

11주차(1/3)

# 로지스틱 회귀 3

파이썬으로 배우는 기계학습

한동대학교  
김영섭 교수

# 로지스틱 회귀

---

## ■ 학습 목표

- 교차 엔트로피 손실함수를 이해한다.
- 로지스틱 회귀 신경망의 역전파 계산한다.
- 소프트맥스 활성화 함수를 이해한다.
- 로지스틱 회귀 신경망을 구현한다.

## ■ 학습 내용

- 교차 엔트로피 손실함수와 제곱 합 오차함수의 비교
- 로지스틱 회귀의 역전파를 행렬로 계산하기
- 소프트맥스 활성화 함수를 이해하기
- 로지스틱 회귀 신경망에 구현하여 적용하기

# 1. 손실함수 비교

---

- 제곱 합 오차 함수
  - 목적 - 함수를 최소로 하는 값 찾기
- 교차 엔트로피 손실 함수
  - 목적 - 함수를 최소로 하는 값 찾기

# 1. 손실함수 비교

---

- 제곱 합 오차 함수
  - 목적 - 함수를 최소로 하는 값 찾기
  - 방법 - 모든 자료 오차의 합이 최소
- 교차 엔트로피 손실 함수
  - 목적 - 함수를 최소로 하는 값 찾기
  - 방법 - 모든 자료의 정확한 분류

# 1. 손실함수 비교

- 제공 합 오차 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료 오차의 합이 최소

$$E = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^{(i)} - \hat{y}^{(i)})^2$$

- 교차 엔트로피 손실 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료의 정확한 분류

$$J = - \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

# 1. 손실함수 비교

## ■ 제곱 합 오차 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료 오차의 합이 최소

$$E = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^{(i)} - \hat{y}^{(i)})^2$$

## ■ 교차 엔트로피 손실 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료의 정확한 분류

$$J = - \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

```
1 def MSEcost(self, A2, Y):  
2     m = Y.shape[1] # m examples  
3     cost = np.sum(E2 * E2) / m  
4     return cost
```

# 1. 손실함수 비교:

## ■ 제곱 합 오차 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료 오차의 합이 최소

$$E = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (y^{(i)} - \hat{y}^{(i)})^2$$

```
1 def MSEcost(self, A2, Y):  
2     m = Y.shape[1]  
3     E2 = Y - A2  
4     cost = np.sum(E2 * E2) / m  
5     return cost
```

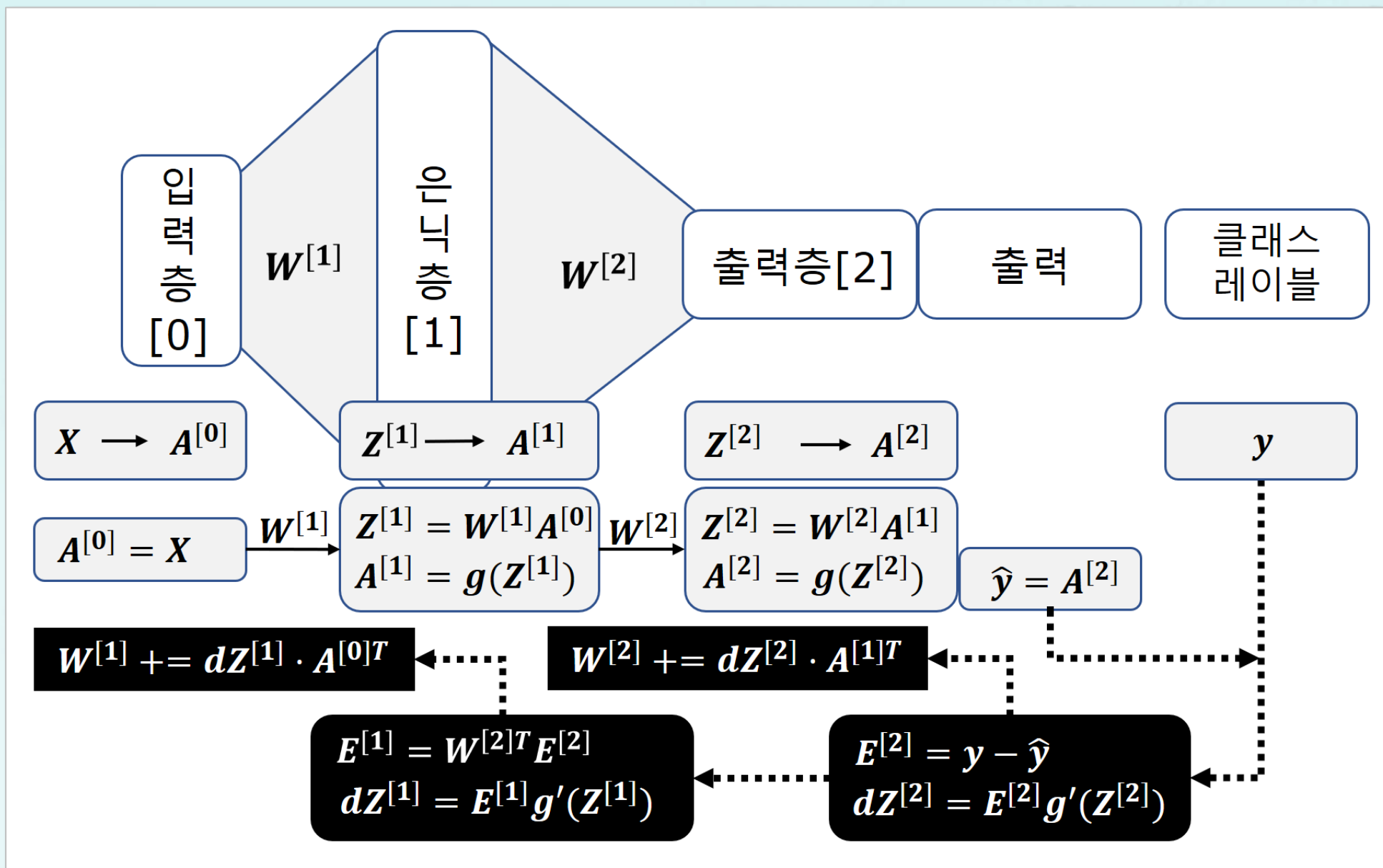
## ■ 교차 엔트로피 손실 함수

- 목적 - 함수를 최소로 하는 값 찾기
- 방법 - 모든 자료의 정확한 분류

$$J = - \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

```
1 def CEcost(self, A2, Y):  
2     m = Y.shape[1] # number of example  
3     logprobs = np.multiply(Y, np.log(A2))  
4     cost = -np.sum(logprobs) / m  
5     cost = np.squeeze(cost)  
6     return cost
```

## 2. 로지스틱 회귀 신경망: 역전파





## 2. 로지스틱 회귀 신경망: 역전파

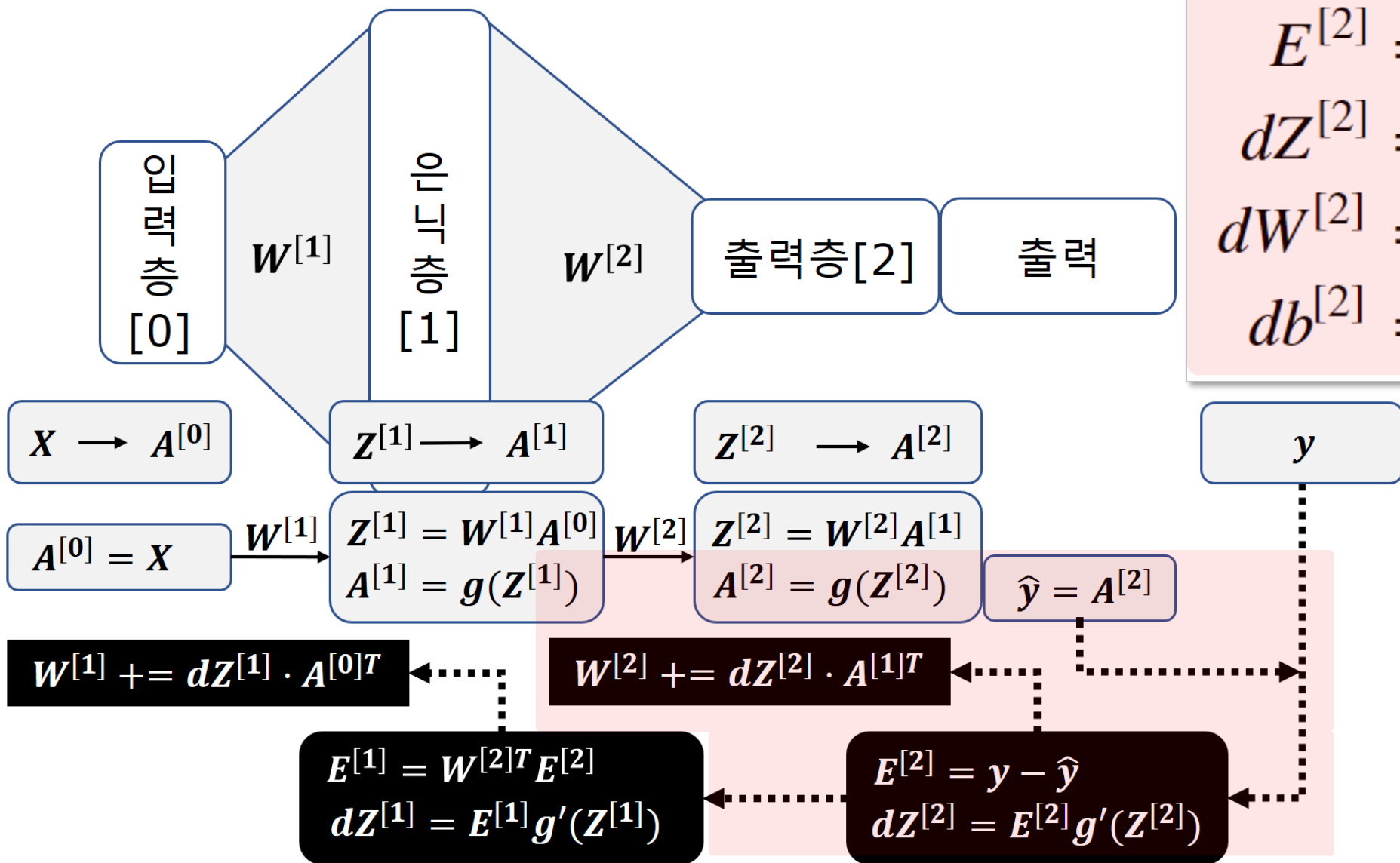
### 출력층 역전파 일반공식

$$E^{[2]} = y - \hat{y}$$

$$dZ^{[2]} = E^{[2]} g^{[2]'}(Z^{[2]})$$

$$dW^{[2]} = dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = dZ^{[2]}$$



## 2. 로지스틱 회귀 신경망: 역전파

- 출력층 오차
  - 출력층 → 은닉층

$$E^{[2]} = y - A^{[2]}$$

$$\begin{aligned} dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$\begin{aligned} dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\ &= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\ &= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T} \end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

### 출력층 역전파 일반공식

$$E^{[2]} = y - \hat{y}$$

$$dZ^{[2]} = E^{[2]} g^{[2]'}(Z^{[2]})$$

$$dW^{[2]} = dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = dZ^{[2]}$$

## 2. 로지스틱 회귀 신경망: 역전파

- 출력층 오차
  - 출력층 → 은닉층

### 출력층 역전파 일반공식

$$E^{[2]} = y - \hat{y}$$

$$dZ^{[2]} = E^{[2]} g^{[2]'}(Z^{[2]})$$

$$dW^{[2]} = dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = dZ^{[2]}$$

$$E^{[2]} = y - A^{[2]}$$

$$\begin{aligned} dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$dW^{[2]} = \frac{\partial E}{\partial W^{[2]}}$$

$$= \frac{1}{m} E^{[2]} \cdot A^{[1]T}$$

$$= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

## 2. 로지스틱 회귀 신경망: 역전파

- 출력층 오차
  - 출력층 → 은닉층

$$\begin{aligned}E^{[2]} &= y - A^{[2]} \\dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\&= E^{[2]}\end{aligned}$$

$$\begin{aligned}dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\&= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\&= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T}\end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

### 출력층 역전파 일반공식

$$E^{[2]} = y - \hat{y}$$

$$dZ^{[2]} = E^{[2]} g^{[2]'}(Z^{[2]})$$

$$dW^{[2]} = dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = dZ^{[2]}$$

## 2. 로지스틱 회귀 신경망: 역전파

- 출력층 오차
  - 출력층 → 은닉층

### 출력층 역전파 일반공식

$$E^{[2]} = y - \hat{y}$$

$$dZ^{[2]} = E^{[2]} g^{[2]'}(Z^{[2]})$$

$$dW^{[2]} = dZ^{[2]} A^{[1]T}$$

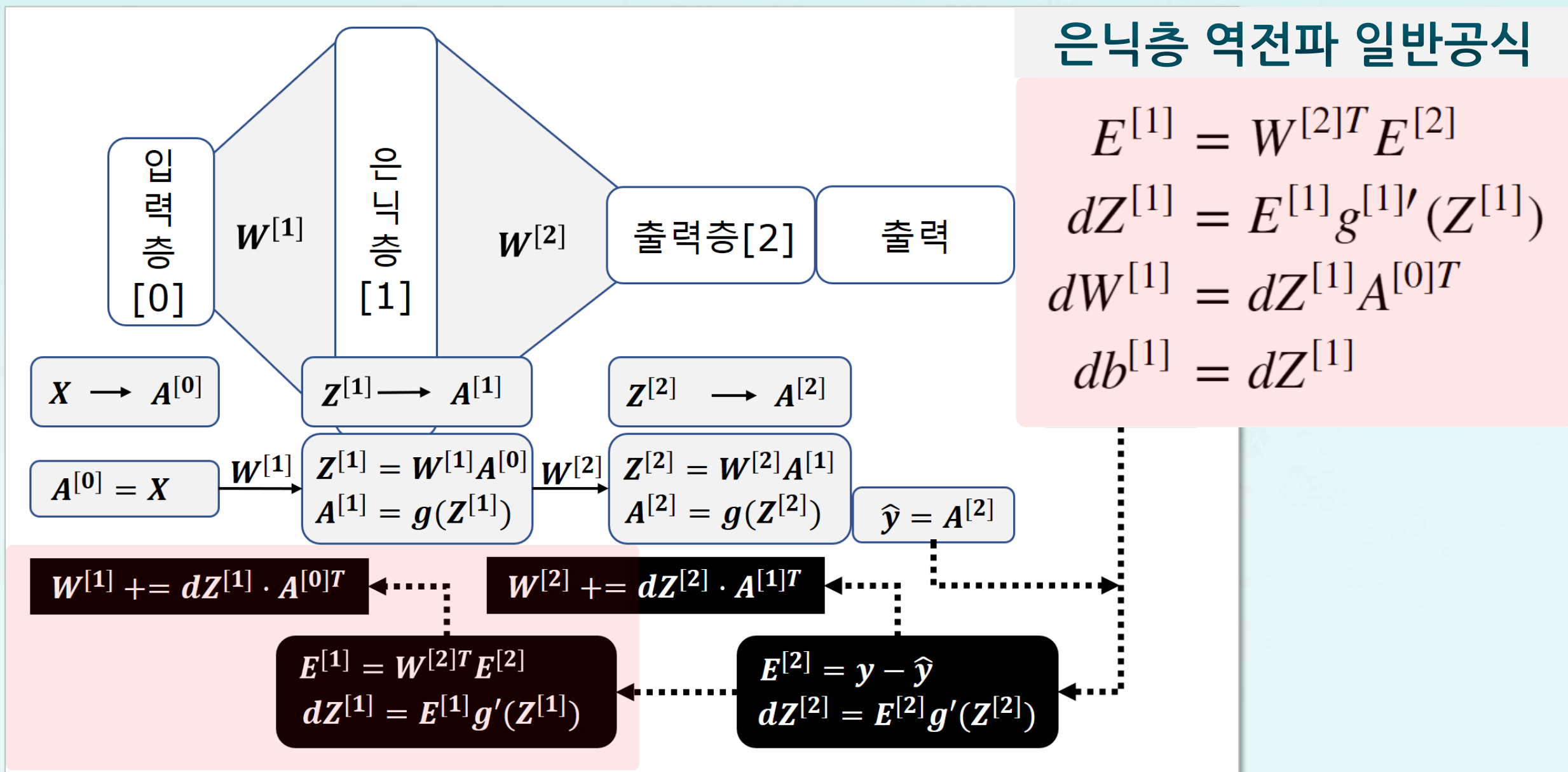
$$db^{[2]} = dZ^{[2]}$$

$$\begin{aligned} E^{[2]} &= y - A^{[2]} \\ dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$\begin{aligned} dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\ &= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\ &= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T} \end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(E^{[2]}, \text{axis} = 1)$$

## 2. 로지스틱 회귀 신경망: 역전파



## 2. 로지스틱 회귀 신경망: 역전파

- 은닉층 오차
  - 은닉층 → 입력층

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$\begin{aligned} dZ^{[1]} &= E^{[1]} * g^{[1]'}(Z^{[1]}) \\ &= E^{[1]} * (1 - \tanh^2(Z^{[1]})) \\ &= E^{[1]} * (1 - A^{[1]2}) \end{aligned}$$

$$dW^{[1]} = dZ^{[1]} \cdot A^{[0]T}$$

$$db^{[1]} = np.sum(dZ^{[1]}, axis = 1)$$

### 은닉층 역전파 일반공식

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$dZ^{[1]} = E^{[1]} g^{[1]'}(Z^{[1]})$$

$$dW^{[1]} = dZ^{[1]} A^{[0]T}$$

$$db^{[1]} = dZ^{[1]}$$

## 2. 로지스틱 회귀 신경망: 역전파

- 은닉층 오차
  - 은닉층 → 입력층

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$\begin{aligned} dZ^{[1]} &= E^{[1]} * g^{[1]'}(Z^{[1]}) \\ &= E^{[1]} * (1 - \tanh^2(Z^{[1]})) \end{aligned}$$

$$= E^{[1]} * (1 - A^{[1]2})$$

$$dW^{[1]} = dZ^{[1]} \cdot A^{[0]T}$$

$$db^{[1]} = np.sum(dZ^{[1]}, axis = 1)$$

$$\begin{aligned} g'(Z^{[1]}) &= \tanh'(Z^{[1]}) \\ &= 1 - \tanh^2(Z^{[1]}) \end{aligned}$$

여기서 \* 는 원소 별 곱셈을 의미합니다.



## 2. 로지스틱 회귀 신경망: 역전파

- 은닉층 오차
  - 은닉층 → 입력층

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$\begin{aligned} dZ^{[1]} &= E^{[1]} * g^{[1]'}(Z^{[1]}) \\ &= E^{[1]} * (1 - \tanh^2(Z^{[1]})) \\ &= E^{[1]} * (1 - A^{[1]2}) \end{aligned}$$

$$A^{[1]} = g(Z^{[1]}) = \tanh(Z^{[1]})$$

$$dW^{[1]} = dZ^{[1]} \cdot A^{[0]T}$$

$$db^{[1]} = np.sum(dZ^{[1]}, axis = 1)$$

여기서 \* 는 원소 별 곱셈을 의미합니다.

## 2. 로지스틱 회귀 신경망: 역전파

- 은닉층 오차
  - 은닉층 → 입력층

### 은닉층 역전파 일반공식

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$dZ^{[1]} = E^{[1]} g^{[1]'}(Z^{[1]})$$

$$dW^{[1]} = dZ^{[1]} A^{[0]T}$$

$$db^{[1]} = dZ^{[1]}$$

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$\begin{aligned} dZ^{[1]} &= E^{[1]} * g^{[1]'}(Z^{[1]}) \\ &= E^{[1]} * (1 - \tanh^2(Z^{[1]})) \\ &= E^{[1]} * (1 - A^{[1]2}) \end{aligned}$$

$$dW^{[1]} = dZ^{[1]} \cdot A^{[0]T}$$

$$db^{[1]} = np.sum(dZ^{[1]}, axis = 1)$$

### 3. 활성화 함수: 소프트 맥스

---

- 소프트 맥스
  - 정규화

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

### 3. 활성화 함수: 소프트 맥스

---

- 소프트 맥스
  - 정규화
  - 상대적 비교

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

### 3. 활성화 함수: 소프트 맥스

---

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

### 3. 활성화 함수: 소프트 맥스

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

- 시그모이드
  - 강아지(**0.9**) 고양이(**0.8**) 호랑이(**0.7**)

### 3. 활성화 함수: 소프트 맥스

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

- 시그모이드
  - 강아지(**0.9**) 고양이(**0.8**) 호랑이(**0.7**)
  - 강아지(**0.1**) 고양이(**0.3**) 호랑이(**0.5**)

### 3. 활성화 함수: 소프트 맥스

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

- 시그모이드
  - 강아지(**0.9**) 고양이(**0.8**) 호랑이(**0.7**)
  - 강아지(**0.1**) 고양이(**0.3**) 호랑이(**0.5**)
- 소프트 맥스
  - 강아지(**0.6**) 고양이(**0.2**) 호랑이(**0.1**)
  - 강아지(**0.0**) 고양이(**0.2**) 호랑이(**0.6**)

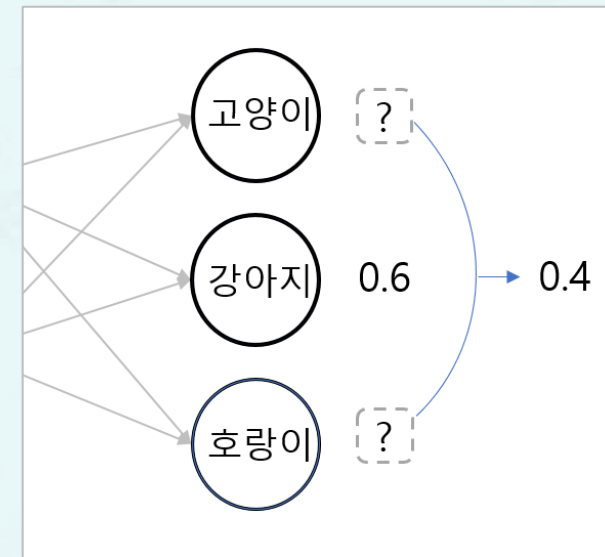


### 3. 활성화 함수: 소프트 맥스

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

- 시그모이드
  - 강아지(**0.9**) 고양이(**0.8**) 호랑이(**0.7**)
  - 강아지(**0.1**) 고양이(**0.3**) 호랑이(**0.5**)
- 소프트 맥스
  - 강아지(**0.6**) 고양이(**0.2**) 호랑이(**0.1**)
  - 강아지(**0.0**) 고양이(**0.2**) 호랑이(**0.6**)



### 3. 활성화 함수: 소프트 맥스

- 소프트 맥스
  - 정규화
  - 상대적 비교
  - 분류

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad j = 1, \dots, K$$

```
def softmax(self, a):  
    exp_a = np.exp(a - np.max(a))  
    return exp_a / np.sum(exp_a)
```

- 시그모이드
  - 강아지(**0.9**) 고양이(**0.8**) 호랑이(**0.7**)
  - 강아지(**0.1**) 고양이(**0.3**) 호랑이(**0.5**)
- 소프트 맥스
  - 강아지(**0.6**) 고양이(**0.2**) 호랑이(**0.1**)
  - 강아지(**0.0**) 고양이(**0.2**) 호랑이(**0.6**)

## 4. 로지스틱 회귀: 구현 - 순전파

- 순전파: **forpass()**
  - 은닉층: **sigmoid()**
  - 출력층: **softmax()**

```
1 def forpass(self, A0):
2     Z1 = np.dot(self.W1, A0) + self.b1
3     A1 = self.g(Z1)
4     Z2 = np.dot(self.W2, A1) + self.b2
5     A2 = self.softmax(Z2)
6     return Z1, A1, Z2, A2
```

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                        keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 구현 - 교차 엔트로피

- 손실함수: **CEcost()**

- 교차 엔트로피

$$J = - \sum_i y^{(i)} \log(\hat{y}^{(i)})$$

```
1 def CEcost(self, A2, Y):
2     m = Y.shape[1] # number of example
3     logprobs = np.multiply(Y, np.log(A2))
4     cost = -np.sum(logprobs) / m
5     cost = np.squeeze(cost)
6     return cost
```

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                        keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 구현 - 역전파

- 출력층에서 은닉층 역전파

$$E^{[2]} = y - A^{[2]}$$

$$\begin{aligned} dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$\begin{aligned} dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\ &= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\ &= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T} \end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                          keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```



## 4. 로지스틱 회귀: 구현 - 역전파

- 출력층에서 은닉층 역전파

$$E^{[2]} = y - A^{[2]}$$

$$\begin{aligned} dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$dW^{[2]} = \frac{\partial E}{\partial W^{[2]}}$$

$$= \frac{1}{m} E^{[2]} \cdot A^{[1]T}$$

$$= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                          keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 구현 - 역전파

- 출력층에서 은닉층 역전파

$$\begin{aligned} E^{[2]} &= y - A^{[2]} \\ dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$\begin{aligned} dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\ &= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\ &= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T} \end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                        keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 구현 - 역전파

- 출력층에서 은닉층 역전파

$$\begin{aligned} E^{[2]} &= y - A^{[2]} \\ dZ^{[2]} &= E^{[2]} g^{[2]'}(Z^{[2]}) \\ &= E^{[2]} \end{aligned}$$

$$\begin{aligned} dW^{[2]} &= \frac{\partial E}{\partial W^{[2]}} \\ &= \frac{1}{m} E^{[2]} \cdot A^{[1]T} \\ &= \frac{1}{m} dZ^{[2]} \cdot A^{[1]T} \end{aligned}$$

$$db^{[2]} = \frac{1}{m} \text{np.sum}(dZ^{[2]}, \text{axis} = 1)$$

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                        keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```



## 4. 로지스틱 회귀: 구현 - 역전파

- 은닉층에서 입력층 역전파

$$E^{[1]} = W^{[2]T} E^{[2]}$$

$$\begin{aligned} dZ^{[1]} &= E^{[1]} * g^{[1]'}(Z^{[1]}) \\ &= E^{[1]} * (1 - \tanh^2(Z^{[1]})) \\ &= E^{[1]} * (1 - A^{[1]2}) \end{aligned}$$

$$dW^{[1]} = dZ^{[1]} \cdot A^{[0]T}$$

$$db^{[1]} = np.sum(dZ^{[1]}, axis = 1)$$

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                          keepdims=True) / self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 구현

- 은닉층에서 입력층 역전파

```
1 #sigmoid
2 def g(self, x):
3     x = np.clip(x, -500, 500)
4     return 1.0/(1.0 + np.exp(-x))
5 def g_prime(self, x):
6     return self.g(x) * (1 - self.g(x))
```

```
1 def fit(self, X, y):
2     self.cost_ = []
3     self.m_samples = len(y)
4     Y = joy.one_hot_encoding(y, self.n_y)
5     for epoch in range(self.epochs):
6         for sample in range(self.m_samples):
7             A0 = np.array(X[sample], ndmin=2).T
8             Y0 = np.array(Y[sample], ndmin=2).T
9             Z1, A1, Z2, A2 = self.forpass(A0)
10            cost = self.CEcost(A2, Y0) #cross-entropy
11            self.cost_.append(cost)
12
13            E2 = Y0 - A2 # Backprop
14            dZ2 = E2
15            dW2 = np.dot(dZ2, A1.T) / self.m_samples
16            db2 = np.sum(dZ2, axis=1,
17                        keepdims=True)/self.m_samples
18            E1 = np.dot(self.W2.T, E2)
19            dZ1 = E1 * self.g_prime(Z1) #sigmoid
20            #dZ1 = E1 * (1 - np.power(A1, 2)) #tanh
21            dW1 = np.dot(dZ1, A0.T)
22            db1 = np.sum(dZ1, axis=1, keepdims=True)
23            self.W1 += self.eta * dW1
24            self.b1 += self.eta * db1
25            self.W2 += self.eta * dW2
26            self.b2 += self.eta * db2
27        return self
```

## 4. 로지스틱 회귀: 실행

```
import joy
(X, y), (Xtest, ytest) = joy.load_mnist()
self_accuracy = []
test_accuracy = []
epoch_list = np.arange(1, 31)
for e in epoch_list:
    nn = LogisticNeuron_stochastic(784, 100, 10,
                                   eta = 0.2, epochs = e)
    nn.fit(X, y)
    self_accuracy.append(nn.evaluate(X, y))
    test_accuracy.append(nn.evaluate(Xtest, ytest))
```

## 4. 로지스틱 회귀: 실행

```
import joy
(X, y), (Xtest, ytest) = joy.load_mnist()
self_accuracy = []
test_accuracy = []
epoch_list = np.arange(1, 31)
for e in epoch_list:
    nn = LogisticNeuron_stochastic(784, 100, 10,
                                    eta = 0.2, epochs = e)
    nn.fit(X, y)
    self_accuracy.append(nn.evaluate(X, y))
    test_accuracy.append(nn.evaluate(Xtest, ytest))
```

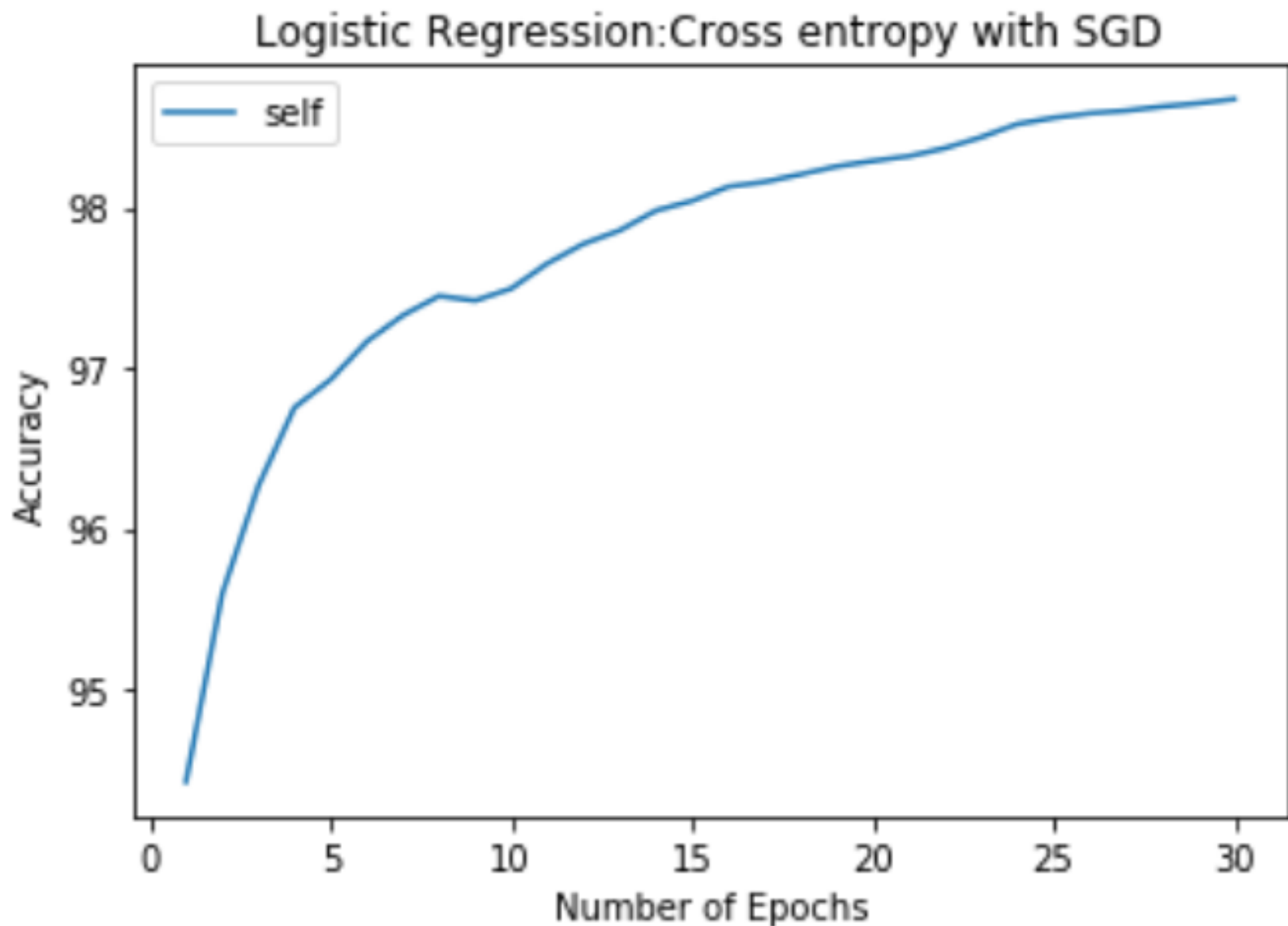
## 4. 로지스틱 회귀: 실행

---

```
plt.plot(epoch_list, self_accuracy, label='self')
plt.plot(epoch_list, test_accuracy, label='test')
plt.xlabel('Number of Epochs')
plt.ylabel('Accuracy')
plt.title('Logistic Regression: Cross entropy with SGD')
plt.legend(loc='best')
plt.show()
```

## 4. 로지스틱 회귀: 실행

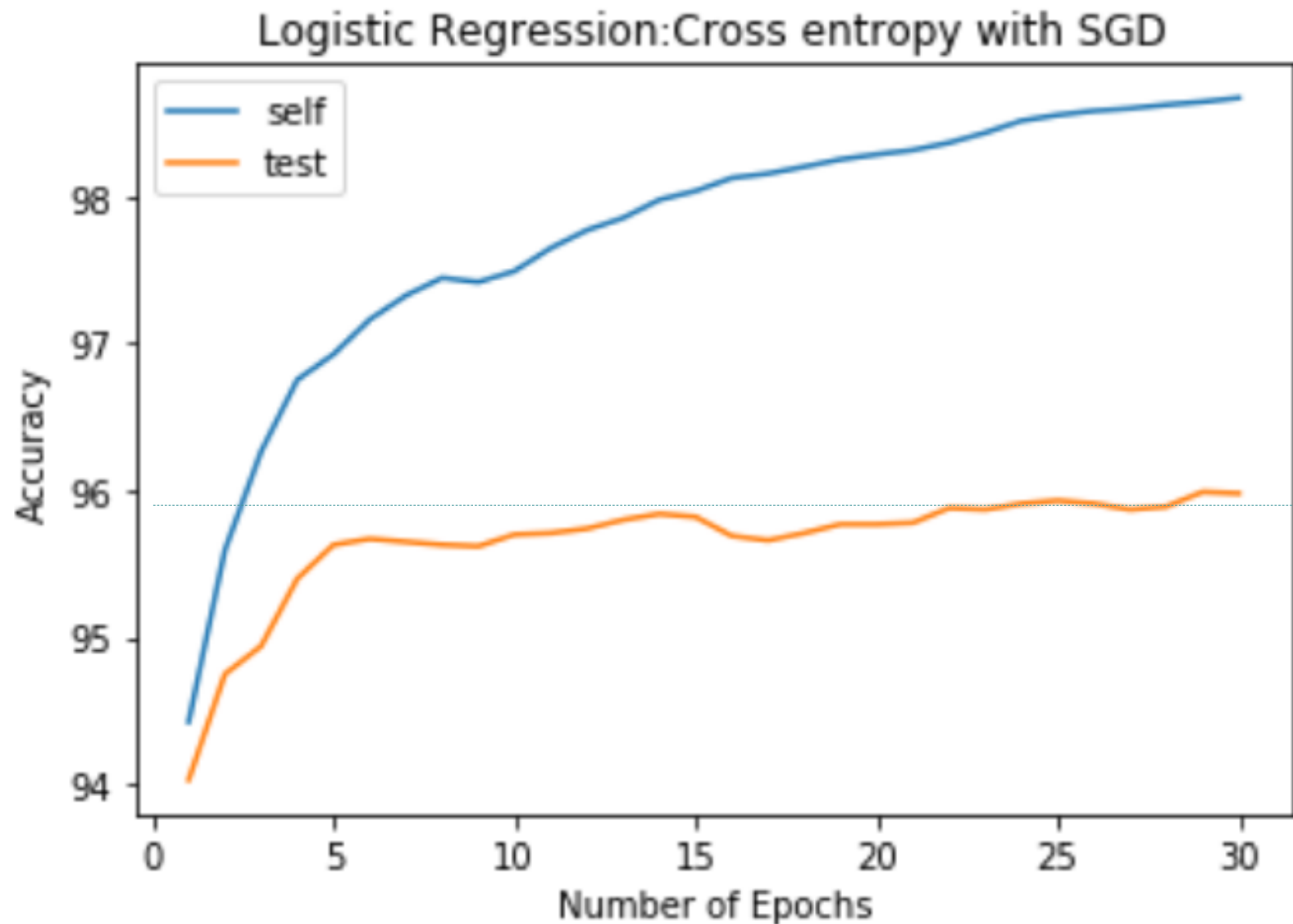
```
plt.plot(epoch_list, se  
plt.plot(epoch_list, te  
plt.xlabel('Number of E  
plt.ylabel('Accuracy')  
plt.title('Logistic Reg  
plt.legend(loc='best')  
plt.show()
```





## 4. 로지스틱 회귀: 실행

```
plt.plot(epoch_list, se
plt.plot(epoch_list, te
plt.xlabel('Number of E
plt.ylabel('Accuracy')
plt.title('Logistic Reg
plt.legend(loc='best')
plt.show()
```



# 로지스틱 회귀

---

- 학습 정리
  - 교차 엔트로피 손실함수의 이해하기
  - 로지스틱 회귀의 역전파를 행렬로 계산하기
  - 소프트맥스 활성화 함수 이해하기
  - 로지스틱 회귀 신경망 구현하기
- 차시 예고
  - **11-2 MNIST Dataset**



10주차(3/3)

# 로지스틱 회귀 3

파이썬으로 배우는 기계학습

한동대학교  
김영섭 교수

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