

6주차(1/3)

객체지향 퍼셉트론 구현

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수

객체지향 퍼셉트론 구현

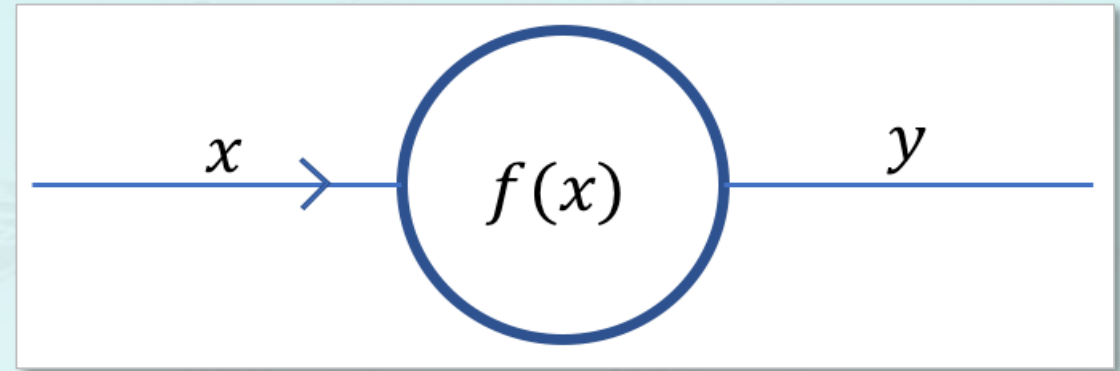
- 학습 목표
 - 객체지향 퍼셉트론을 구현한다.
 - 객체지향 프로그래밍의 장점을 잘 활용한다.
- 학습 내용
 - 퍼셉트론 클래스 설계
 - 객체지향 퍼셉트론 구현하기
 - 객체지향 프로그래밍 기법 활용하기

1. 함수의 변화

- 객체지향 퍼셉트론

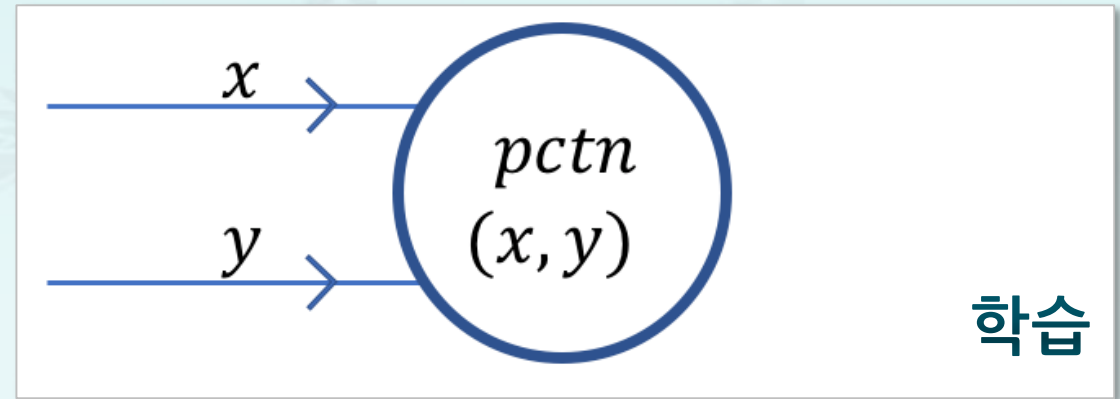
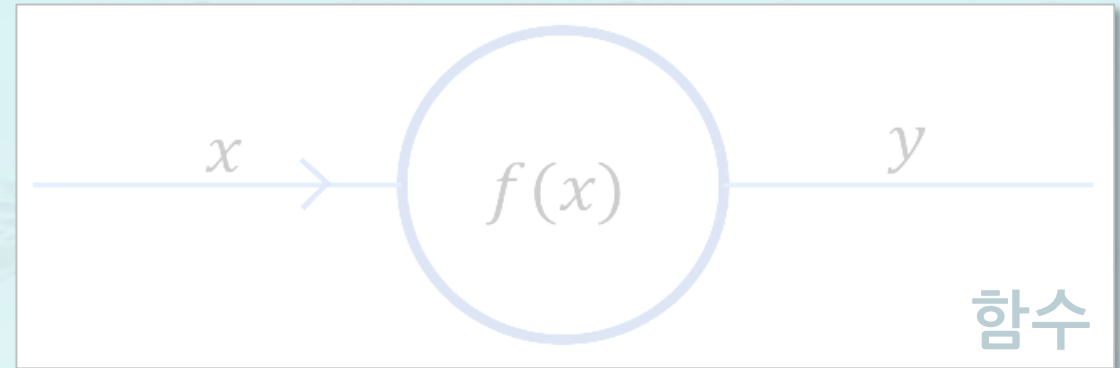
1. 함수의 변화: 고전적 함수

- 함수 $f(x)$
 - 프로그래머가 함수를 코딩한대로 출력한다



1. 함수의 변화: 기계학습에서의 함수(학습)

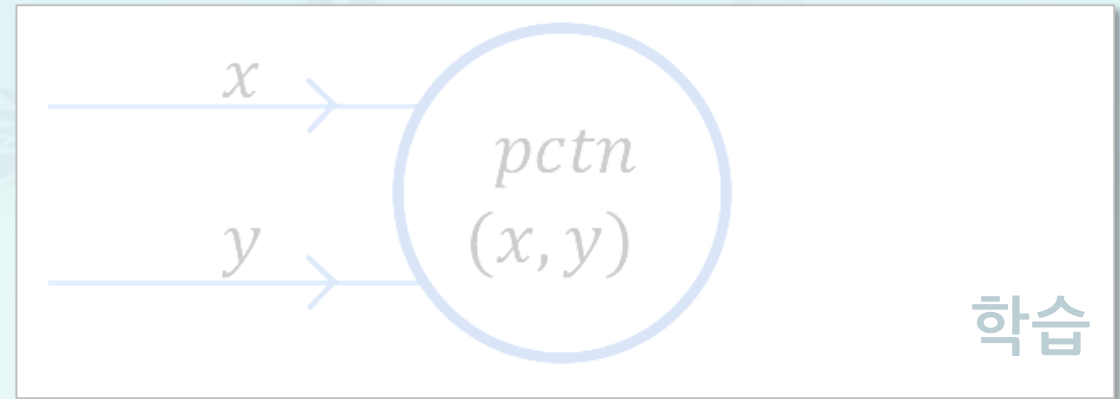
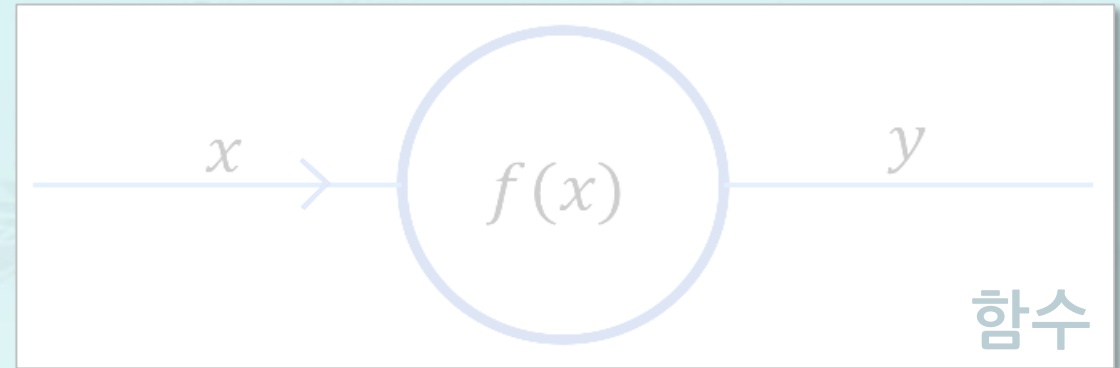
- 함수 $f(x)$
 - 프로그래머가 함수를 코딩한대로 출력한다
- 기계학습(함수) : $pctn(x, y)$
 - 학습 단계
 - 기계가 입력과 출력으로 학습한다



예측

1. 함수의 변화: 기계학습에서의 함수(예측)

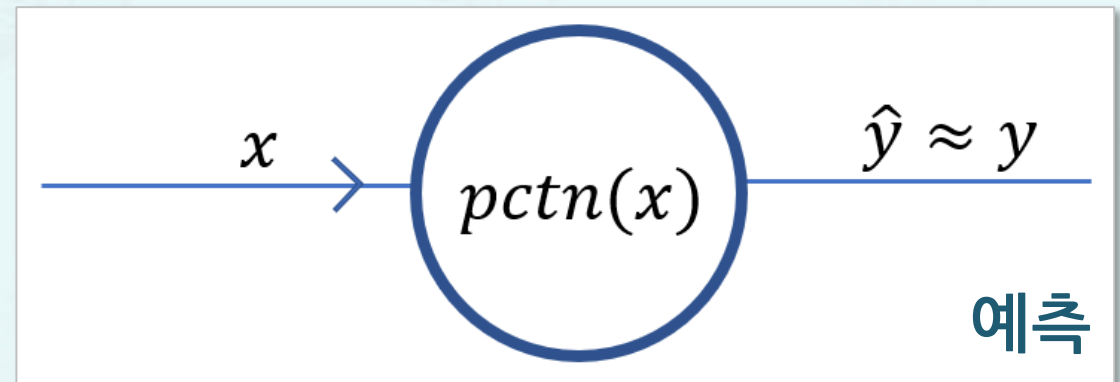
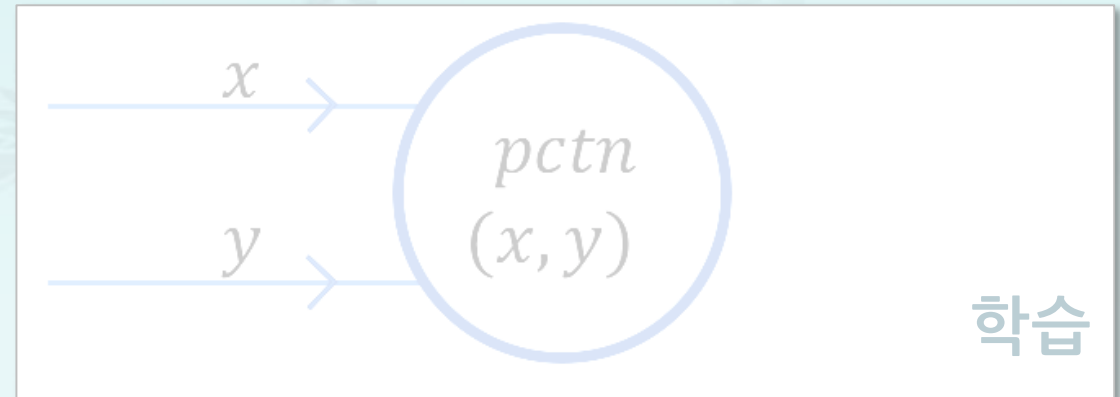
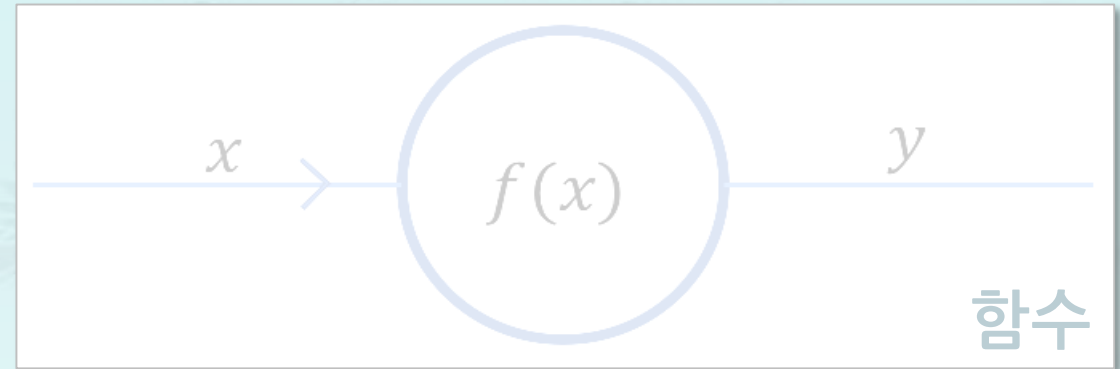
- 함수 $f(x)$
 - 프로그래머가 함수를 코딩한대로 출력한다
- 기계학습(함수) : $pctn(x, y)$
 - 학습 단계
 - 기계가 입력과 출력으로 학습한다
- 기계학습(함수) : $pctn(x, y)$
 - 예측 단계:



예측

1. 함수의 변화: 기계학습에서의 함수(예측)

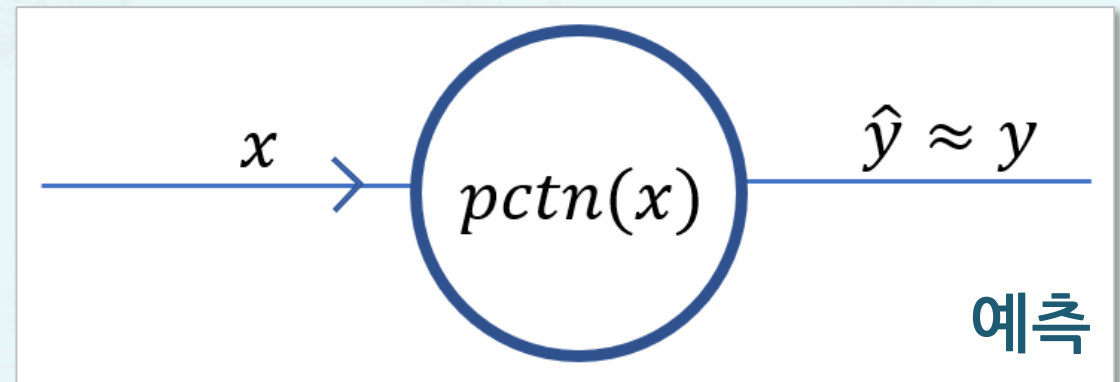
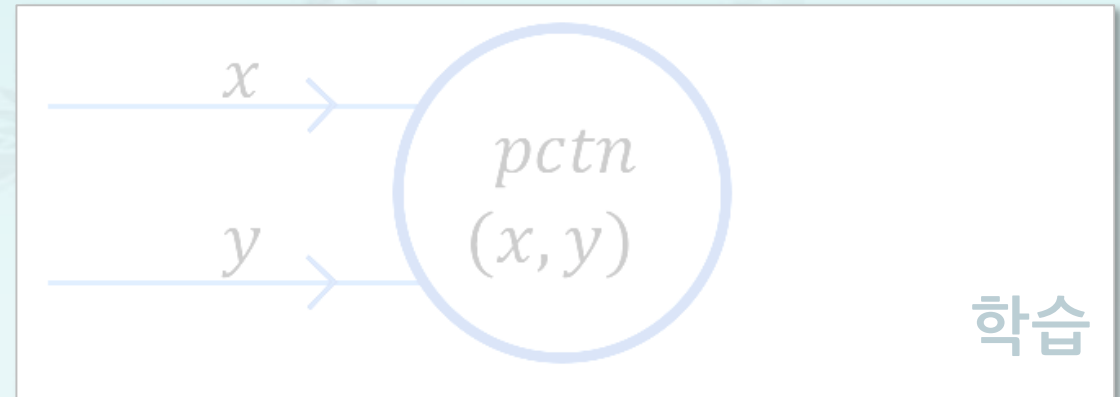
- 함수 $f(x)$
 - 프로그래머가 함수를 코딩한대로 출력한다
- 기계학습(함수) : $pctn(x, y)$
 - 학습 단계
 - 기계가 입력과 출력으로 학습한다
- 기계학습(함수) : $pctn(x, y)$
 - 예측 단계:
 - 기계가 학습한대로 예측한다



1.함수의 변화: 기계학습에서의 함수(객체)

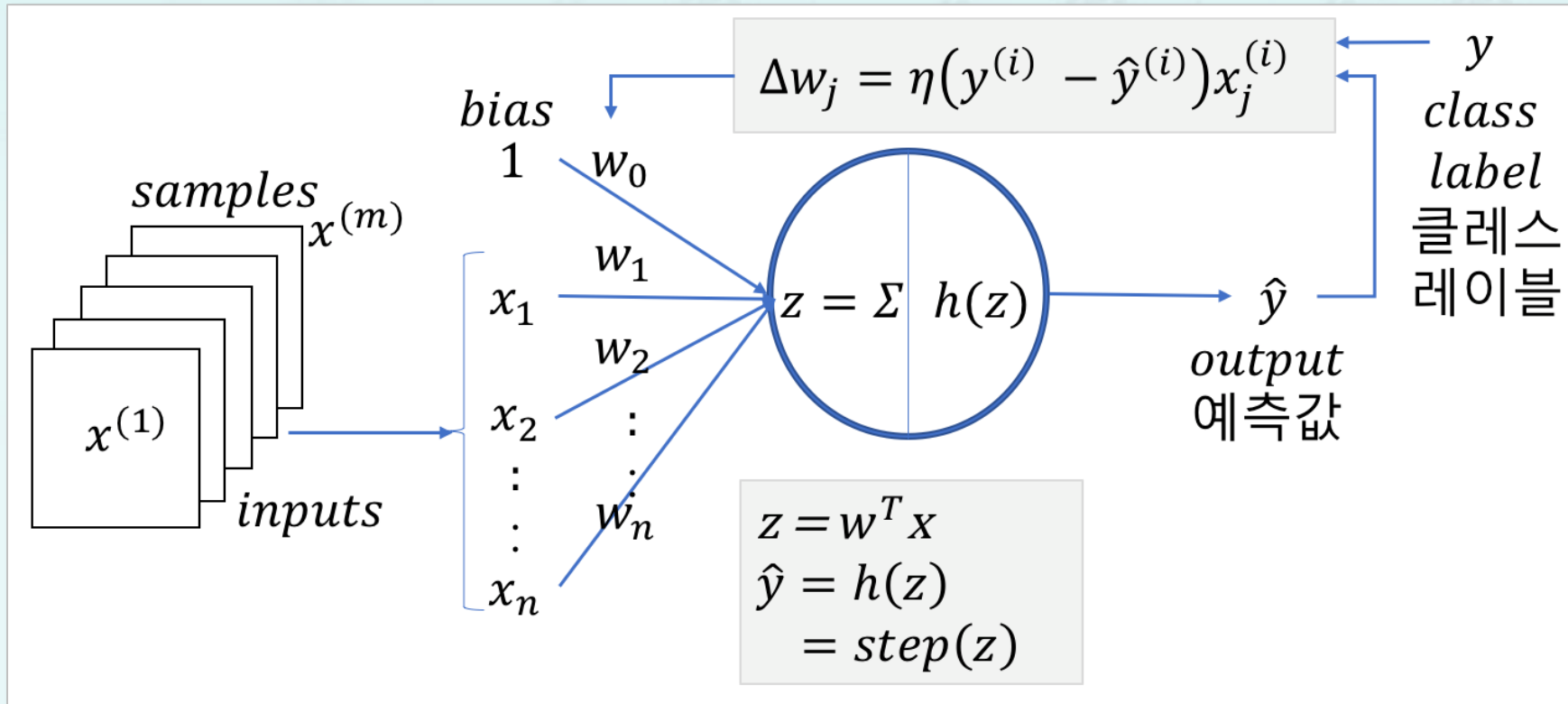
- 기계학습 : 객체 $pctn$

-



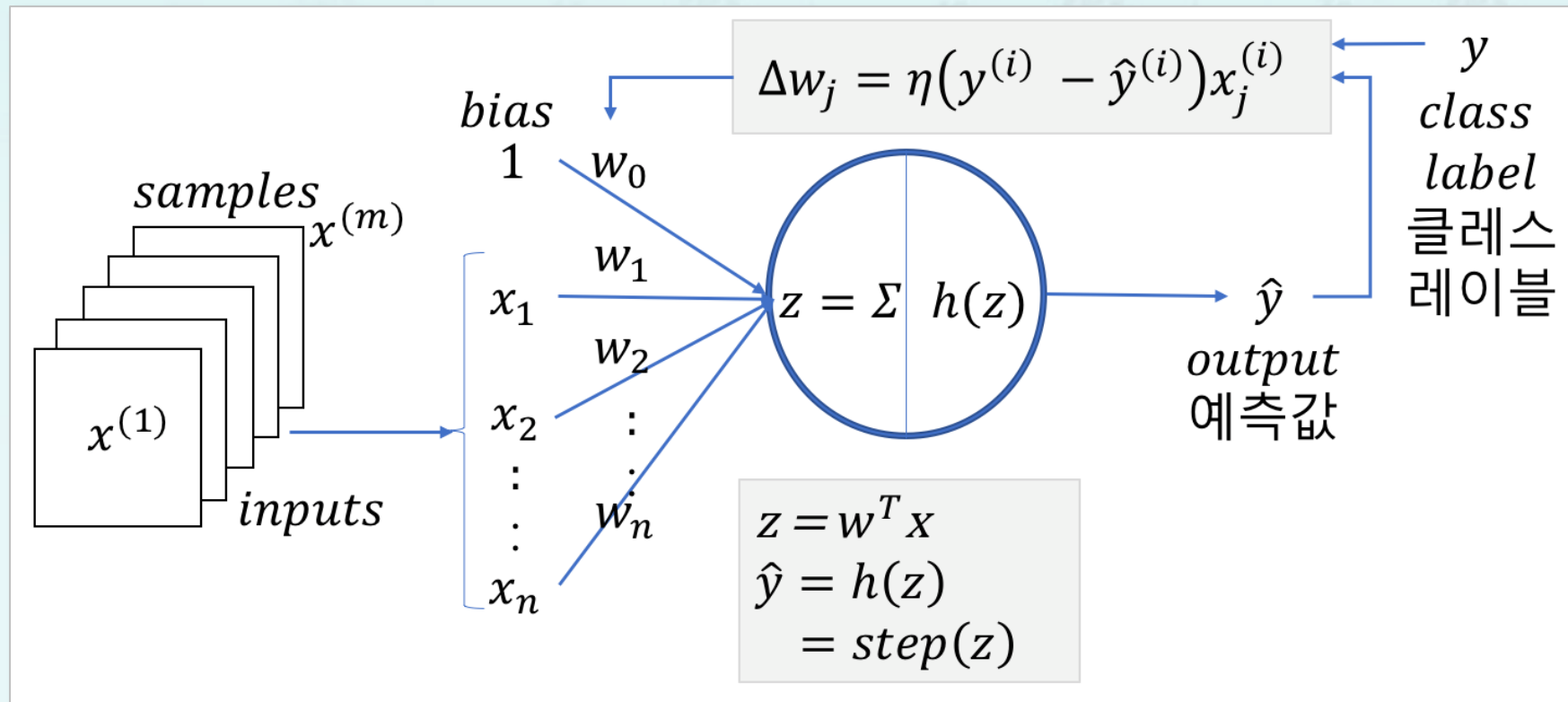
2. 객체지향 퍼셉트론

- 객체
 - 객체의 속성(인스턴스 변수)



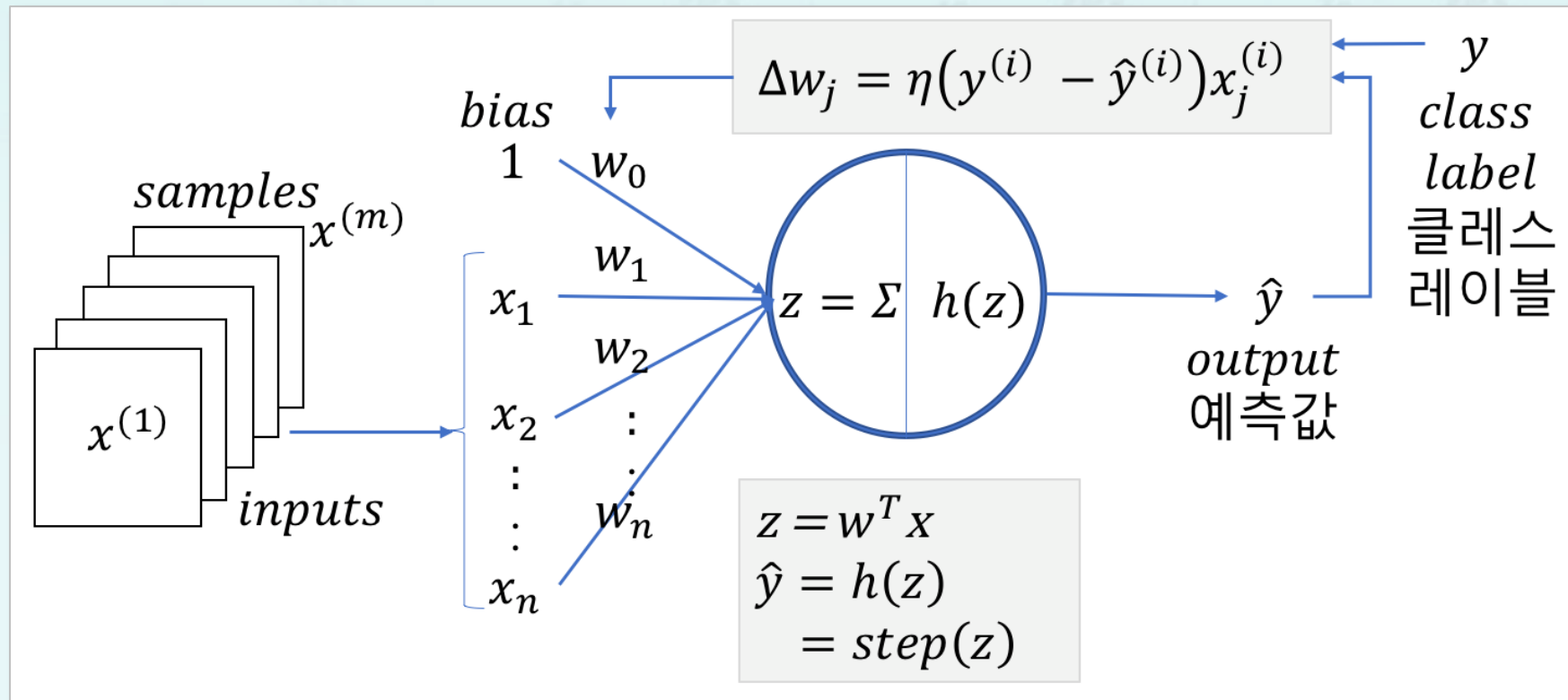
2. 객체지향 퍼셉트론: 속성

- 객체
 - 객체의 속성(인스턴스 변수)



2. 객체지향 퍼셉트론: 속성

- 객체
 - 객체의 속성(인스턴스 변수)

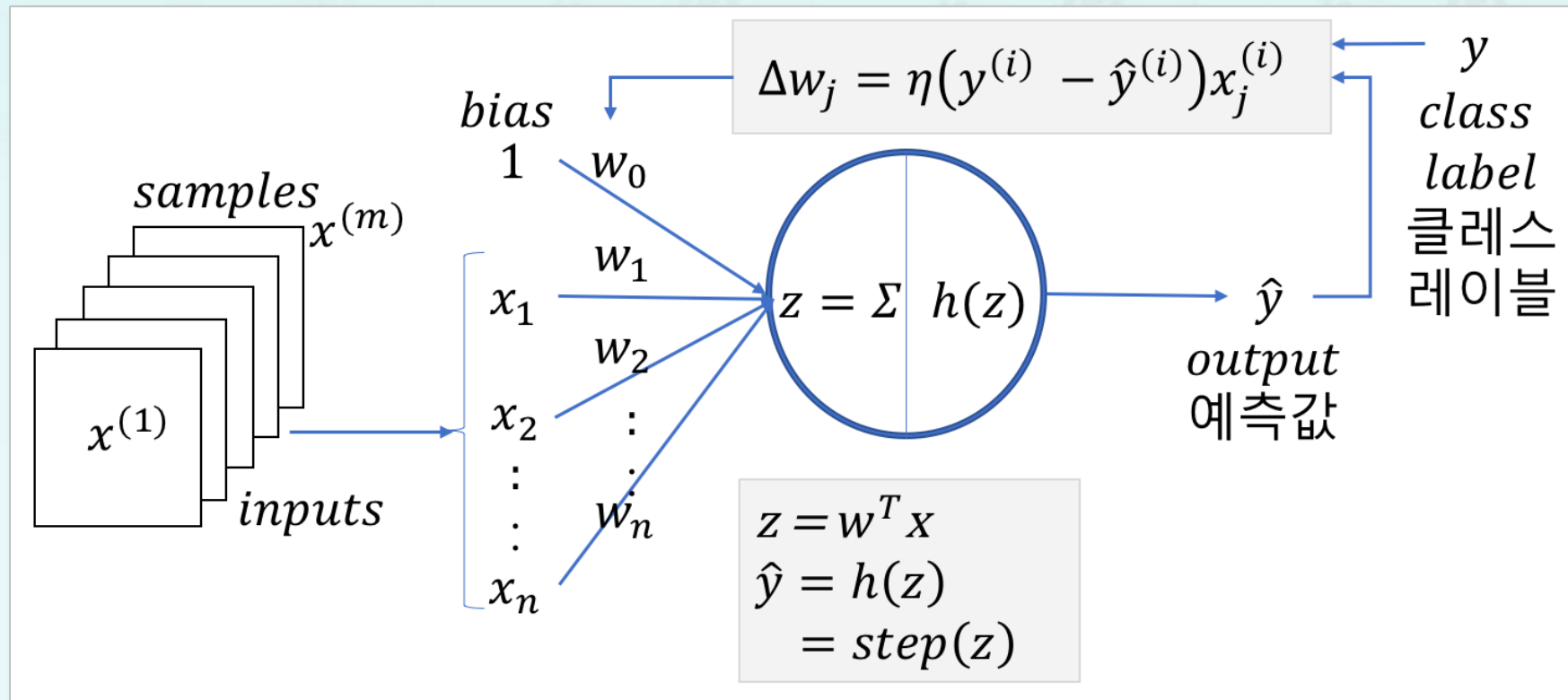


[속성:데이터]

1. 입력 (x)
2. 출력 (y)
3. 순입력 (z)
4. 레이블 (yhat)
5. 가중치 (w)
6. 학습률 (eta)
7. 반복횟수 (epochs)
8. 랜덤시드 (random_seed)

2. 객체지향 퍼셉트론: 속성

- 객체
 - 객체의 속성(인스턴스 변수)

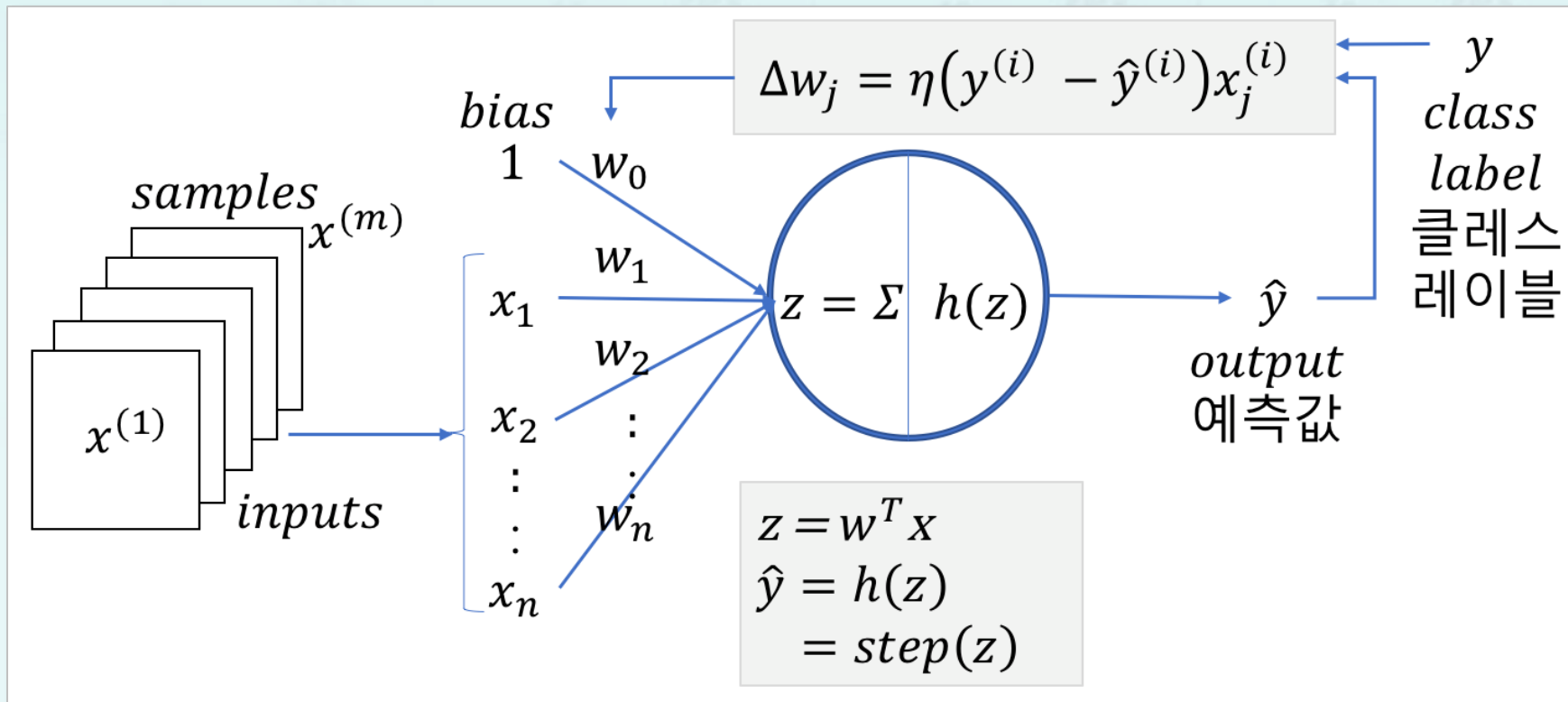


[속성:데이터]

1. 입력 (x)
2. 출력 (y)
3. 순입력 (z)
4. 레이블 (yhat)
5. 가중치 (w)
6. 학습률 (eta)
7. 반복횟수 (epochs)
8. 랜덤시드 (random_seed)

2. 객체지향 퍼셉트론: 기능

- 객체
 - 객체의 속성(인스턴스 변수)
 - 객체가 기능(메소드) ←

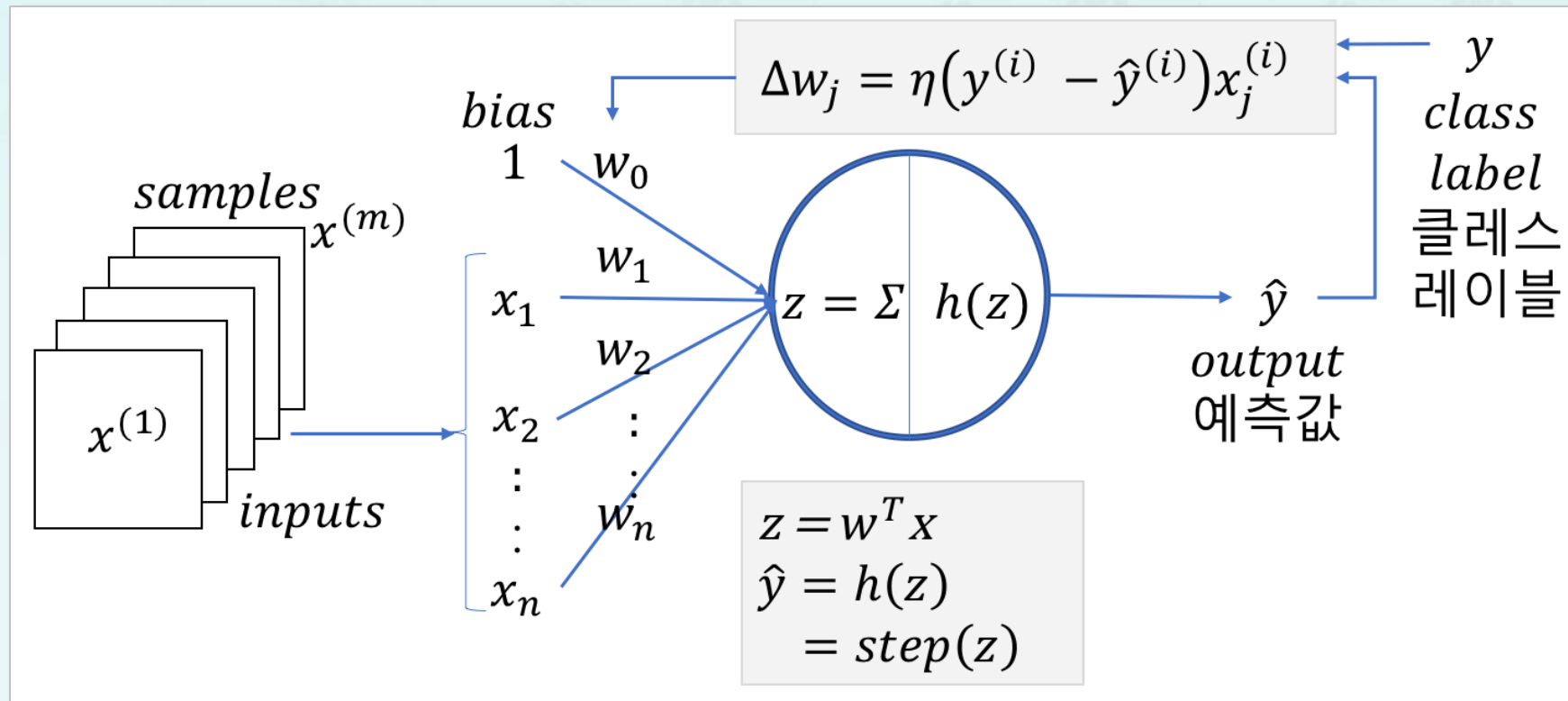


[기능:메소드] ←

- 1.
- 2.
- 3.
- 4.

2. 객체지향 퍼셉트론: 기능

- 객체
 - 객체의 속성(인스턴스 변수)
 - 객체가 기능(메소드)



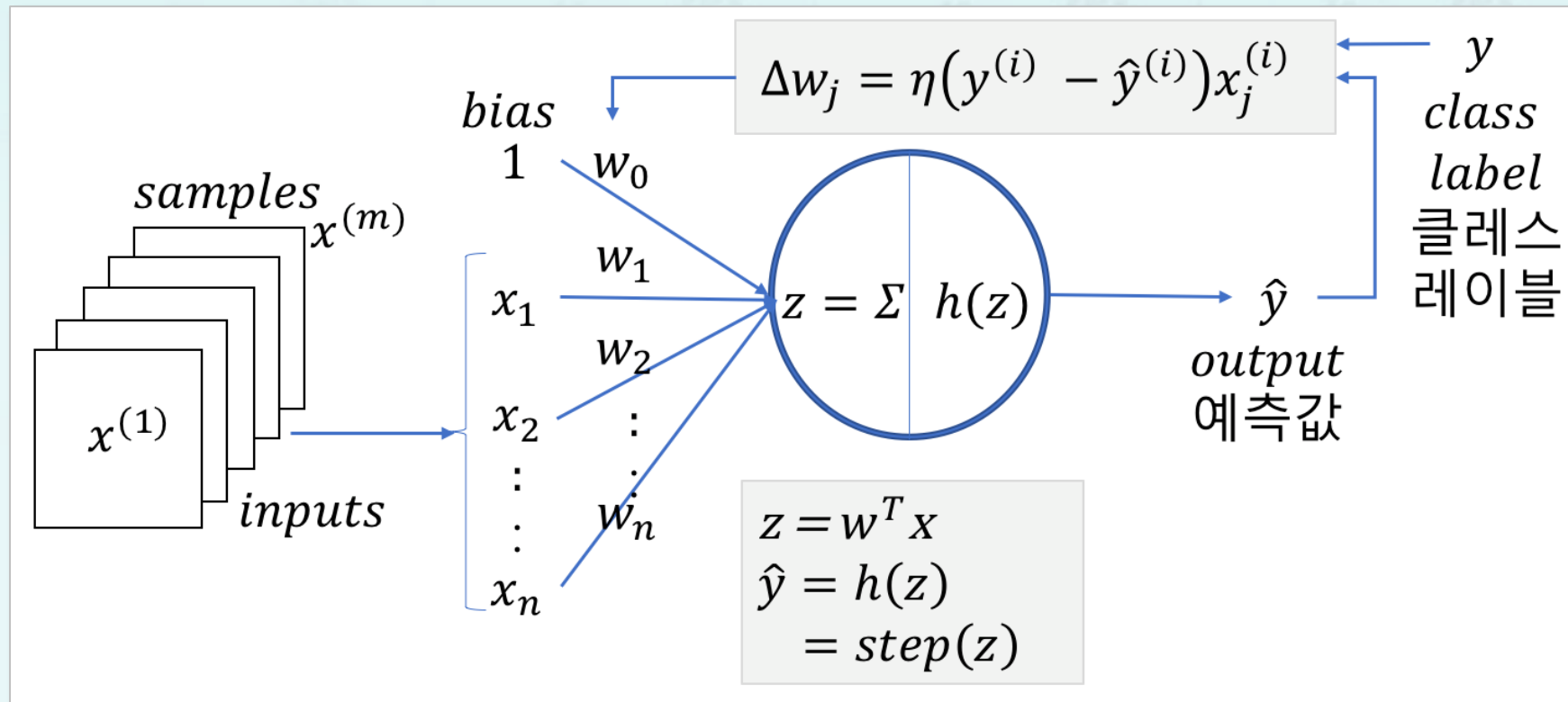
[기능:메소드]

1. 학습 (fit)
- 2.
- 3.
- 4.



2. 객체지향 퍼셉트론: 기능

- 객체
 - 객체의 속성(인스턴스 변수)
 - 객체가 기능(메소드)

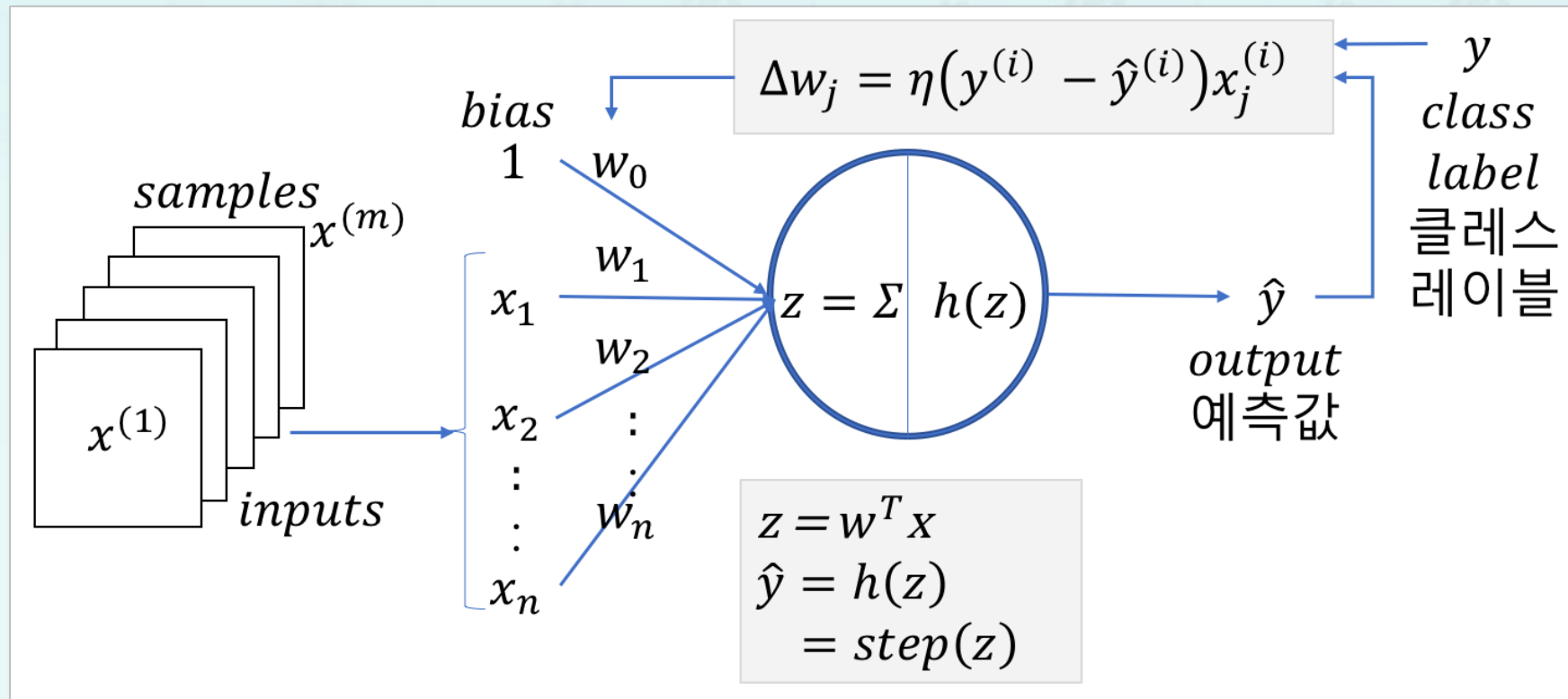


[기능:메소드]

1. 학습 (fit)
2. 순입력 (net_input)
- 3.
- 4.

2. 객체지향 퍼셉트론: 기능

- 객체
 - 객체의 속성(인스턴스 변수)
 - 객체가 기능(메소드)

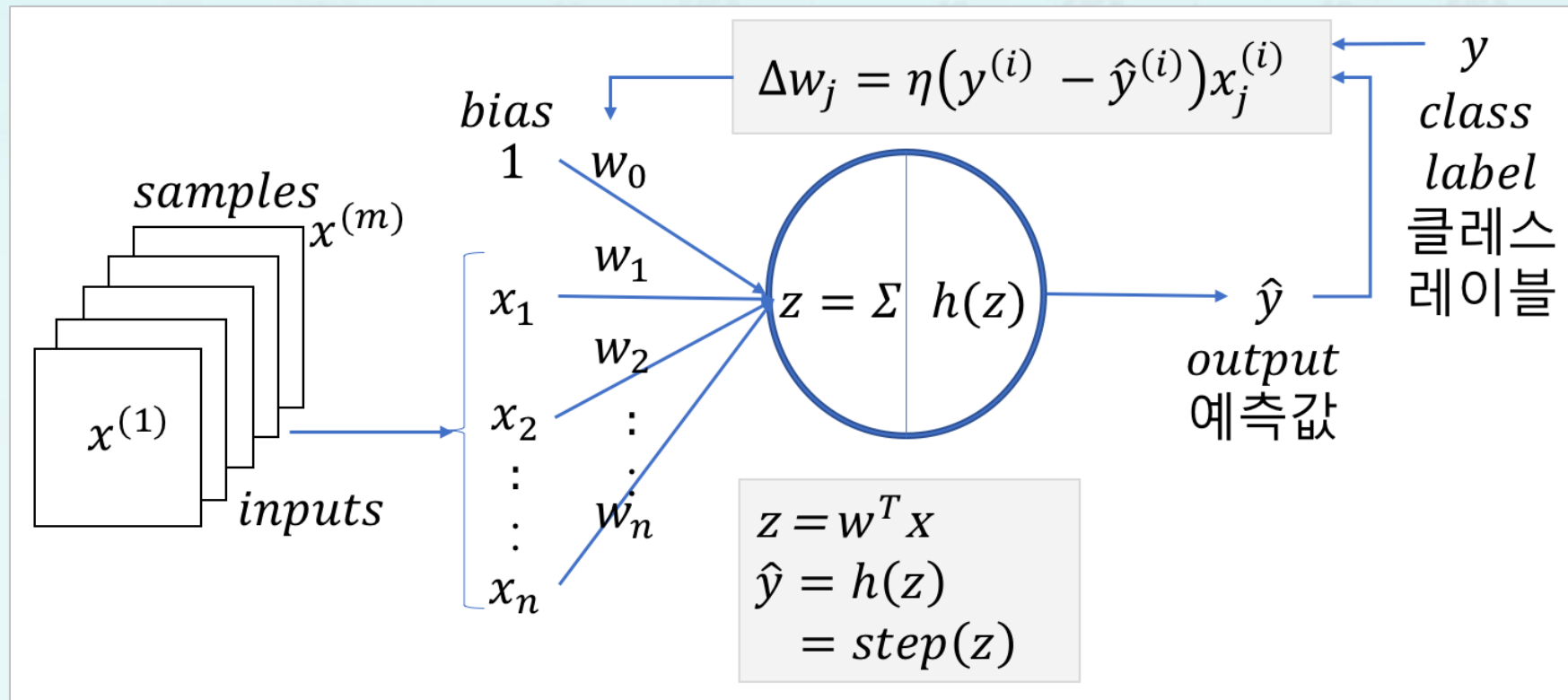


[기능:메소드]

1. 학습 (fit)
2. 순입력 (net_input)
3. 활성화 (activate)
- 4.

2. 객체지향 퍼셉트론: 기능

- 객체
 - 객체의 속성(인스턴스 변수)
 - 객체가 기능(메소드)




[기능:메소드]

1. 학습 (fit)
2. 순입력 (net_input)
3. 활성화 (activate)
4. 예측 (predict)

3. 객체지향 퍼셉트론 구현: 클래스 - 이름

- 클래스
 - 이름: 대문자로 시작




```
1 class Perceptron:
2     """ implements an one-neuron
3     perceptron which performs a linear
4     binary classification """
5     def __init__(self, eta=0.1, epochs=10,
6                 random_seed=1):
7         self.eta = eta
8         self.epochs = epochs
9         self.random_seed = random_seed
10
11     def fit(self, X, y, X0=False):
12         if X0 == False:
13             X = np.c_[ np.ones(len(y)), X]
14             np.random.seed(self.random_seed)
15             self.w = np.random.random(X.shape[1])
```

3. 객체지향 퍼셉트론 구현: 클래스 - 생성자

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화



```
1 class Perceptron:
2     """ implements an one-neuron
3     perceptron which performs a linear
4     binary classification """
5     def __init__(self, eta=0.1, epochs=10,
6                 random_seed=1):
7         self.eta = eta
8         self.epochs = epochs
9         self.random_seed = random_seed
10
11     def fit(self, X, y, X0=False):
12         if X0 == False:
13             X = np.c_[ np.ones(len(y)), X]
14             np.random.seed(self.random_seed)
15             self.w = np.random.random(X.shape[1])
```

3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

- 클래스


- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드

```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8         self.cost_ = []
9         self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```

3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

■ 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - `X`: 입력값
 - `y`: 클래스 레이블
 - `X0`: 편향 여부



```
def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4     np.random.seed(self.random_seed)
5     self.w = np.random.random(X.shape[1])
6
7     self.maxy, self.miny = y.max(), y.min()
8     self.cost_ = []
9     self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```

3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열

```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8         self.cost_ = []
9         self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```

3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

■ 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치

```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8         self.cost_ = []
9         self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```


3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치

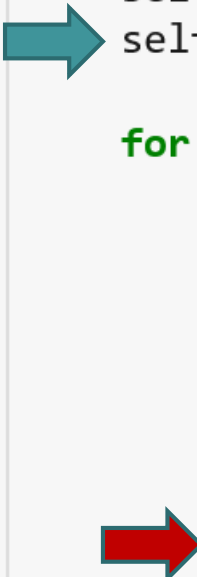
```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8     self.cost_ = []
9     self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```


3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

■ 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치

```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8         self.cost_ = []
9         self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```



3. 객체지향 퍼셉트론 구현: 클래스 - 학습메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치

```
1 def fit(self, X, y, X0=False):
2     if X0 == False:
3         X = np.c_[ np.ones(len(y)), X]
4         np.random.seed(self.random_seed)
5         self.w = np.random.random(X.shape[1])
6
7         self.maxy, self.miny = y.max(), y.min()
8         self.cost_ = []
9         self.w_ = np.array([self.w])
10
11     for i in range(self.epochs):
12         errors = 0
13         for xi, yi in zip(X, y):
14             yhat = self.activate(xi)
15             delta = self.eta * (yi - yhat) * xi
16             self.w = self.w + delta
17             if (y != yhat): errors += 1
18         self.cost_.append(errors)
19         self.w_ = np.vstack([self.w_, self.w])
20     return self
```

3. 객체지향 퍼셉트론 구현: 클래스 - 순입력메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치
- `net_input()`: 순입력 메소드

```
29         self.cost_.append(errors)
30         self.w_ = np.vstack([self.w_, self.w])
31         return self
32
33     def net_input(self, X):
34         if X.shape[0] == self.w.shape[0]:
35             z = np.dot(self.w.T, X)
36         else:
37             z = np.dot(X, self.w[1:]) + self.w[0]
38         return z
39
40     def activate(self, X):
41         mid = (self.maxy + self.miny) / 2
42         return np.where(self.net_input(X) >
43                         mid, self.maxy, self.miny)
44
45     def predict(self, X):
46         return self.activate(X)
```

3. 객체지향 퍼셉트론 구현: 클래스 - 활성화메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치
- `net_input()`: 순입력 메소드
- `activate()`: 활성화 메소드

```
29         self.cost_.append(errors)
30         self.w_ = np.vstack([self.w_, self.w])
31         return self
32
33     def net_input(self, X):
34         if X.shape[0] == self.w.shape[0]:
35             z = np.dot(self.w.T, X)
36         else:
37             z = np.dot(X, self.w[1:]) + self.w[0]
38         return z
39
40     def activate(self, X):
41         mid = (self.maxy + self.miny) / 2
42         return np.where(self.net_input(X) >
43                         mid, self.maxy, self.miny)
44
45     def predict(self, X):
46         return self.activate(X)
```

3. 객체지향 퍼셉트론 구현: 클래스 - 예측메소드

- 클래스

- 이름: 대문자로 시작
- 생성자: `__init__()`
 - 객체생성, 인스턴스 변수 초기화
- `fit()`: 학습 메소드
 - 매개변수
 - 가중치: 1차원 배열
 - `cost_`: 각 `epoch`의 손실
 - `w_`: 각 `epoch`의 가중치
- `net_input()`: 순입력 메소드
- `activate()`: 활성화 메소드
- `predict()`: 예측 메소드

```
29         self.cost_.append(errors)
30         self.w_ = np.vstack([self.w_, self.w])
31         return self
32
33     def net_input(self, X):
34         if X.shape[0] == self.w.shape[0]:
35             z = np.dot(self.w.T, X)
36         else:
37             z = np.dot(X, self.w[1:]) + self.w[0]
38         return z
39
40     def activate(self, X):
41         mid = (self.maxy + self.miny) / 2
42         return np.where(self.net_input(X) >
43                         mid, self.maxy, self.miny)
44
45     def predict(self, X):
46         return self.activate(X)
```

객체지향 퍼셉트론 구현

- 학습 정리
 - 퍼셉트론 클래스 설계
 - 퍼셉트론 함수를 객체지향 퍼셉트론으로 전환
 - **OOP**프로그래밍 경험하기
- 차시 예고
 - **6-2** 객체지향 퍼셉트론 활용

6주차(1/3)

객체지향 퍼셉트론

파이썬으로 배우는 기계학습

한동대학교
김영섭 교수