Chapter. 02

선형대수학 응용

# 경사하강법과 로지스틱 회귀

FASTCAMPUS
ONLINE
금융공학/퀀트 I

강사. 장순용

# 키포인트

- 로지스틱 회귀.
- 경사하강법 (Gradient Descent).

# I 로지스틱회귀



- 독립변수 $\{X_i\}$를 선형조합하여 Logit $S$를 만든다.

$$S = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_K X_K$$

# I 로지스틱회귀



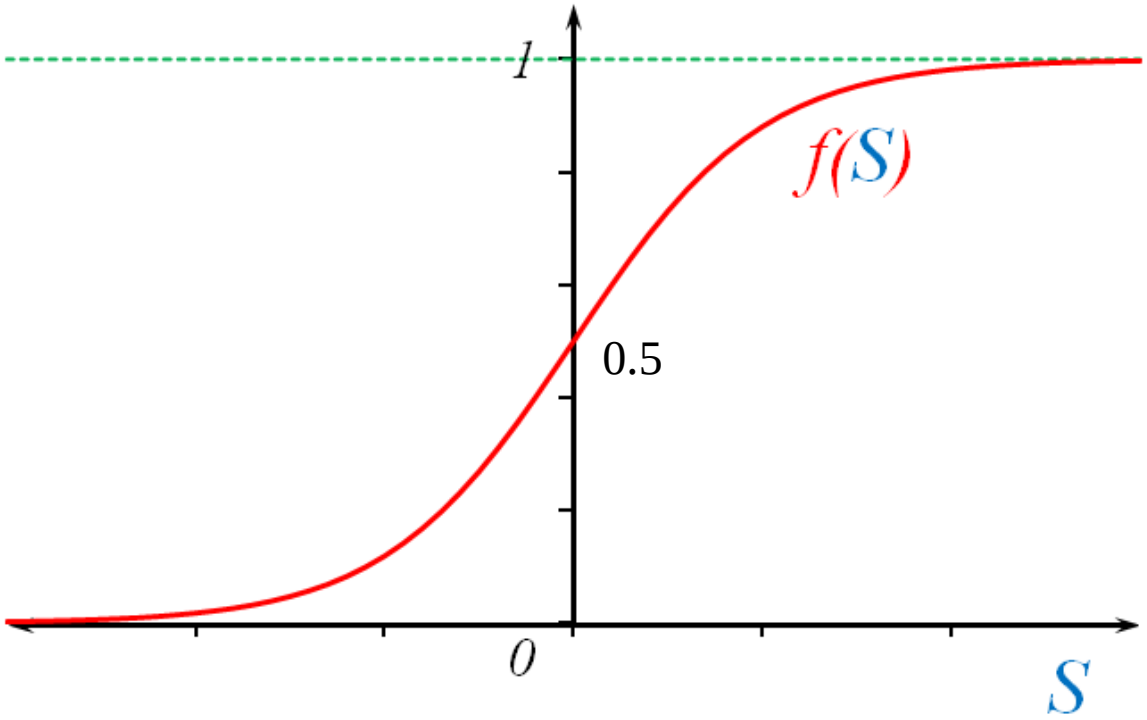
- 종속변수 $Y$의 값이 1이 될 조건부 확률 $P(Y = 1|\{x_i\})$은 “로지스틱 함수” 또는 “Sigmoid 함수”를 사용해서 계산된다.

$$f(S) = \frac{e^{S}}{1 + e^{S}}$$

$\Rightarrow$ 인공신경망에서 “활성화 함수” (activation function)의 역할을 함.

# I 로지스틱회귀

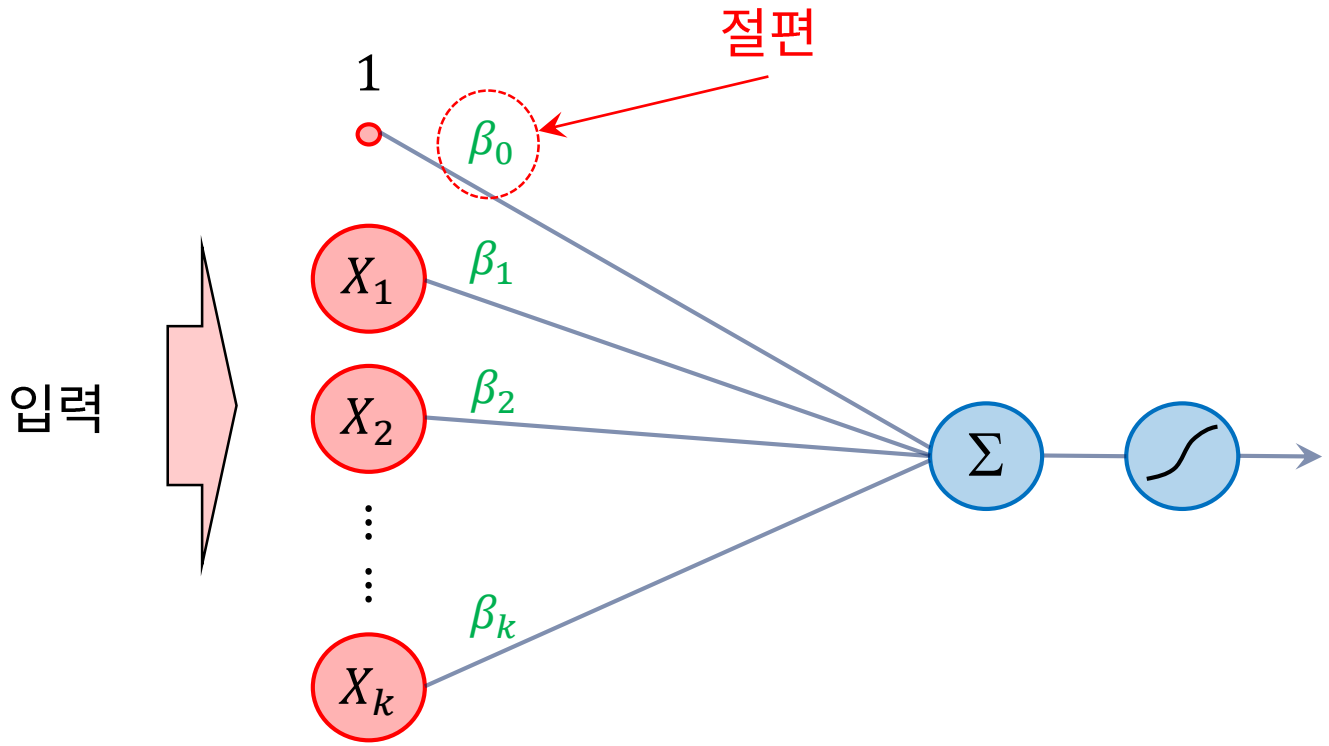





FAST CAMPUS
ONLINE
장순용 강사.

# I 로지스틱회귀 구조

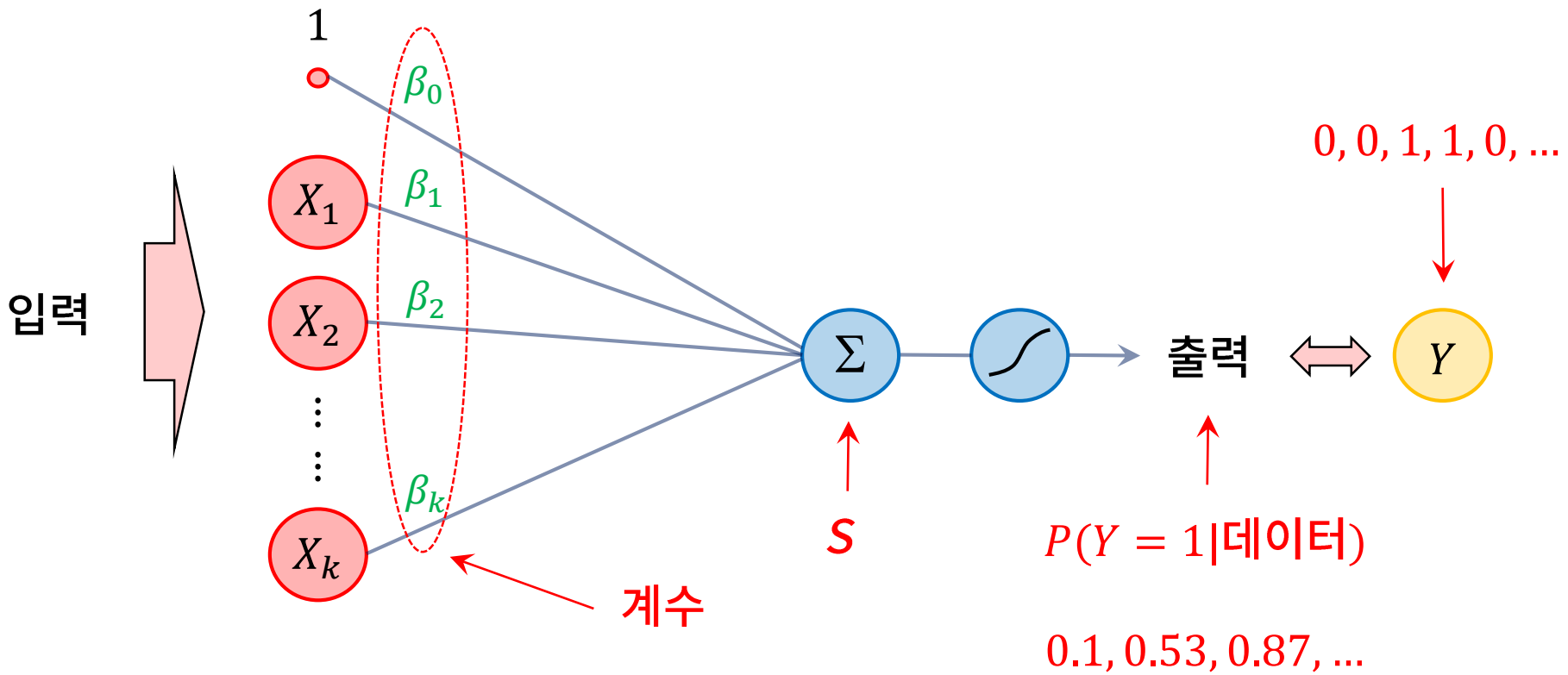





FAST CAMPUS
ONLINE
장순용 강사.

# I 로지스틱회귀 예측






FAST CAMPUS
ONLINE
장순용 강사.

# l 로지스틱회귀 학습



- 학습이란 모형의 파라미터, $\beta$ 계수들의 값을 구하는 것.

$$S = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_K X_K$$

$$f(S) = \frac{e^S}{1+e^S}$$

# I 로지스틱회귀 학습



- 로그우도 $L$ 을 최소화하는 방법으로 $\beta$ 계수들의 값을 구할 수 있다.
- 로그우도 $L$ 은 일종의 “손실함수”이다.

  $\Rightarrow$ 예측 오류에 의한 “손실”을 최소화 하고자 한다.

# I 로지스틱회귀 학습



- 로그우도 $L$ 의 수식은 다음과 같다:

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{N} Log\left(1 + e^{-y_i \boldsymbol{\beta}^t \boldsymbol{x}_i}\right)$$

$\Rightarrow \boldsymbol{x}_i$와 $y_i$는 실제 데이터 값을 의미한다. $(y_i = -1 \; or \; +1)$

$$\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,K} \end{bmatrix} \quad , \quad \boldsymbol{x}_i^t = [1, x_{i,1}, x_{i,2}, \cdots, x_{i,K}]$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_K \end{bmatrix} \quad , \quad \boldsymbol{\beta}^t = [\beta_0, \beta_1, \cdots, \beta_K]$$

# I 로지스틱회귀 학습



- 로그우도 $L$ 의 수식은 다음과 같다:

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{N} Log\left(1 + e^{-y_i \boldsymbol{\beta}^{\boldsymbol{t}} \boldsymbol{x}_i}\right)$$

- $L$의 값은 gradient 방향으로 증가율 최고.

  ⇒ -gradient 방향으로는 감소율 최고.

# I 로지스틱회귀 학습



- $L$의 gradient는 다음과 같이 구할 수 있다:

$$\nabla L(\boldsymbol{\beta}) = -\sum_{i=1}^{N} \frac{y_i \boldsymbol{x}_i e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}{1 + e^{-\boldsymbol{\beta}^t \boldsymbol{x}_i}}$$

$\Rightarrow$ $L$을 벡터미분 하여 구할 수 있다.

$$\nabla L(\boldsymbol{\beta}) = \begin{bmatrix} \frac{\partial L}{\partial \beta_0} \\ \frac{\partial L}{\partial \beta_1} \\ \vdots \\ \frac{\partial L}{\partial \beta_K} \end{bmatrix}$$

# I 로지스틱회귀 학습



- Gradient descent 알고리즘: 감소율 최고인 $-\nabla L$로 반복적으로 이동.

  a). $\boldsymbol{\beta}$를 임의의 값으로 초기화 한다.

  b). Gradient $\nabla L$를 계산한다.

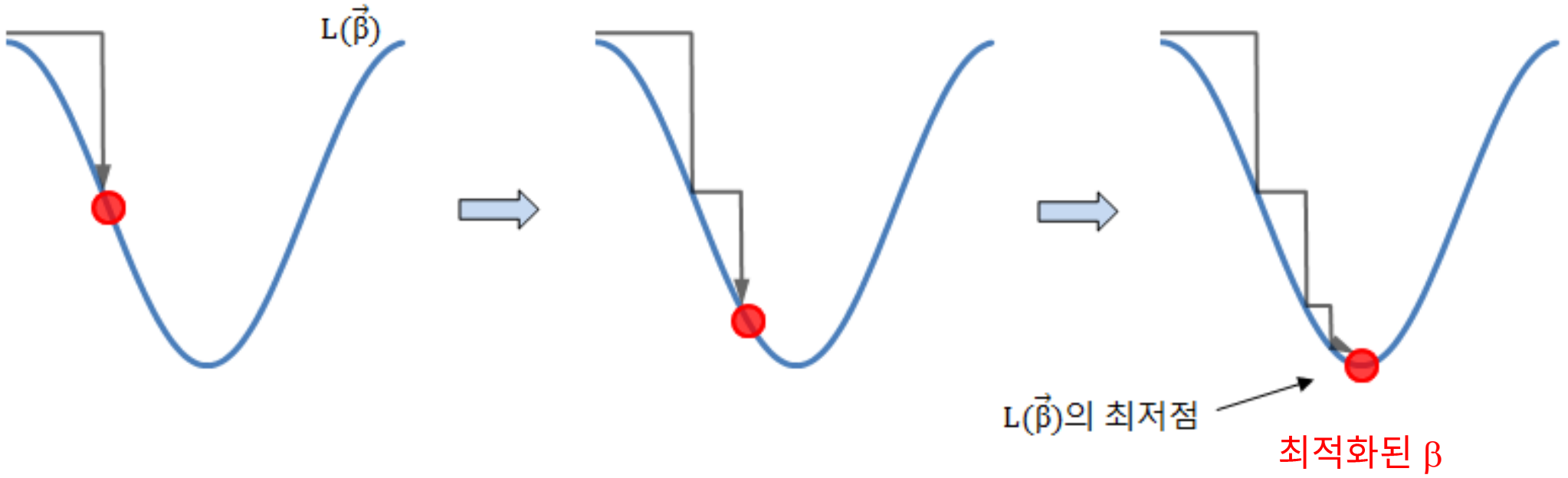
  c). $\boldsymbol{\beta}$를 $\boldsymbol{\beta} - \eta \nabla L$와 같이 갱신한다. “Learning rate” $\eta$로 수렴 속도 조절.

  d). 스텝 b)로 돌아가서 일정 횟수만큼 반복한다.

# I 로지스틱회귀 학습



- Gradient descent 알고리즘: 감소율 최고인 $-\nabla L$로 반복적으로 이동.



# I 로지스틱회귀 학습



- 다음과 같이 코딩으로 경사하강법을 구현할 수 있다 (Python):

```
def sigmoid(x):
    s = 1.0/(1.0 + np.exp(-x))
    return s
def gradient(X, Y, beta):
    z = np.dot(X,beta.T)*Y
    ds = -Y*(1-sigmoid(z))*X
   return ds.sum(axis=0)
```

# I 로지스틱회귀 학습



- 다음과 같이 코딩으로 경사하강법을 구현할 수 있다 (Python):

```
def train(self, input_X, input_Y, n_epochs):
    ones_column = np.ones((input_X.shape[0],1))
    X = np.concatenate((ones_column,input_X), axis=1)
    Y = (2*input_Y - 1).reshape(-1,1)                    # 0 or 1  =>  -1 or 1.
    for n in range(n_epochs):
        self.beta = self.beta - self.rate*gradient(X,Y,self.beta)
    return self.beta
```

끝.



# 감사합니다.


FAST CAMPUS
ONLINE
장순용 강사.