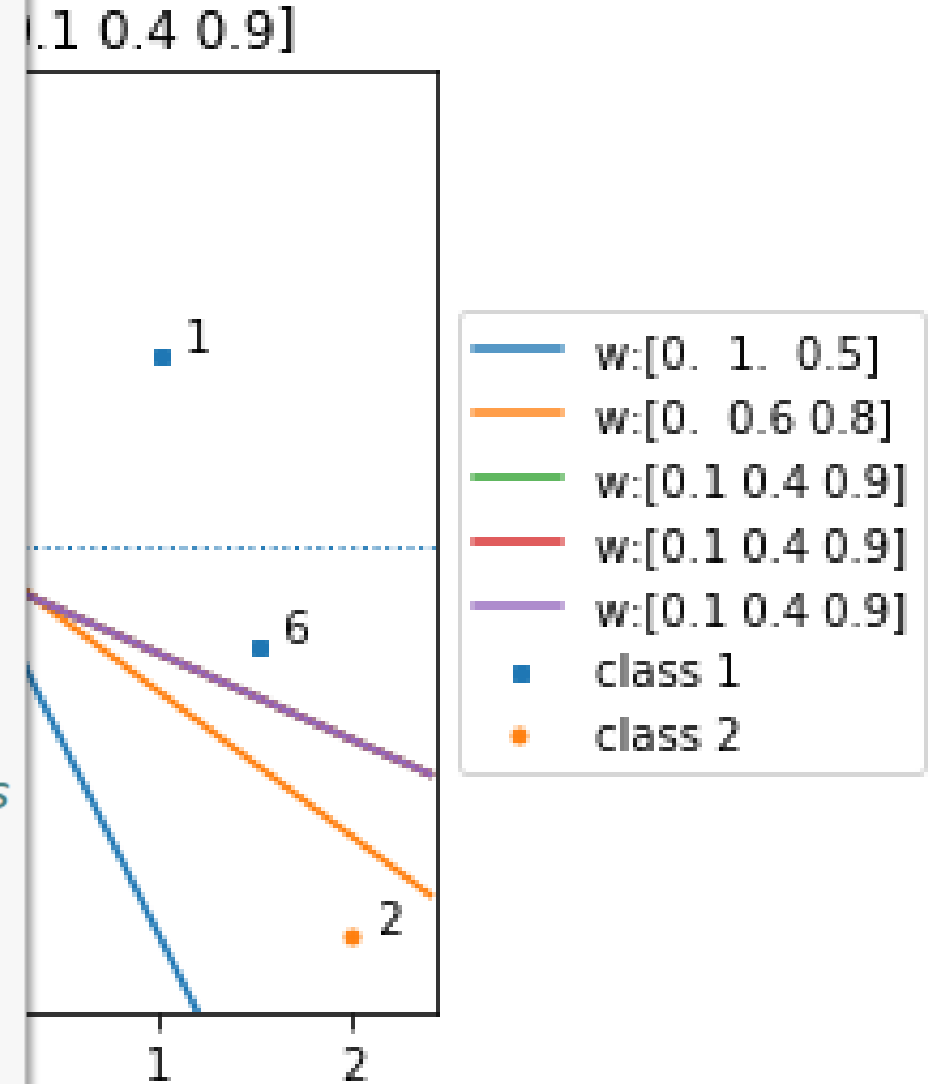
5. 퍼셉트론 함수 구현: 판별식 시각화 결과 분석

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0,
7               [-1.0, -1.0,
8               [-2.0, 1.0]
9 X = np.c_[ np.ones(len(x)), x]
10 y = np.array([1, -1, -1])
11 w = np.array([0, 1.0, 0])
12
13 W = np.array([w]) # a
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annot=True)
```



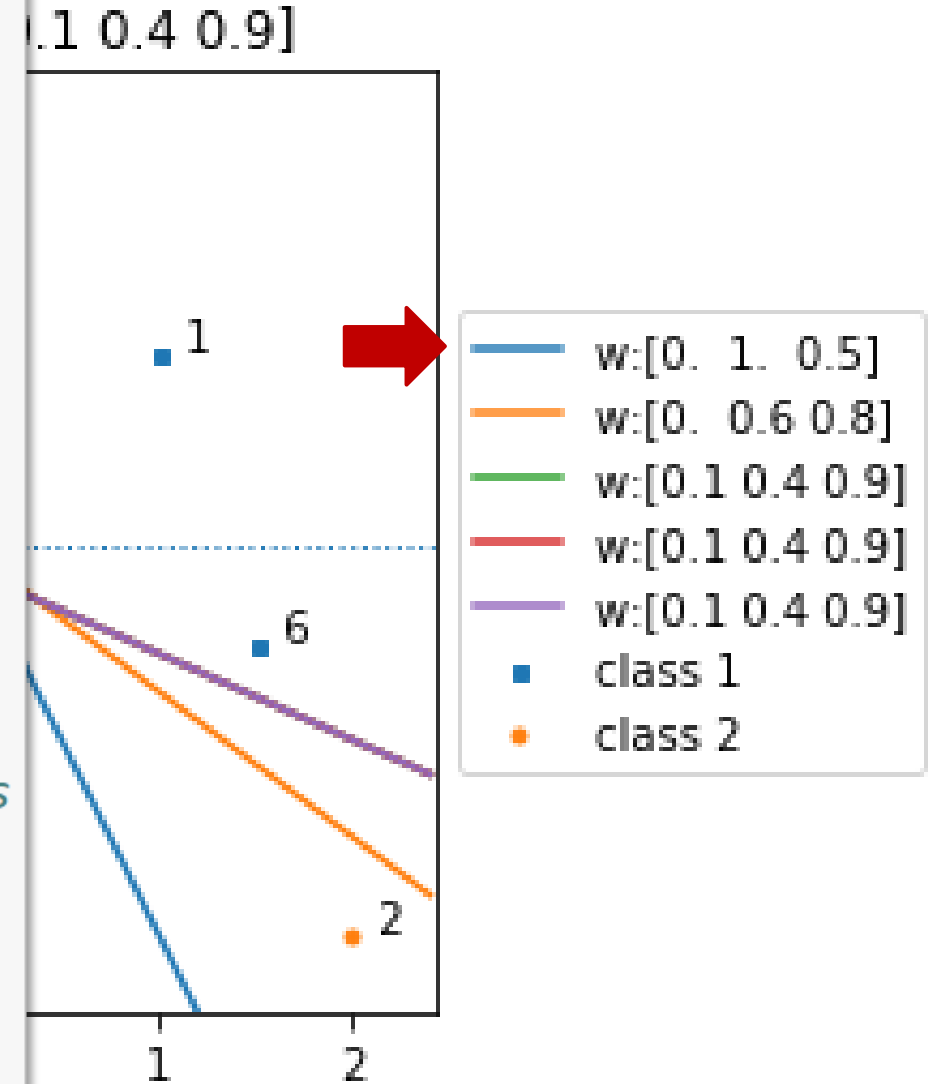
5. 퍼셉트론 함수 구현: 판별식 시각화 결과 분석

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12
13 W = np.array([w]) # adding the initial weights
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y, w, eta=0.05, epochs=1)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annotate=True)
```



5. 퍼셉트론 함수 구현: 판별식 시각화 결과 분석

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4 %run code/plot_xyw.py
5
6 x = np.array([[1.0, 1.0], [2.0, -2.0],
7               [-1.0, -1.5], [-2.0, -1.0],
8               [-2.0, 1.0], [1.5, -0.5]])
9 X = np.c_[ np.ones(len(x)), x ]
10 y = np.array([1, -1, -1, -1, 1, 1])
11 w = np.array([0, 1.0, 0.5])
12
13 W = np.array([w]) # adding the initial weights
14 epochs = 4
15 for _ in range(epochs):
16     w = perceptron(X, y, w, eta=0.05, epochs=1)
17     W = np.vstack([W, w])
18 plot_xyw(x, y, W, annotate=True)
```



퍼셉트론 코딩

- 학습 정리
 - 학습 자료의 준비
 - 학습 자료로 연산하기
 - 학습률과 에포크
 - 퍼셉트론 함수 구현
- 차시 예고
 - **5-1** 기계학습 작업 흐름도

4주차(3/3)

퍼셉트론 코딩

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

여러분 곁에 항상 열려 있는 K-MOOC 강의실에서 만나 뵙기를 바랍니다.