

6주차(2/3)

객체지향 퍼셉트론 활용

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

객체지향 퍼셉트론 활용

- 학습 목표
 - 객체지향 퍼셉트론 모델을 검증한다.
 - 객체지향 프로그래밍의 장점을 활용한다.
- 학습 내용
 - 객체지향 퍼셉트론의 장점 활용
 - **Joy DataSet** 으로 객체지향 퍼셉트론 검증

1. 객체지향 퍼셉트론 클래스 : 객체 만들기

- 속성: 인스턴스 변수
 - eta, epochs, random_seed
 - w, w_
 - cost_
 - miny, maxy
- 기능: 메소드
 - __init__()
 - fit()
 - net_input()
 - activate()
 - predict()



- 속성(데이터)
 - 차 길이 : 4m
 - 차 무게 : 1380kg
- 기능(함수)
 - 전진
 - 후진

2. 퍼셉트론 객체: 자료 읽어오기

- joy_data.txt

- data 폴더
- OS 코멘드



```
!head -5 data/joy_data.txt
```

1.72	3.12	1
0.31	1.85	1
1.56	2.85	1
2.64	2.41	1
1.23	2.54	1

```
!tail -5 data/joy_data.txt
```

-2.26	0.01	-1
-1.41	-0.23	-1
-1.20	-0.71	-1
-1.69	0.70	-1
-1.52	-1.14	-1

2. 퍼셉트론 객체: 자료 읽어오기

- joy_data.txt
 - import joy

```
import joy
X, y = joy.joy_data()
print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
print(X[:5], y[:5])
print(X[-5:], y[-5:])
```

2. 퍼셉트론 객체: 자료 읽어오기

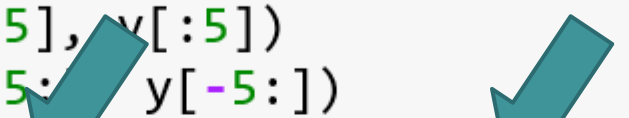
- joy_data.txt
 - import joy
 - joy.joy_data() 호출

```
import joy
X, y = joy.joy_data()
print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
print(X[:5], y[:5])
print(X[-5:], y[-5:])
```

2. 퍼셉트론 객체: 자료 읽어오기

- joy_data.txt
 - import joy
 - joy.joy_data() 호출
 - X 형상: (100, 2)
 - y 형상: (100,)

```
import joy
X, y = joy.joy_data()
print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
print(X[:5], y[:5])
print(X[-5:], y[-5:])
```



X.shape=(100, 2), y.shape=(100,)

```
[[1.72  3.12]
 [0.31  1.85]
 [1.56  2.85]
 [2.64  2.41]
 [1.23  2.54]] [1 1 1 1 1]
[[-2.26  0.01]
 [-1.41 -0.23]
 [-1.2   -0.71]
 [-1.69  0.7 ]
 [-1.52 -1.14]] [-1 -1 -1 -1 -1]
```

2. 퍼셉트론 객체: 자료 읽어오기

- joy_data.txt
 - import joy
 - joy.joy_data() 호출
 - X 형상: (100, 2)
 - y 형상: (100,)
 - 클래스 레이블: 1 혹은 -1

```
import joy
X, y = joy.joy_data()
print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
print(X[:5], y[:5])
print(X[-5:], y[-5:])
```

X.shape=(100, 2), y.shape=(100,)

[[1.72 3.12]

[0.31 1.85]

[1.56 2.85]

[2.64 2.41]

[1.23 2.54]]

[1 1 1 1 1]

[[-2.26 0.01]

[-1.41 -0.23]

[-1.2 -0.71]

[-1.69 0.7]

[-1.52 -1.14]]

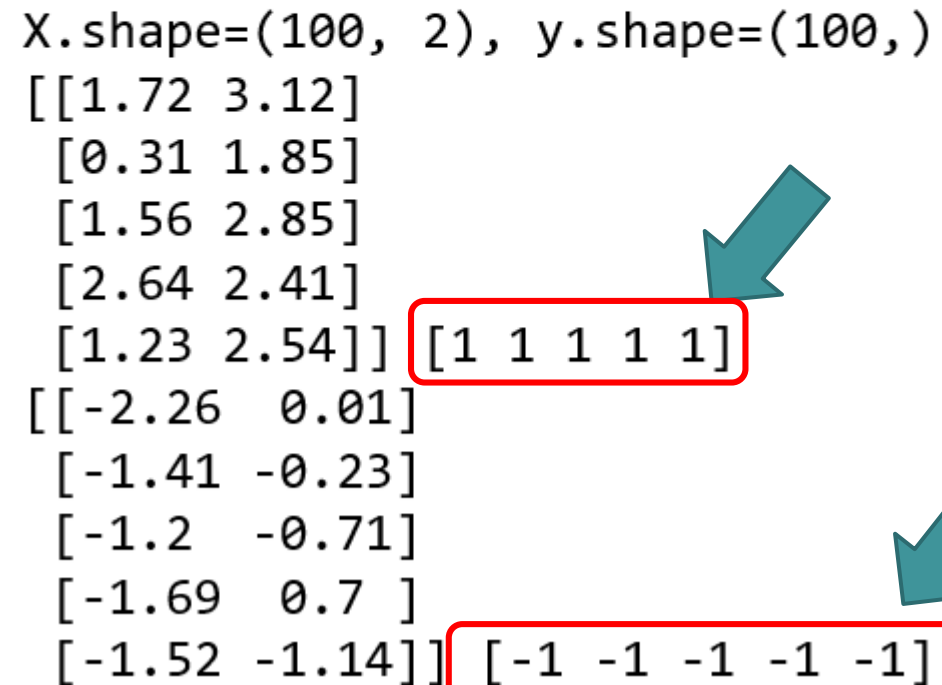
[-1 -1 -1 -1 -1]

2. 퍼셉트론 객체: 자료 읽어오기

- joy_data.txt
 - import joy
 - joy.joy_data() 호출
 - X 형상: (100, 2)
 - y 형상: (100,)
 - 클래스 레이블: 1 혹은 -1
 - Shuffling 전처리 필요성

```
import joy
X, y = joy.joy_data()
print('X.shape={}, y.shape={}'.format(X.shape, y.shape))
print(X[:5], y[:5])
print(X[-5:], y[-5:])
```

```
X.shape=(100, 2), y.shape=(100,)
[[1.72  3.12]
 [0.31  1.85]
 [1.56  2.85]
 [2.64  2.41]
 [1.23  2.54]] [1 1 1 1 1]
[[-2.26  0.01]
 [-1.41 -0.23]
 [-1.2   -0.71]
 [-1.69  0.7 ]
 [-1.52 -1.14]] [-1 -1 -1 -1 -1]
```



2. 퍼셉트론 객체: 자료 전처리

- `joy_data.txt`
 - `import joy`
 - `joy.joy_data()` 호출
 - `X` 형상: (100, 2)
 - `y` 형상: (100,)
 - 클래스 레이블: 1 혹은 -1
 - **Shuffling** 전처리 필요성

```
X, y = joy.joy_data()  
print(X[:5], y[:5])
```

```
[[1.72 3.12]  
 [0.31 1.85]  
 [1.56 2.85]  
 [2.64 2.41]  
 [1.23 2.54]] [1 1 1 1 1]
```



```
X, y = joy.joy_data(standardized=True  
                    shuffled=True)  
print(X[:5], y[:5])
```



```
[[ -1.28409207 -1.15278304]  
 [-1.43790346 -0.83714568]  
 [ 0.41249418  0.10034442]  
 [-1.37731109 -1.63330561]  
 [-0.0769057  -1.03500791]] [-1 -1 1 -1 -1]
```

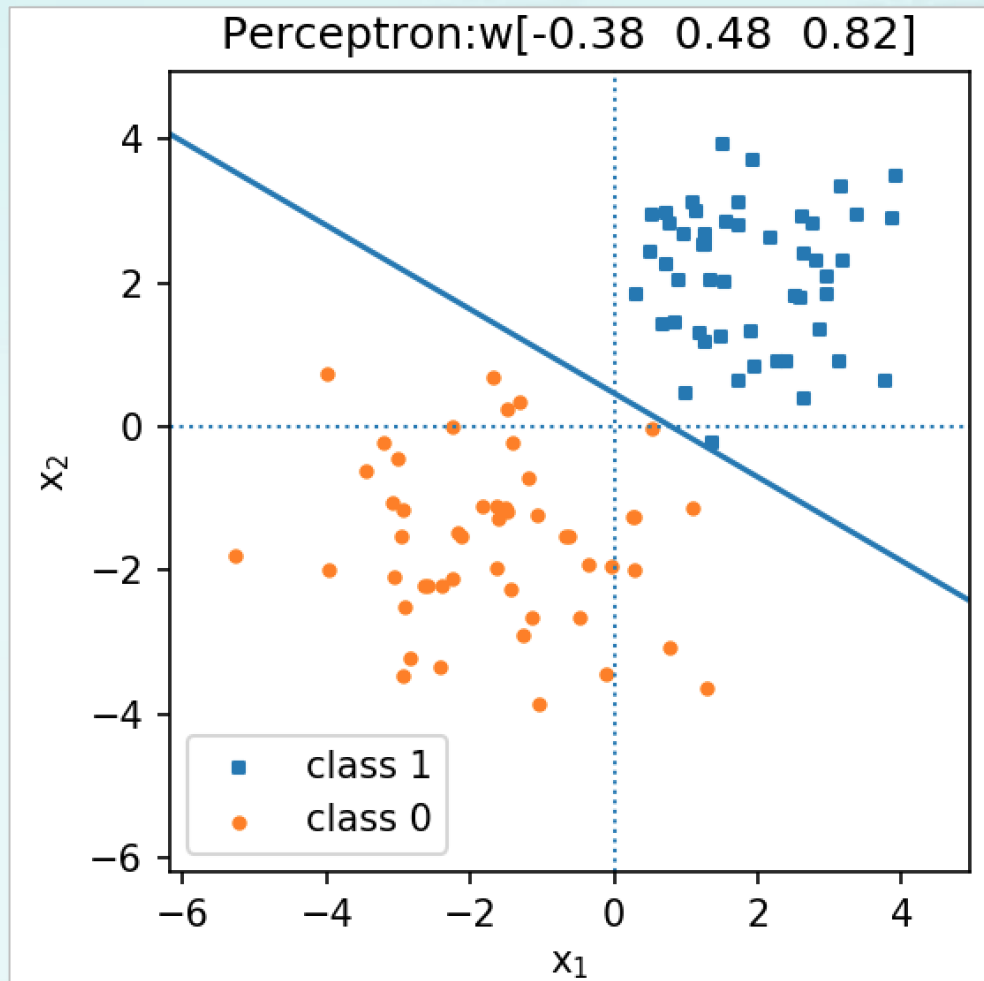
2. 퍼셉트론 객체: 활용 코드

1. 학습자료 읽어오기
2. 퍼셉트론 객체 생성
3. 퍼셉트론 학습
4. 시각화를 통한 자료 분류

```
1 import joy
2 X, y = joy.joy_data()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

3. 객체지향 퍼셉트론 활용: OOP 장점

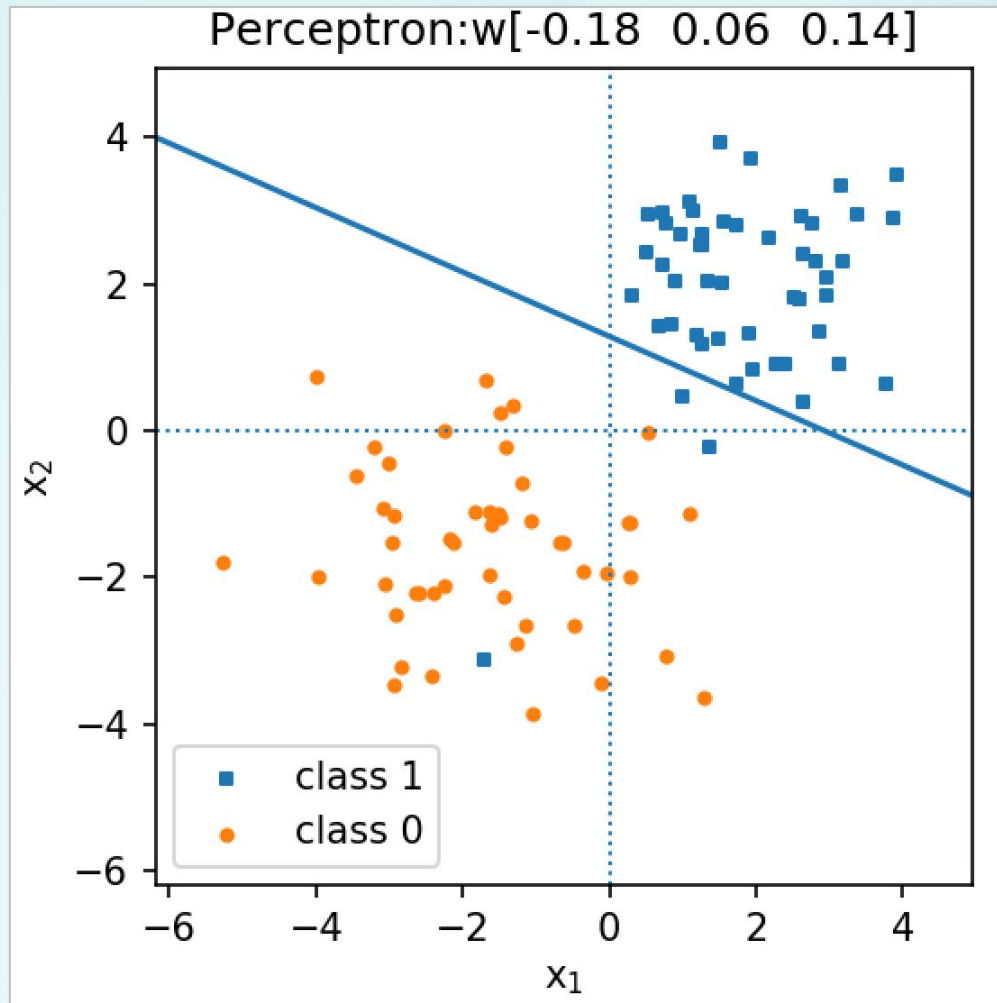
- joy_data.txt



```
1 import joy
2 X, y = joy.joy_data()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

3. 객체지향 퍼셉트론 활용: OOP 장점

- joy_Ndata.txt

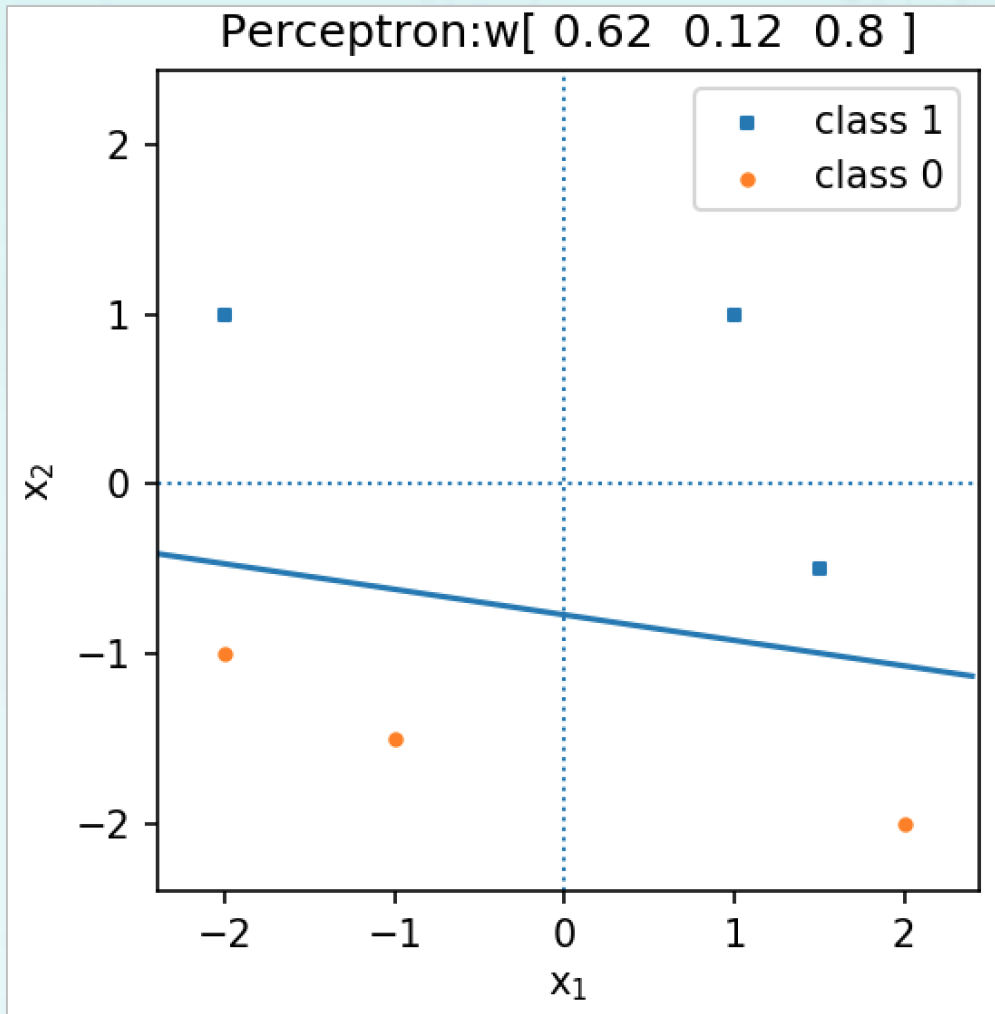


```
1 import joy
2 X, y = joy.joy_data()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

```
1 import joy
2 X, y = joy.joy_Ndata()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

3. 객체지향 퍼셉트론 활용: OOP 장점

- toy_data.txt



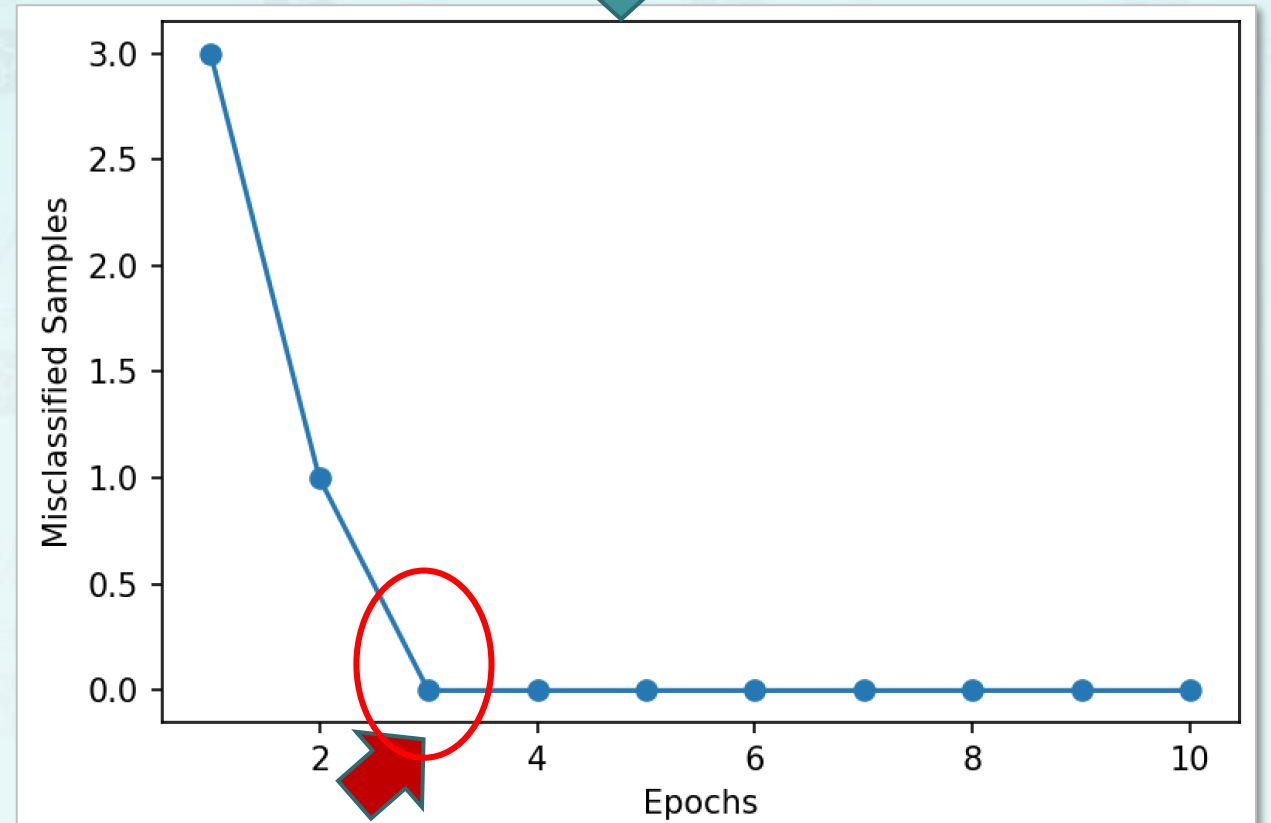
```
1 import joy
2 X, y = joy.joy_data()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

```
1 import joy
2 X, y = joy.joy_Ndata()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

```
1 import joy
2 X, y = joy.toy_data()
3 ppn = Perceptron(eta = 0.1, epochs=10)
4 ppn.fit(X, y)
5 joy.plot_xyw(X, y, ppn.w)
```

3. 객체지향 퍼셉트론 활용: 반복학습과 오류

```
plt.plot(range(1, len(ppn.cost_)+1), ppn.cost_,  
         marker='o')  
plt.xlabel('Epochs')  
plt.ylabel('Misclassified Samples')  
plt.savefig('Epochs', dpi=150)
```



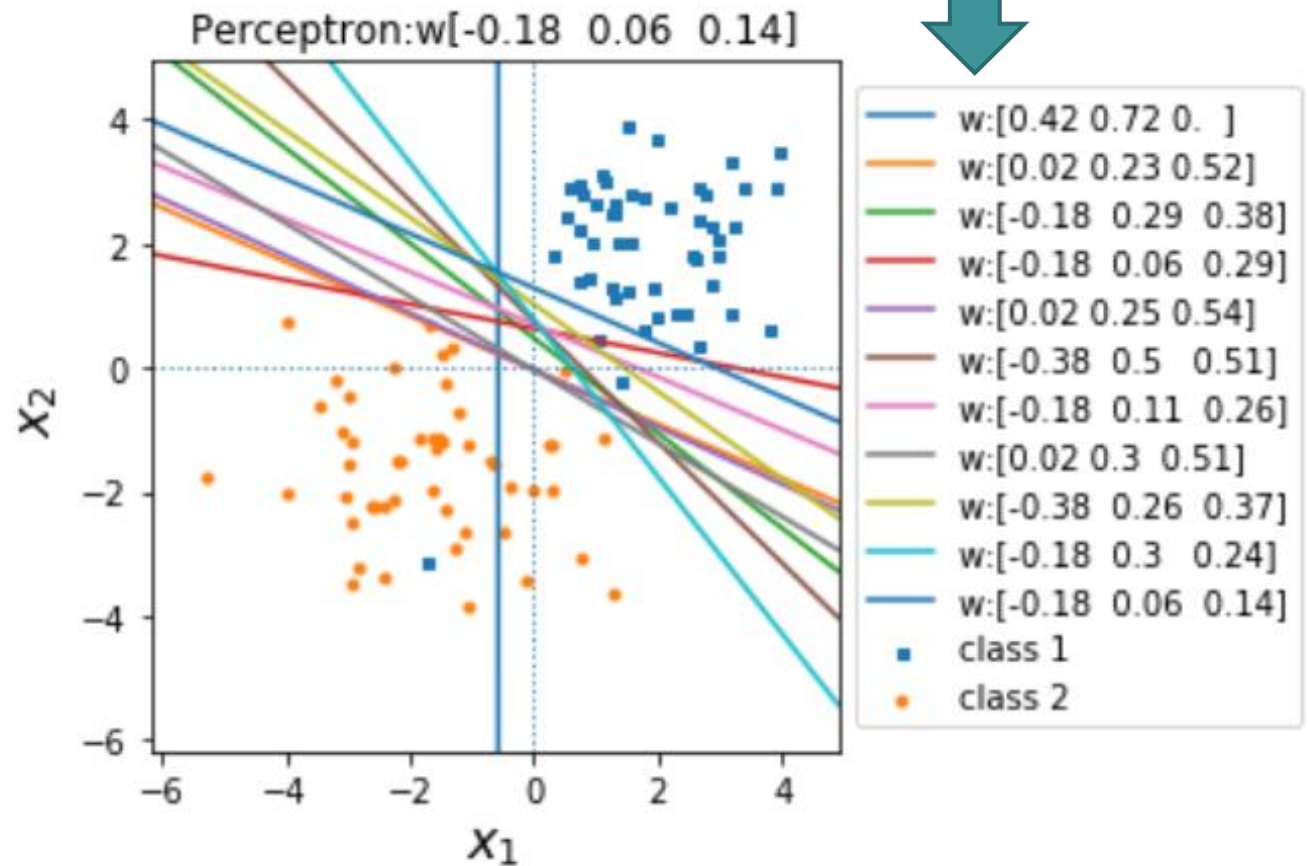
3. 객체지향 퍼셉트론 활용: 가중치 변화 추적

fit() method:

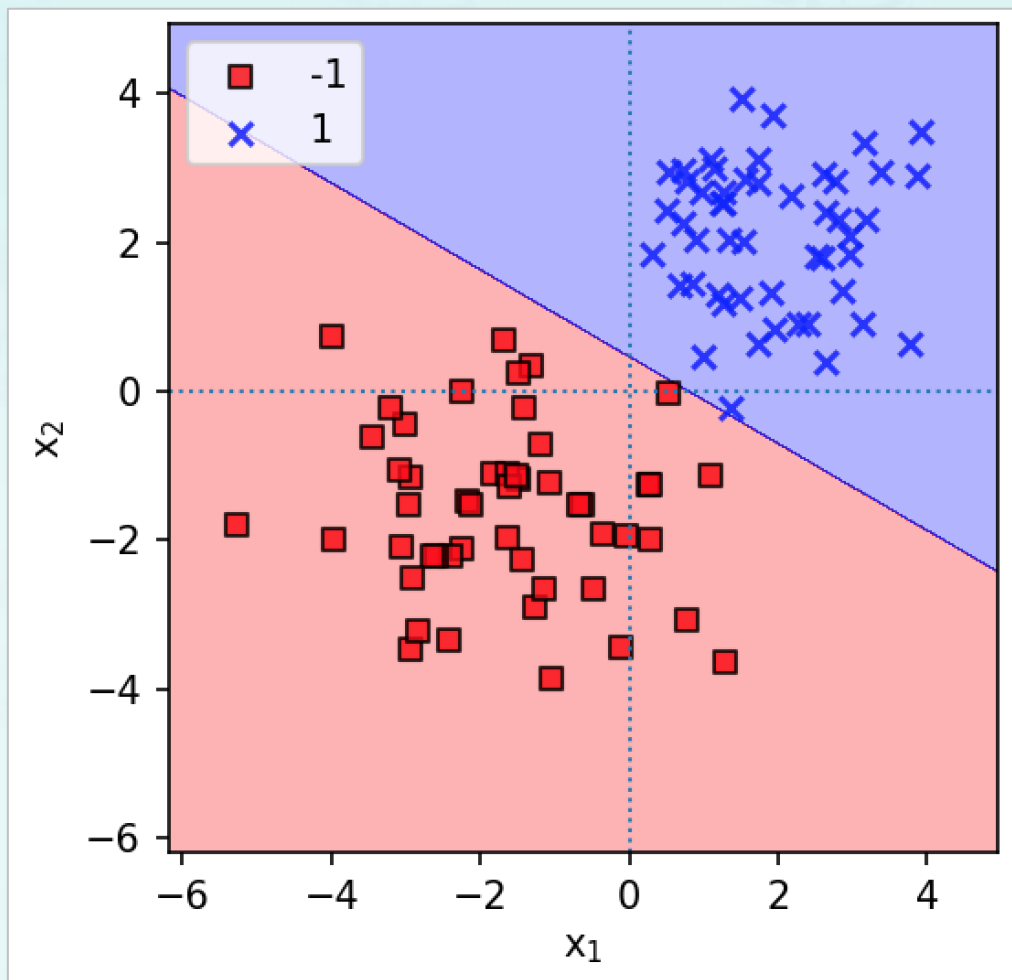
```
self.w = self.w + delta
errors += 1
self.cost_.append(errors)
self.w_ = np.vstack([self.w_, self.w])
return self
```


3. 객체지향 퍼셉트론 활용: 가중치 변화 추적

```
1 X, y = joy.getXY('data/joy_dataNoise.txt')
2 ppn = Perceptron(eta = 0.1, epochs=10)
3 ppn.fit(X, y)
4 plot_xyw(X, y, ppn.w_, savefig='joy_dataNoise')
```



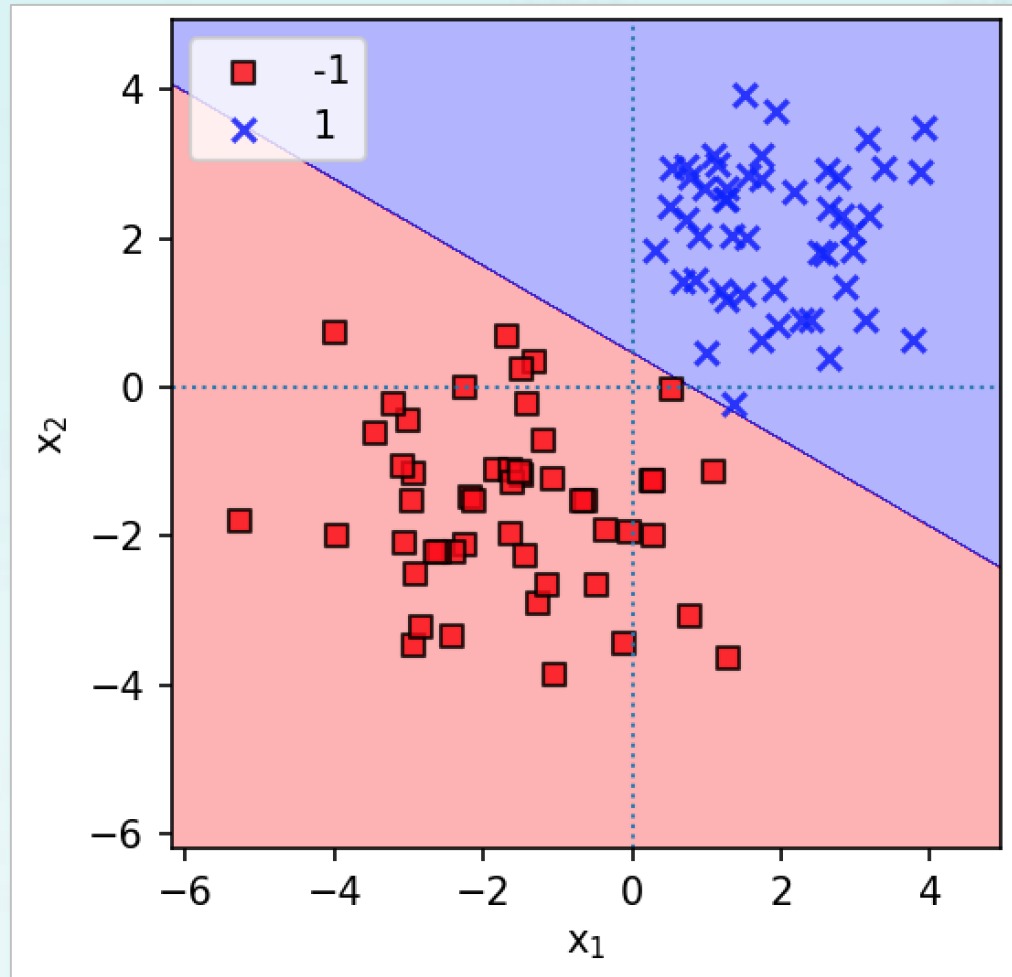
4. plot_decision_regions() : 코드와 시각화 결과



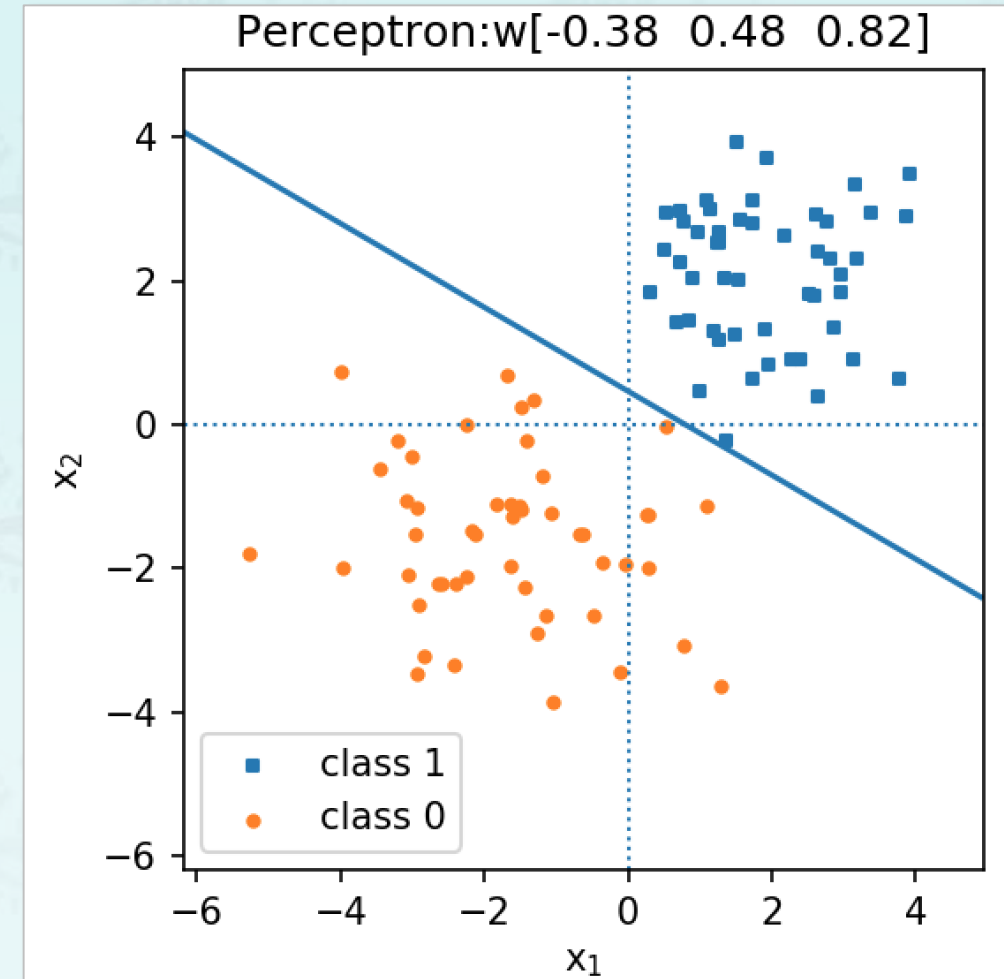
```
import joy
X, y = joy.joy_data()
ppn = Perceptron(eta = 0.1, epochs=10)
ppn.fit(X, y)
joy.plot_decision_regions(X, y, ppn)
```



4. plot_decision_regions() : 코드와 시각화 결과

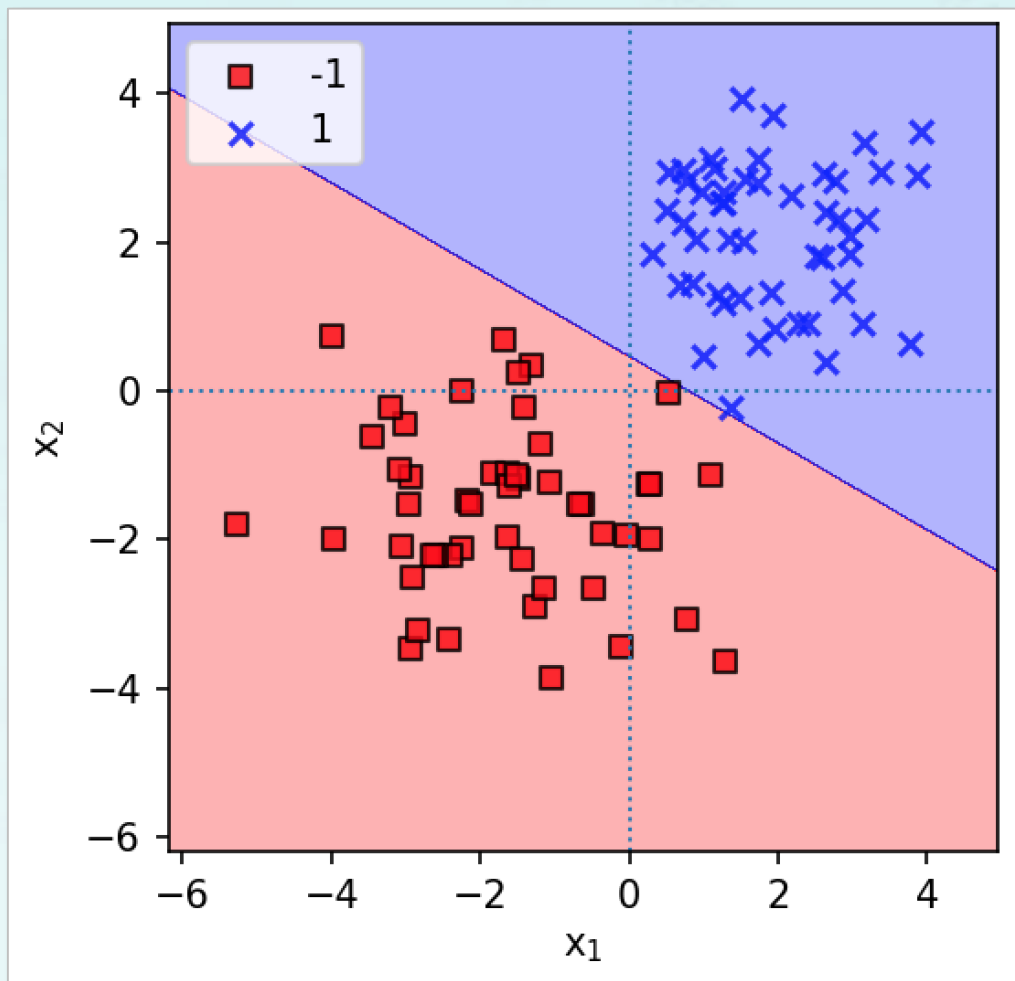


plot_decision_regions()



plot_xyw()

4. plot_decision_regions() : 코드와 시각화 결과




```
import joy
X, y = joy.joy_data()
ppn = Perceptron(eta = 0.1, epochs=10)
ppn.fit(X, y)
joy.plot_decision_regions(X, y, ppn)
```




5. 파이썬과 객체: first class objects

- 파이썬에는 모든 것이 객체
- 객체는 매개변수로 사용 가능, 저장 가능, 반환 가능
- 이런 객체들을 **first class objects** 라 부름

```
import joy
X, y = joy.joy_data()
ppn = Perceptron(eta = 0.1, epochs=10)
ppn.fit(X, y)
joy.plot_decision_regions(X, y, ppn)
```



```
import joy
X, y = joy.joy_data()
ppn = Perceptron(eta = 0.1, epochs=10)
ppn.fit(X, y)
joy.plot_xyw(X, y, ppn.w)
```



5. 파이썬과 객체: first class objects

- 매개변수로 퍼셉트론 객체 전달
- 전달 후 함수 안에서 전달 된 객체의 모든 인스턴스 변수와 메소드 사용 가능

```
import joy
X, y = joy.joy_data()
ppn = Perceptron(eta = 0.1, epochs=10)
ppn.fit(X, y)
joy.plot_decision_regions(X , y, ppn)
```



객체지향 퍼셉트론 활용

- 학습 정리
 - 객체지향 퍼셉트론으로 **Joy DataSet** 적용하기
 - **OOP**의 장점을 활용한 프로그래밍 기법