분산파일시스템 기반 빅데이터 플랫폼 개발자 양성 과정

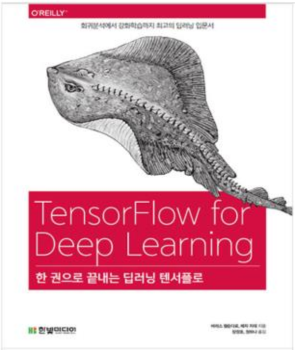
KOREA SOFTWARE INDUSTRY ASSOCIATION

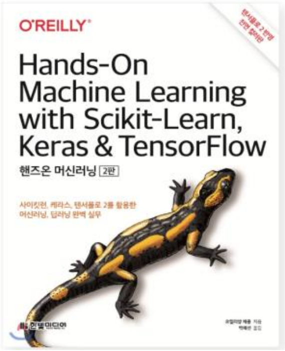
한국소프트웨어산업협회 ITSQF 재직자 직무 교육

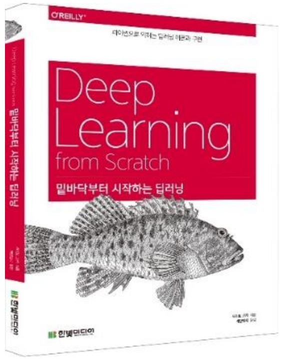
이경화 *(oracle@ssu.ac.kr)*

Deep Learning

Tensorflow.Keras

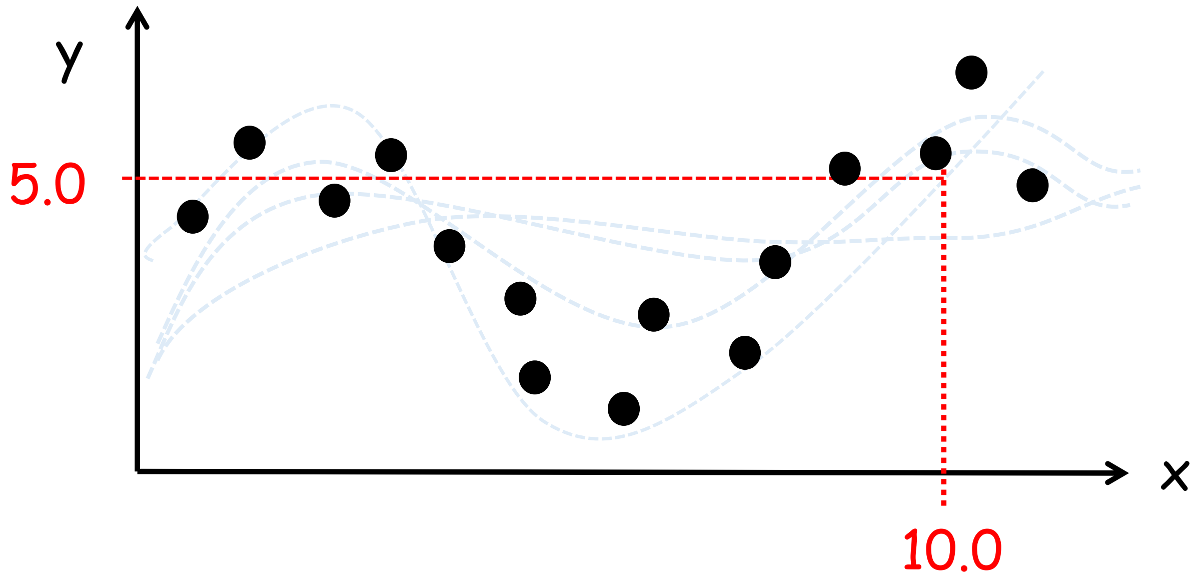






Goal of Machine Learning

- 주어진 데이터를 가장 잘 설명하는 함수란?
  - 함수 f가 주어진 데이터 x에 가장 잘 부합하는 W값을 구한다
  - 구해진 W값을 사용해 y값을 계산한다



$$y = f(x)$$

$$f(x_1, x_2, \dots, x_n; w_1, w_2, \dots, w_m)$$

$$f(x; 1.0, 1.0, 1.0)$$

$$f(x; 1.5, 1.0, 1.5)$$

$$f(x; 1.5, -1.0, 1.5)$$

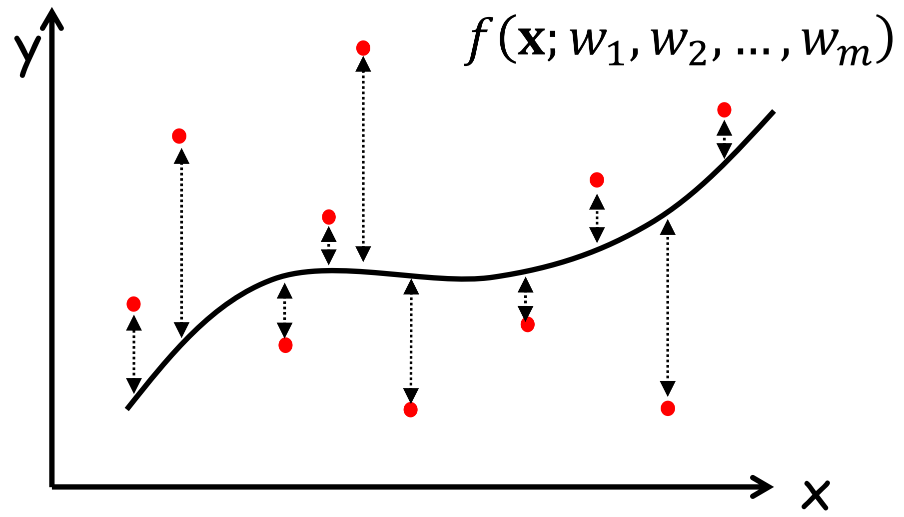
$$f(x; 1.7, -1.2, 1.6)$$

$$5.0 = f(10.0\ ; 1.7, -1.2, 1.6)$$

Goal of Machine Learning

KOSA 한국소프트웨어산업협회

- 주어진 데이터를 가장 잘 설명하는 함수란?
  - 주어진 데이터와 오류를 최소하는 함수
  - $Error = \sum_{(\mathbf{x},y)\in Data} \left(y - f(\mathbf{x}; w_1, w_2, \dots, w_m)\right)^2$
  - 즉 Error를 최소화하는 $w_1, w_2, \dots, w_m$ 값을 갖는 함수
  - $E(w_1, w_2, \dots, w_m) = \sum_{(\mathbf{x},y)\in Data} \left(y - f(\mathbf{x}; w_1, w_2, \dots, w_m)\right)^2$



Goal of Machine Learning

- **Minimized the function E**

$$E(w_1) = {w_1}^2 + 2w_1 + 3$$

1. E를 w_1에 대해 편미분 : $\dfrac{dE(w_1)}{dw_1} = 2w_1 + 2$

2. 편미분한 값을 0으로 만드는 w_1 해 구하기 : $2w_1 + 2 = 0$
   $w_1 = -1$

Goal of Machine Learning

- **Minimized the function E**

$$E(w_1, w_2, \dots, w_m) = \sum_{(\mathrm{x},y)\in Data} \left(y - f(\mathrm{x}; w_1, w_2, \dots, w_m)\right)^2$$

- E를 w_i들에 대해서 편미분 후, 이 값들을 모두 0으로 만드는 w_i를 해 구하기

$$\left.\begin{aligned} \frac{\partial E}{\partial w_1} &= 0 \\ \frac{\partial E}{\partial w_2} &= 0 \\ \frac{\partial E}{\partial w_m} &= 0 \end{aligned}\right\}$$

를 동시에 만족하는 $w_1, w_2, \dots, w_m$ 구하기

Minimized the Sum of Squared Errors(SSE)

- **Minimized the function E**

$$E = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} \left(\boxed{(b_0} + \boxed{b_1} * x_i \right) - y_i)^2$$

$$\frac{\partial E}{\partial b_0} = 0$$

$$\frac{\partial E}{\partial b_1} = 0$$

2 Equations, 2 Unknowns (b_0, b_1)
Solve to get b_0 and b_1 for minima

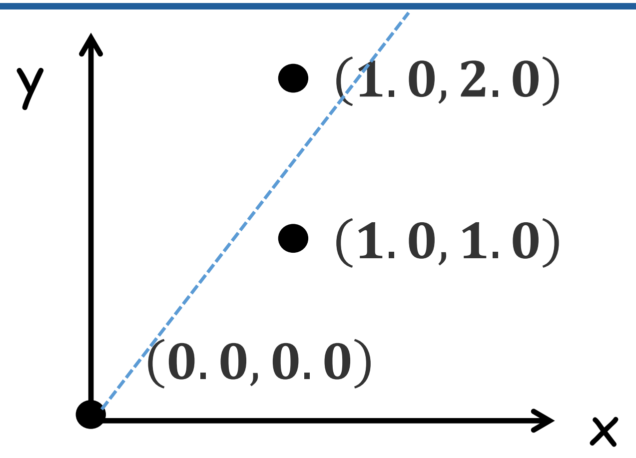
$$\hat{y} = b_0 + b_1 * x$$

Goal of Machine Learning

- **Minimized the function E**

$$\boldsymbol{f(x; w_0, w_1) = \boxed{w_1} x + \boxed{w_0}}$$

$$\boldsymbol{Data = \{(0.0, 0.0), (1.0, 1.0), (1.0, 2.0)\}}$$



- **SSE =** $E = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$

$$2.0 = f(1.0; w_0, w_1) \qquad [\boldsymbol{f(0.0; w_0, w_1) - 0.0}]^2$$

$$1.0 = f(1.0; w_0, w_1) \qquad [\boldsymbol{f(1.0; w_0, w_1) - 1.0}]^2$$

$$0.0 = f(0.0; w_0, w_1) \qquad [\boldsymbol{f(1.0; w_0, w_1) - 2.0}]^2$$

$$E(w_0, w_1) = (\boldsymbol{0.0} - w_0)^2 + \left(\boldsymbol{1.0} - (w_1 + w_0)\right)^2 + \left(\boldsymbol{2.0} - (w_1 + w_0)\right)^2$$

$$\boldsymbol{E(w_0, w_1) = 2w_1{}^2 + 3w_0{}^2 - 6w_1 - 6w_0 + 4w_1 w_0 + 5}$$

$$\boldsymbol{f(x; w_0, w_1) = 1.5x + 0.0}$$

$$\frac{\partial E}{\partial \boldsymbol{w_1}} = 4w_1 + 4w_0 - 6 \qquad \boldsymbol{w_1 = 1.5}$$

$$\frac{\partial E}{\partial \boldsymbol{w_0}} = 4w_1 + 6w_0 - 6 \qquad \boldsymbol{w_0 = 0.0}$$

Gradient Descent Algorithm

- **Minimizing Cost**

$$H(x) = Wx + b$$

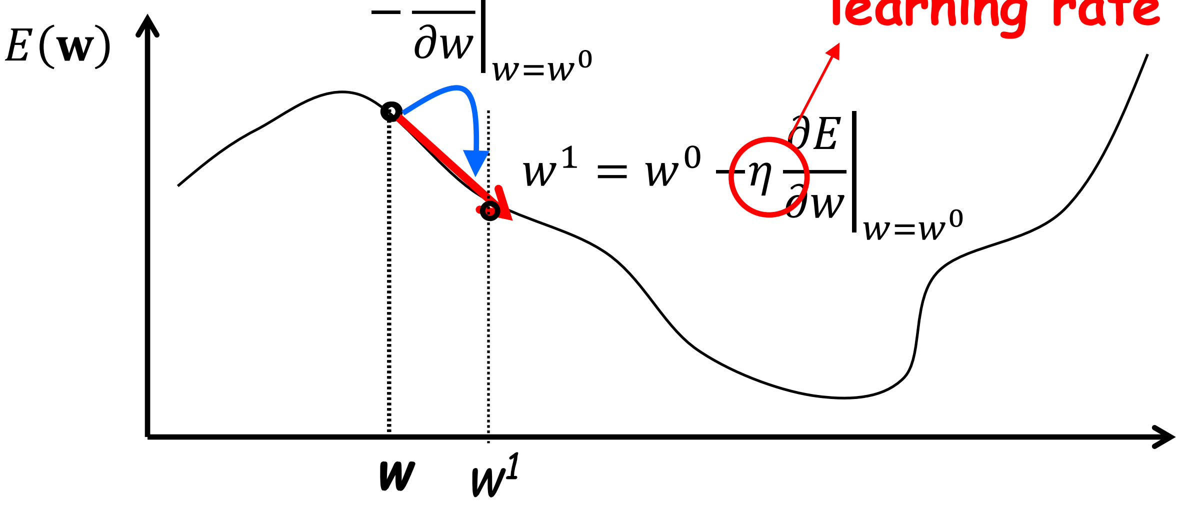
$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

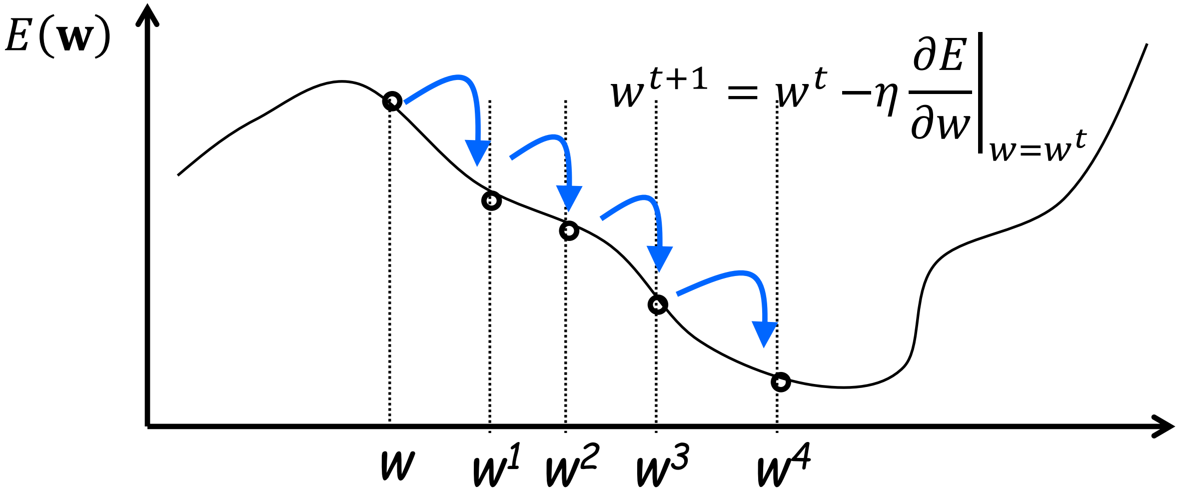
⬇

$$cost(W) = \frac{1}{2m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

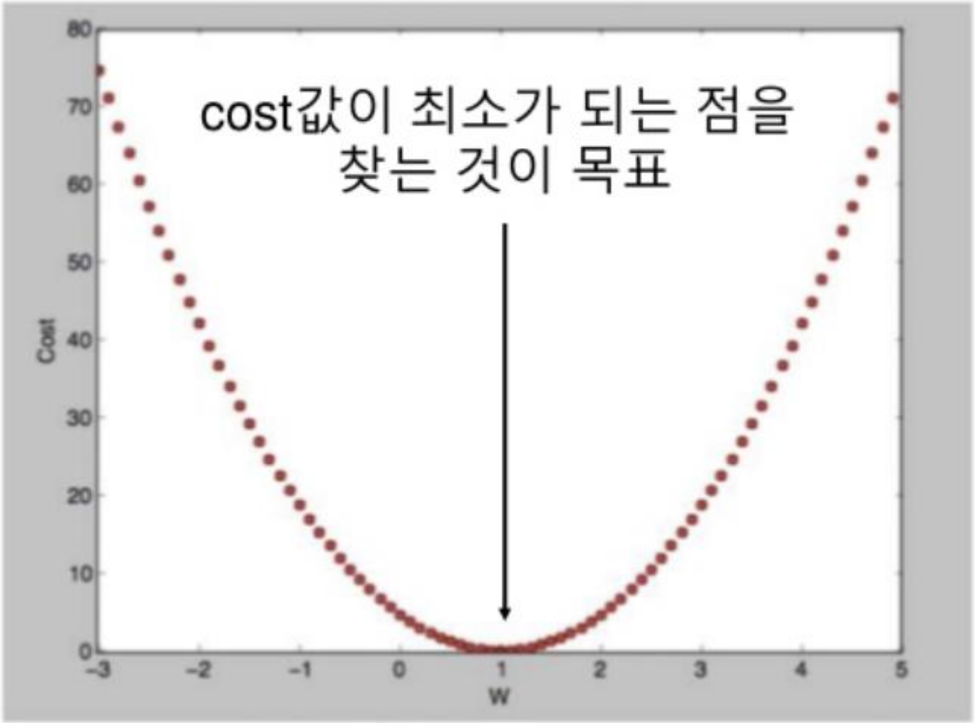
랜덤 임의의 시작점 $-\left.\frac{\partial E}{\partial w}\right|_{w=w^0}$



기울기가 0인 곳에 도달할 때까지 반복



Gradient Descent Algorithm



(e.g. MSE)

$$H(x) = Wx$$

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

무작위로 $H(x)$ 을 그어서
$cost(W)$ 가 최소가 되는 점을 찾는다?

↓

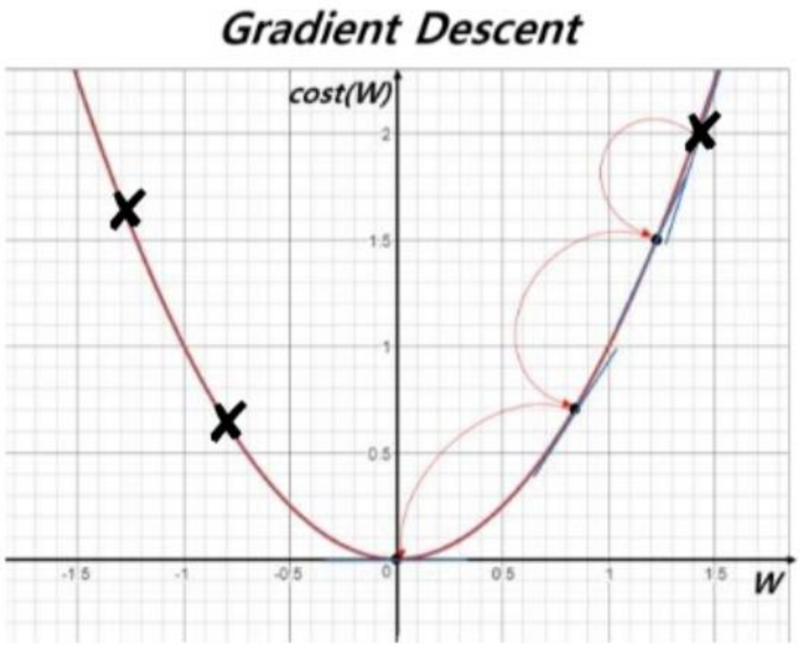
$cost(W)$ 가 최소가 되는 점을
기계적으로 찾아내야 함

↓ optimization

Gradient descent algorithm

그림출처 : 모두의 딥러닝,김성훈 : https://youtu.be/TxIVr-nk1so

Gradient Descent Algorithm



step size, learning rate
= 수렴 속도 조절

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}$$

$\frac{\partial}{\partial W} cost(W)$: 가파른 정도(slope)와 방향

✗ = 시작점

1. 시작점의 경사도를 따라서 조금 이동

2. 이동된 위치의 경사도를 따라서 조금 이동

3. $cost(W)$ 이 최소인 지점까지 반복

Gradient Descent Algorithm

- **Setting the learning rate**

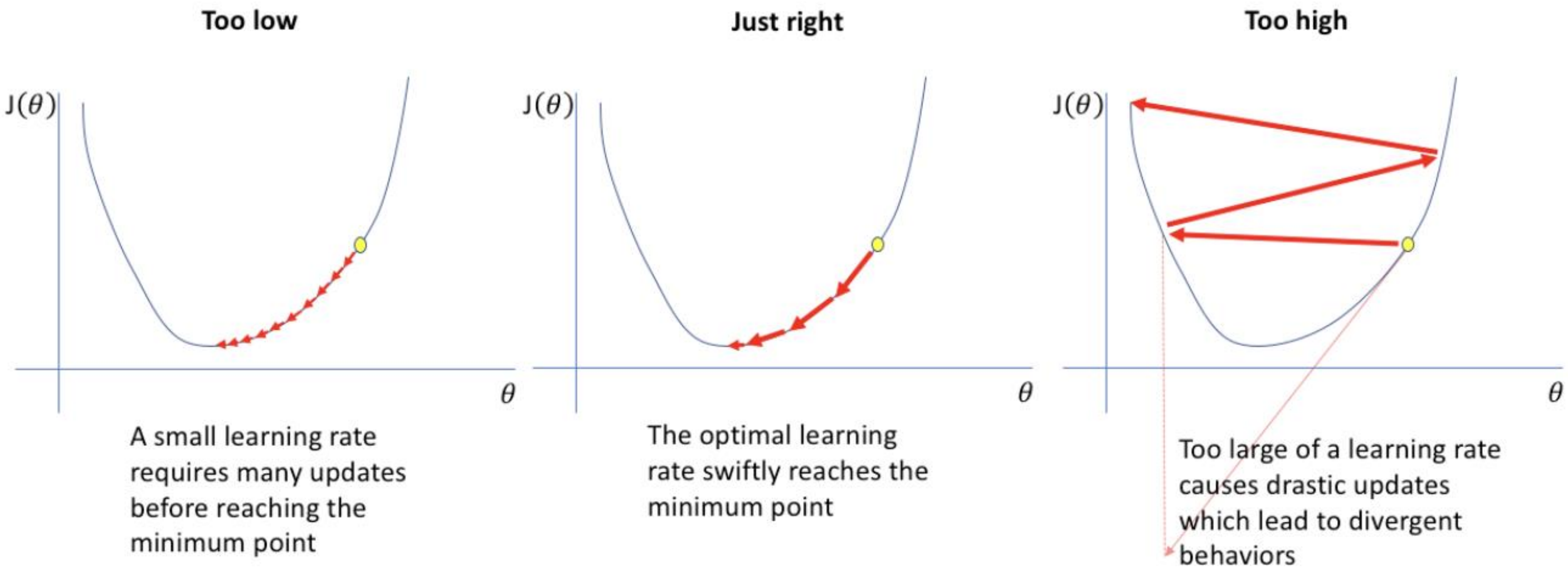


그림 출처 : https://www.jeremyjordan.me/nn-learning-rate/

Gradient Descent Algorithm

- **Minimizing Cost**

$$E(w_0, w_1) = 2{w_1}^2 + 3{w_0}^2 - 6w_1 - 6w_0 + 4w_1w_0 + 5$$

$$\frac{\partial E}{\partial w_1} = 4w_1 + 4w_0 - 6$$

$$\frac{\partial E}{\partial w_0} = 4w_1 + 6w_0 - 6$$

$$w^{t+1} = w^t - \eta \left.\frac{\partial E}{\partial w}\right|_{w=w^t}$$

$$w_0^{t+1} = w_0^t - \eta(4w_1^t + 6w_0^t - 6)$$

$$w_1^{t+1} = w_1^t - \eta(4w_1^t + 4w_0^t - 6)$$

랜덤 임의의 시작점

$$w_0^0 = 1 \ , \ w_1^0 = 1$$

$$w_0^0 = 1$$
$$w_1^0 = 1$$

$$w_0^1 = 1 - 0.1(4 \times 1 + 6 \times 1 - 6) = 0.6$$
$$w_1^1 = 1 - 0.1(4 \times 1 + 4 \times 1 - 6) = 0.8$$

$$w_0^2 = 0.6 - 0.1(4 \times 0.8 + 6 \times 0.6 - 6) = 0.54$$
$$w_1^2 = 0.8 - 0.1(4 \times 0.8 + 4 \times 0.6 - 6) = 0.84$$

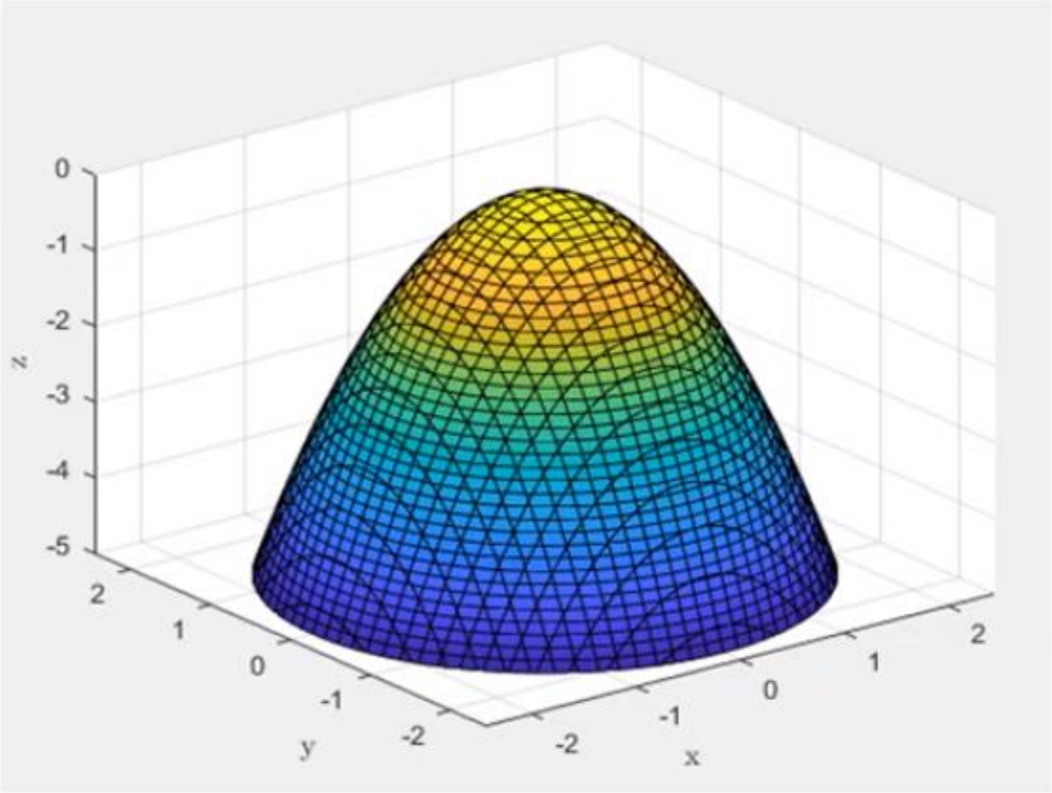
$$w_0^3 = 0.54 - 0.1(4 \times 0.84 + 6 \times 0.54 - 6) = 0.480$$
$$w_1^3 = 0.84 - 0.1(4 \times 0.84 + 4 \times 0.54 - 6) = 0.888$$

…

$$w_0^{100} = 0.00007713$$
$$w_1^{100} = 1.49989171$$

Gradient Descent Algorithm

Convex function



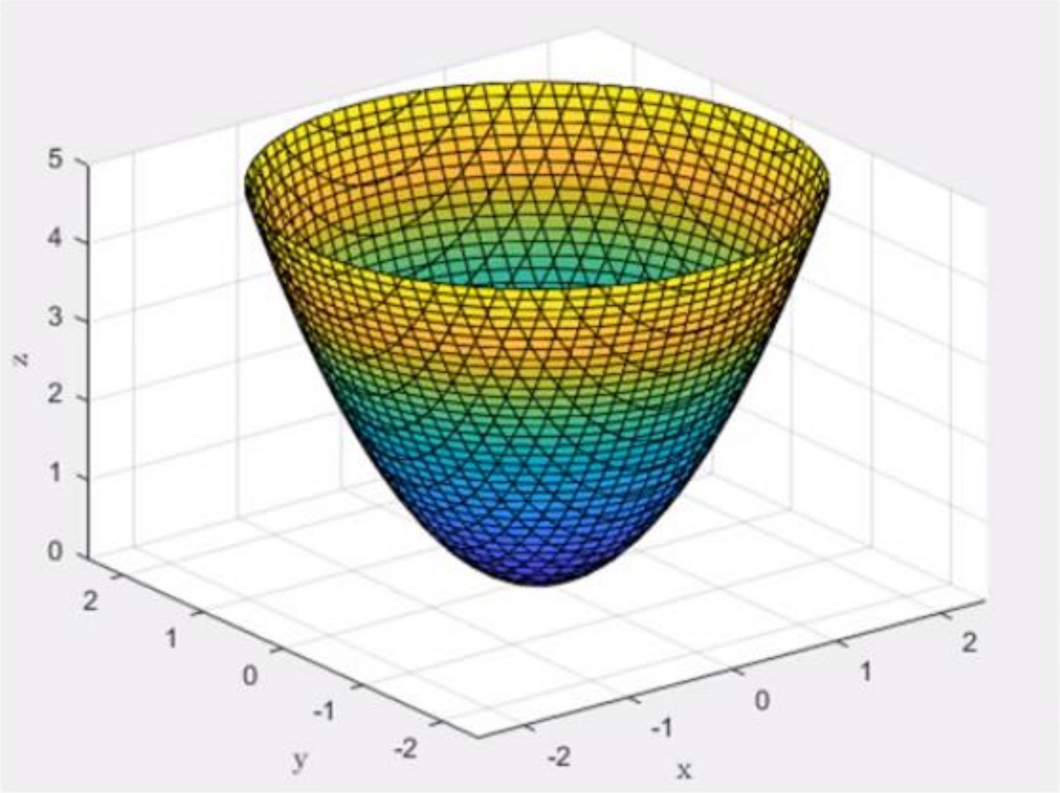


그림 출처 : https://*www.youtube.com/watch?v=8vK18nqsflY*

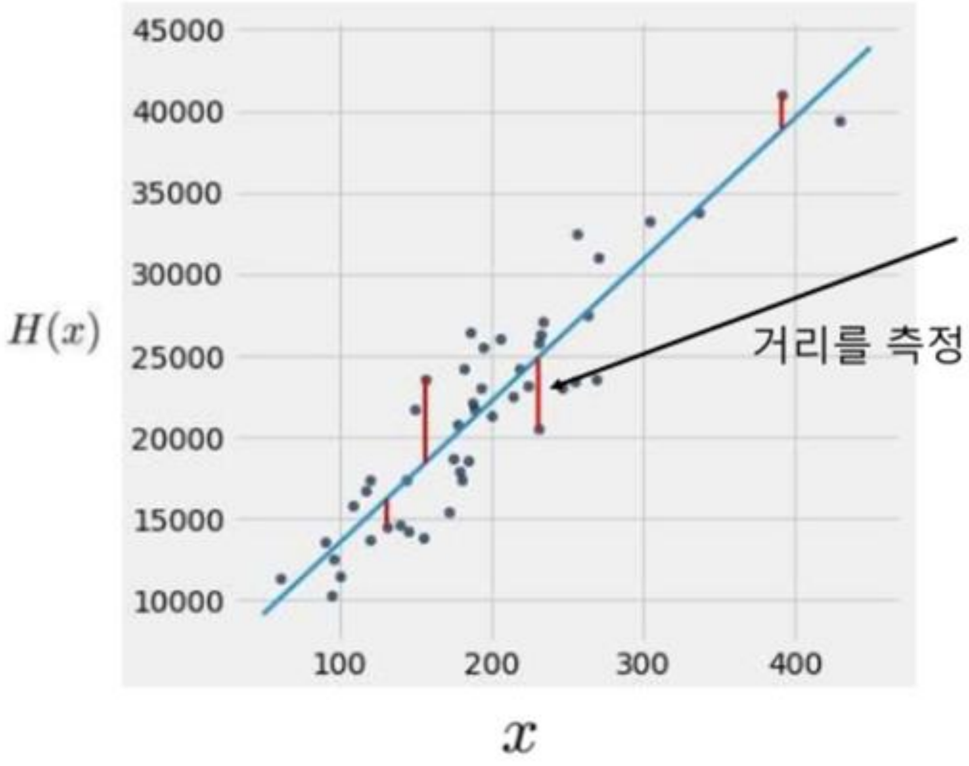
Regression (회귀)

- 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링
  - $y = (w_1 * x_1 + w_1 * x_1 + \cdots + w_n * x_n)$

  - y : 종속 변수(결정 값)
  - x_1 : 독립 변수(피처)
  - w_1 : 회귀 계수(Regression coefficients)

- 최적의 회귀 모델
  - 피처(x)와 결정값(y)를 학습해 최적의 회귀 계수(w)를 찾는 것
    → 오류(실제 값과 회귀 모델의 차이) 합을 최소화
  - 일반 선형 회귀
  - 규제 적용 모델: 릿지(Ridge), 라쏘(Lasso), 엘라스틱넷(ElasticNet)
  - 로지스틱 회귀(Logistic Regression)

Linear Regression - Cost Function



$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$\underset{W,b}{\text{minimize}}\, cost(W, b)$$

- 오류
  - 실제 값과 회귀 모델의 차이
  - 남은 오류의 의미로 잔차 라고도 함
- 비용(cost)/손실(loss) 함수
  - 오류 합을 구하는 함수
- 최적의 회귀 모델
  - 오류 합을 최소화하는 회귀 계수를 갖는 함수

cost 함수

↓

W , b 의 함수

↓

Cost값을 작게 가지는 W , b 를 학습
= linear regression 의 학습

Linear Regression - Cost Function

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

x	Y
1	1
2	2
3	3

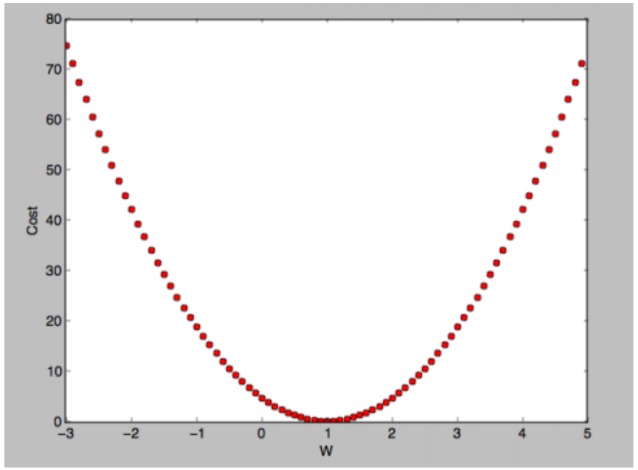
- W=1, cost(W)=0

$$\frac{1}{3}((1*1-1)^2 + (1*2-2)^2 + (1*3-3)^2)$$

- W=0, cost(W)=4.67

$$\frac{1}{3}((0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2)$$

- W=2, cost(W)=?



How to minimize cost?

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

그림출처 : 모두의 딥러닝,김성훈 : https://youtu.be/TxIVr-nk1so

Multi-variable linear regression

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

$$H(x_1, x_2, x_3) = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$cost(W, b) = \frac{1}{m} \sum_{I=1}^{m} (H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

$$H(x_1, x_2, x_3, \ldots, x_n) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_n x_n + b$$

$$H(X) = XW$$

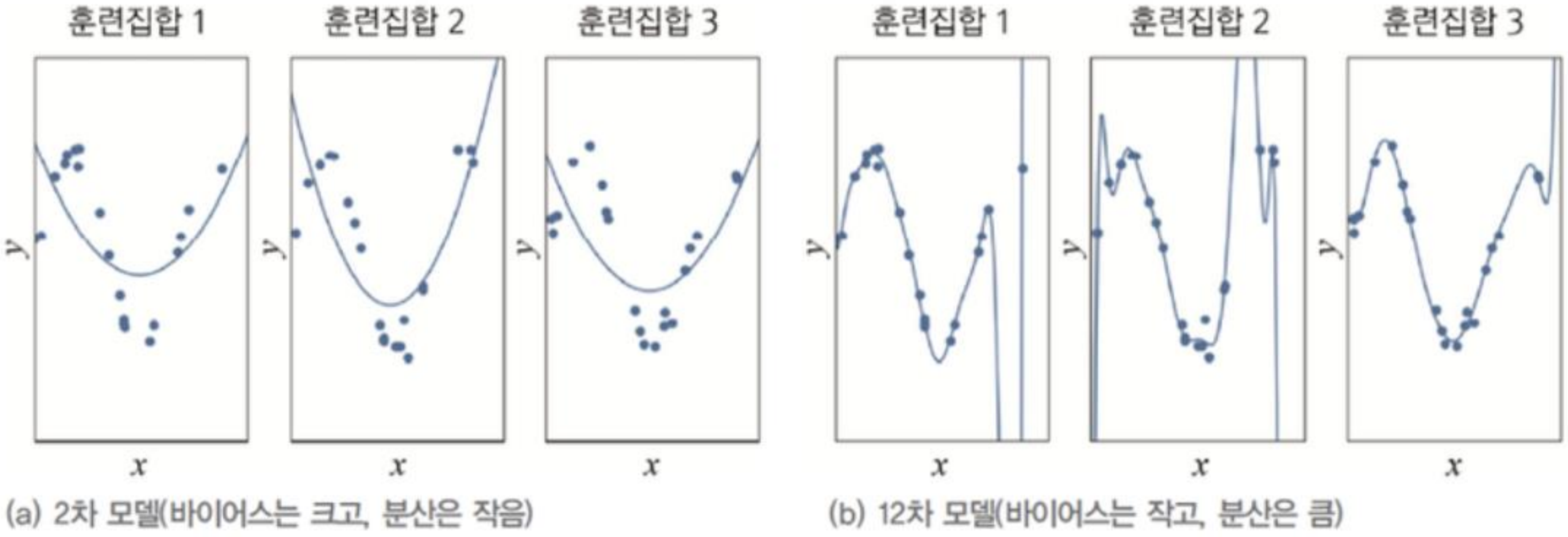
$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

[5, 3] [3,1] [5,1]

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

Multi-variable linear regression

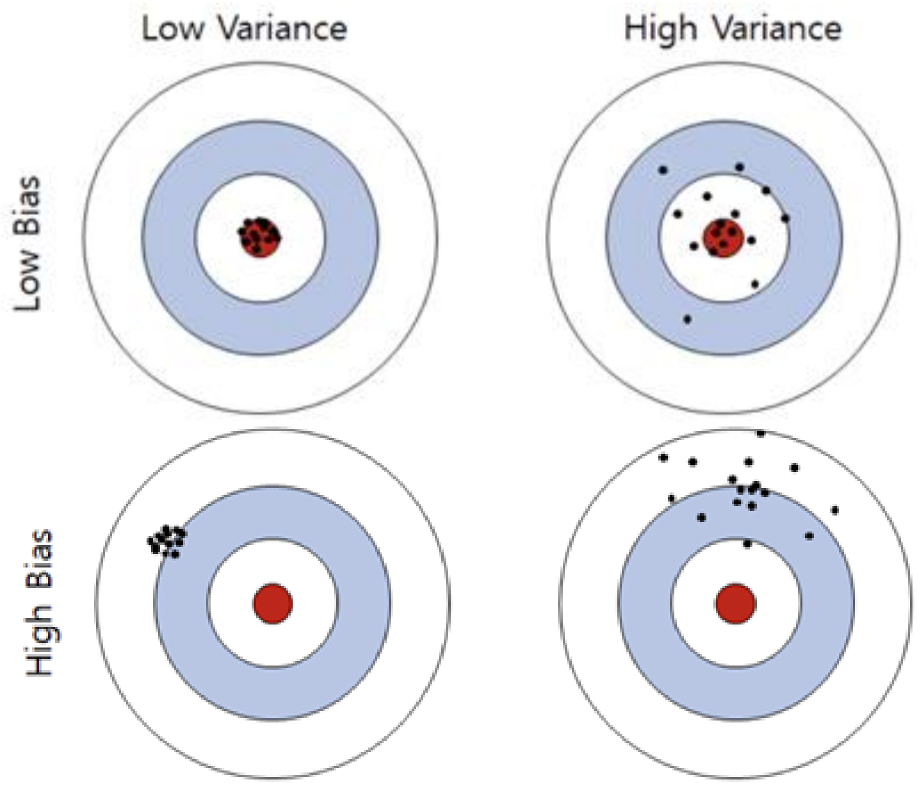
- 회귀가 독립 변수에 대한 다항식으로 표현
- sklearn.preprocessing.PolynomialFeatures : 다항식 피처 변환
  - 차수에 따른 과소적합/과적합



그림출처: 도서 machine learning/오일석

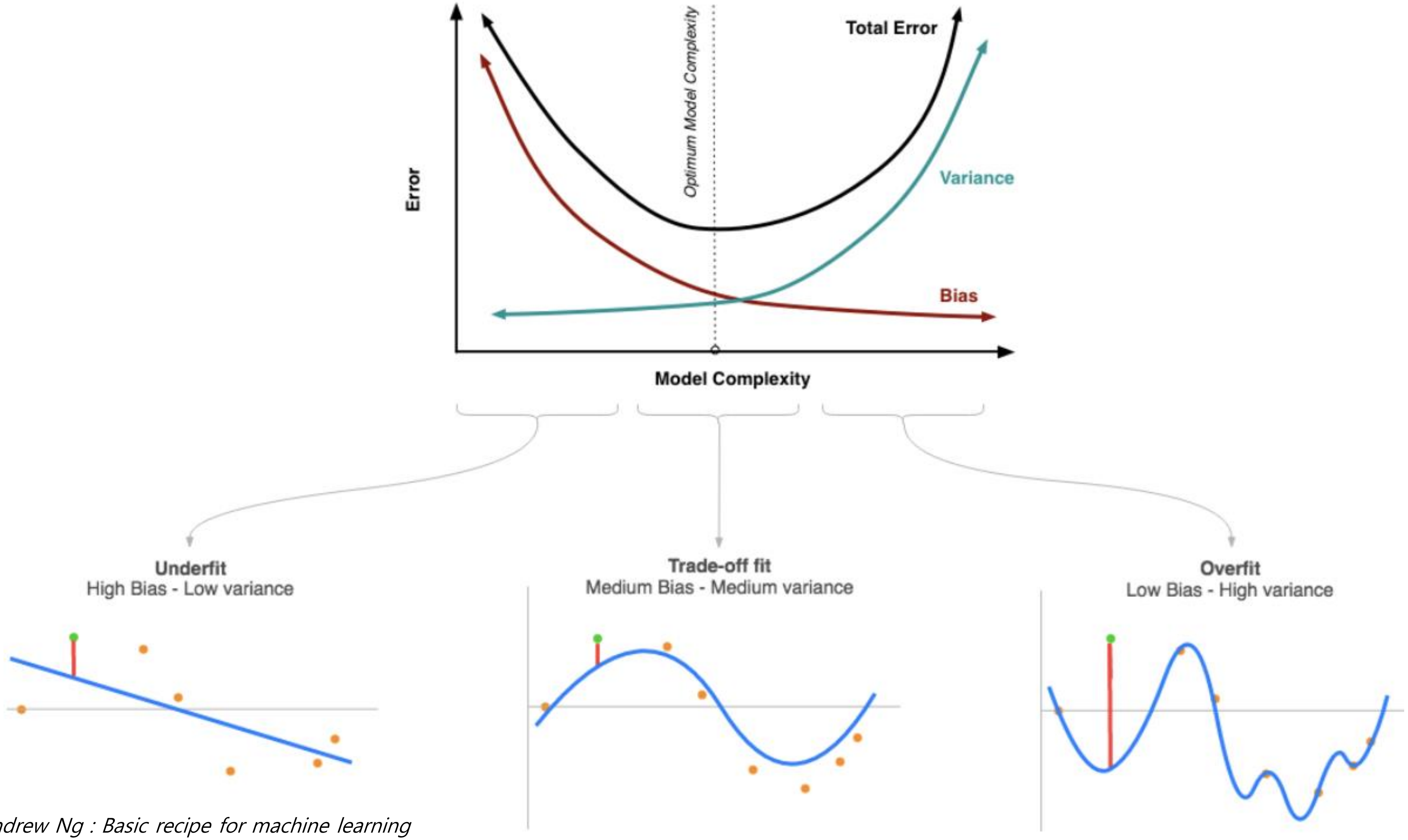
편향-분산(Bias-Variance) trade-off

- 고편향 : 한 방향성으로 치우침
- 고분산 : 지나치게 높은 변동성
- 편향↑, 분산↓: Underfitting
- 편향↓, 높은↑: Overfitting



그림출처 : Coursera Andrew Ng : Basic recipe for machine learning

편향-분산(Bias-Variance) trade-off



그림출처 : Coursera Andrew Ng : Basic recipe for machine learning

Regularized Linear Regression

- **과대적합을 감소시키기 위해 모델의 가중치(weight)를 규제(제한)**
- **L1 :** 가중치 벡터의 ℓ1norm 사용

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha\sum_{i=1}^{n} |\theta_i|$$

- **L2 :** 가중치 벡터의 **(**ℓ2norm^2) / 2 사용

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

- **Elastic-Net :** 두 정규화 항을 합쳐서 r로 규제 정도를 조절

$$J(\theta) = \mathrm{MSE}(\theta) + r\alpha\sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha\sum_{i=1}^{n}\theta_i^2$$

※ r=0 릿지, r=1 라쏘

그림출처 : 파이썬 머신러닝 완벽 가이드

Regularized Linear Regression

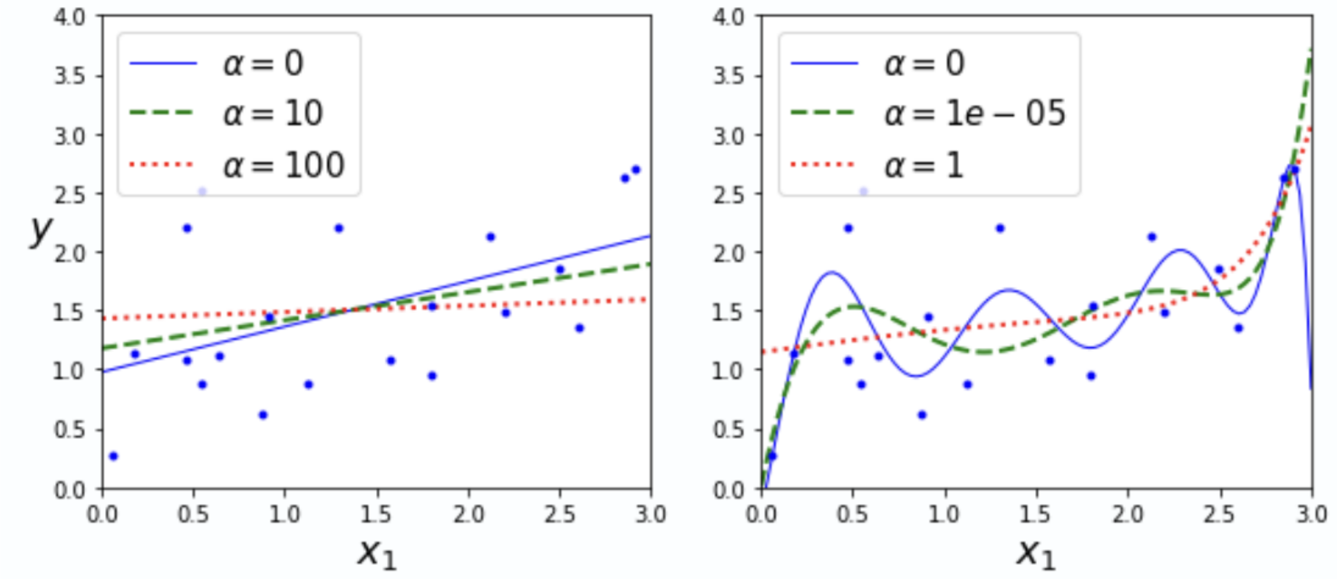
- Ridge
  - L2-Norm을 사용한 회귀 모델
  - 영향을 미치지 않는 피쳐 가중치를 0에 가깝게 만듦
  - 하이퍼파라미터 a(alpha)는 모델을 얼마나 규제할지 조절
    : alpha 값이 커지면 회귀 계수 W의 값을 작게 해 과적합 개선
    : a = 0 , 선형회귀와 같음
    : a = 큰 값 , 모든 가중치가 거의 0에 가까워짐

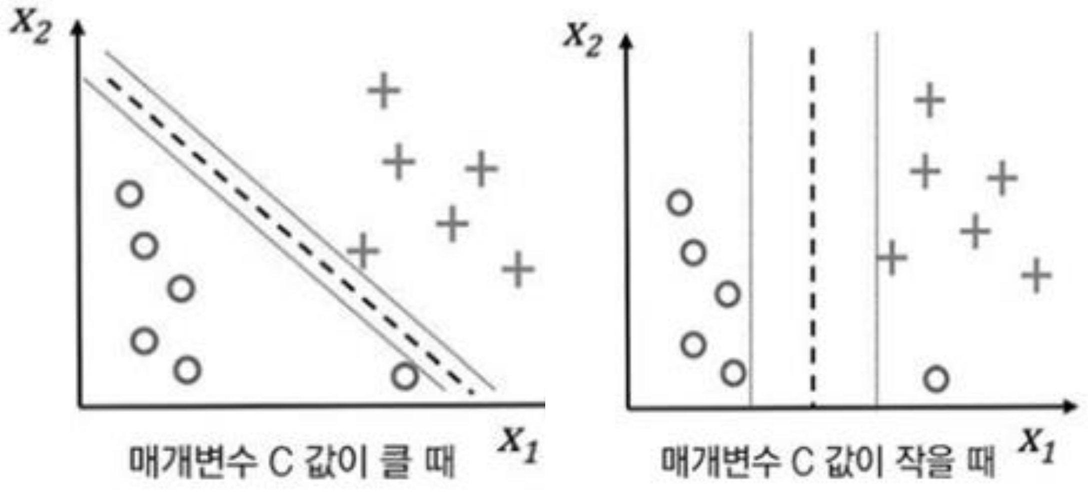
$$Min(RSS(W) + alpha * ||W||_2^2)$$

- Lasso
  - L1-Norm을 사용한 회귀 모델
  - 덜 중요한 변수의 가중치를 완전히 제거 → 피쳐선택의 효과

- ElasticNet
  - Ridge + Lasso

Regularized Linear Regression





α 증가 : 직선에 가까워 짐(편향↑, 분산↓: Underfitting)

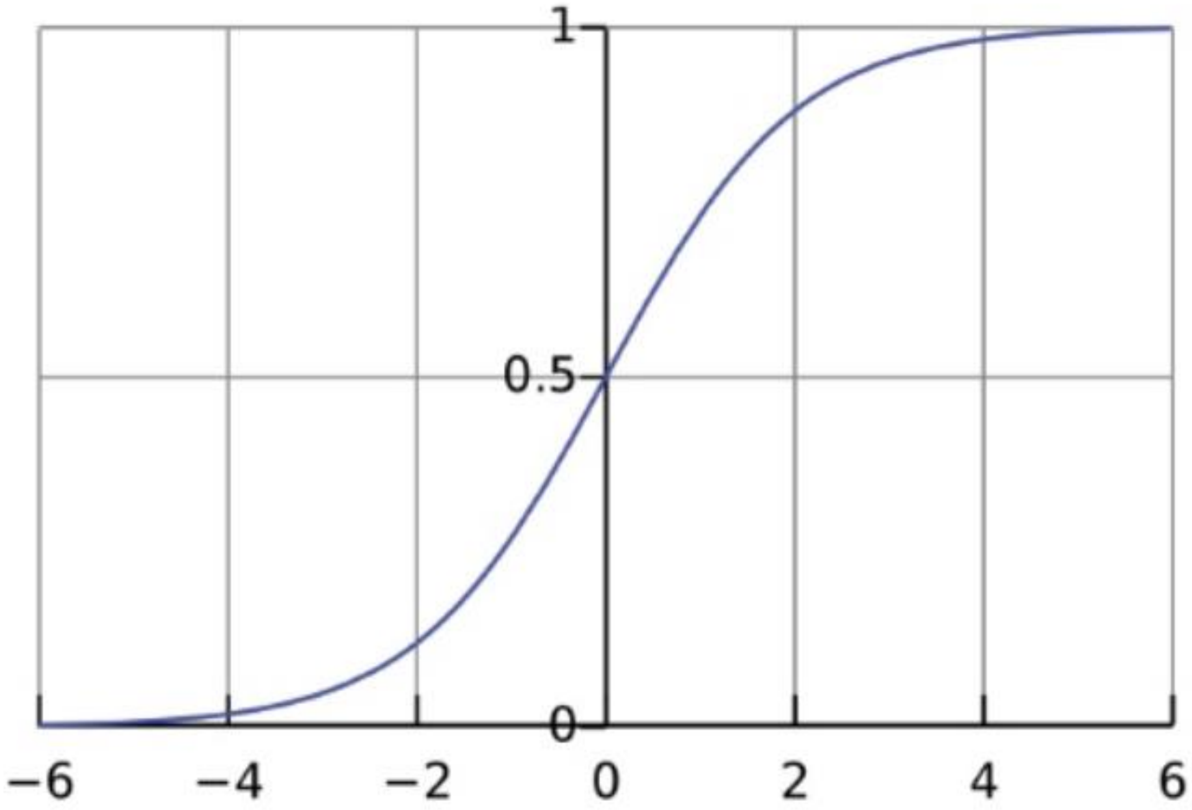
규제가 있는 로지스틱 회귀 모델의 경우 : C값으로 조절(C↓, 편향↑, 분산↓)

[kaggle] bike-sharing demand prediction

- https://www.kaggle.com/c/bike-sharing-demand
  - 피처 타입, Null 데이터 확인
  - Null이 많은 피처는 드롭(숫자형 피처는 평균값 대체)
  - 대여일자 datetime 타입 처리
  - 카테고리 값 인코딩 처리
  - 왜곡된 타킷 값 로그 변환 처리
  - 문자형 피처 : 원-핫 인코딩
  - 회귀 계수가 높은 피처는 이상치 데이터 확인 후 제거

```
from scipy.stats import skew
features_index = house_df.dtypes[house_df.dtypes != 'object'].index
        skew_features = house_df[features_index].apply(lambda x : skew(x)) #
        skew_features_top = skew_features[skew_features > 1]
        print(skew_features_top.sort_values(ascending=False))
house_df[skew_features_top.index] = np.log1p(house_df[skew_features_top.index])
```

Cost Function – Logistic Classification



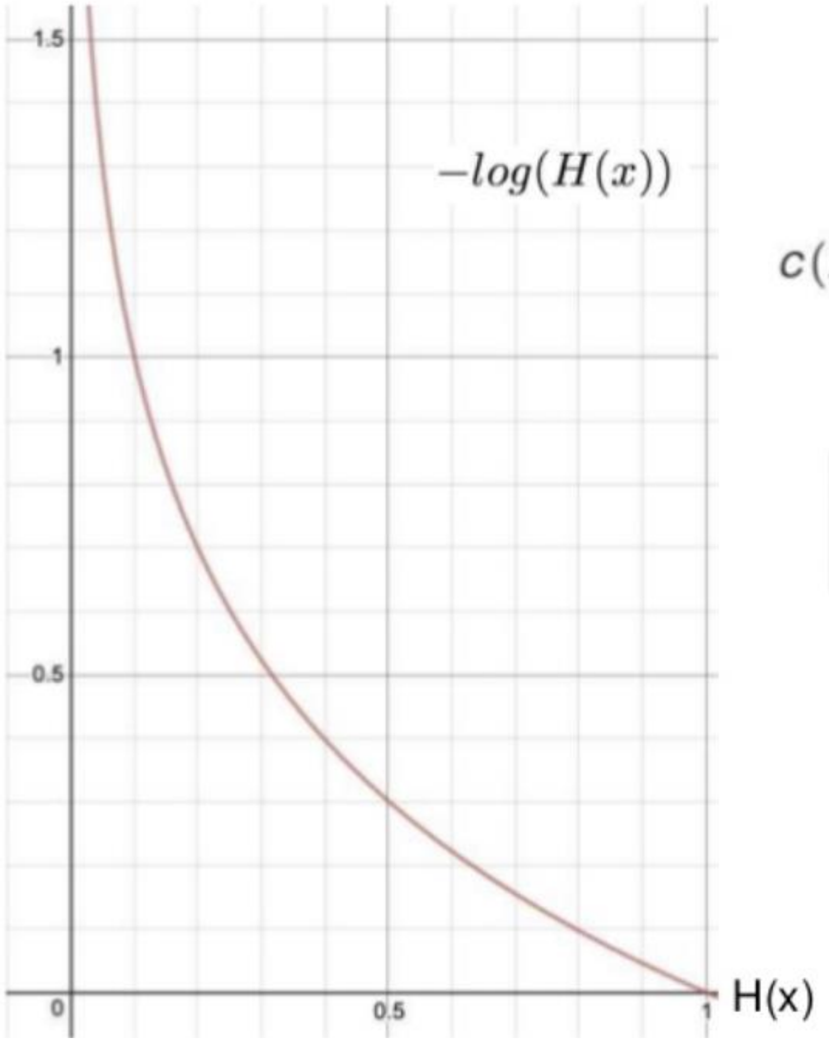
linear hypothesis

(e.g. logistic hypothesis = sigmoid function)

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

(WX = linear hypothesis)

Cost Function – Logistic Classification



$$cost(W) = \frac{1}{m}\sum \quad c(H(x), y)$$

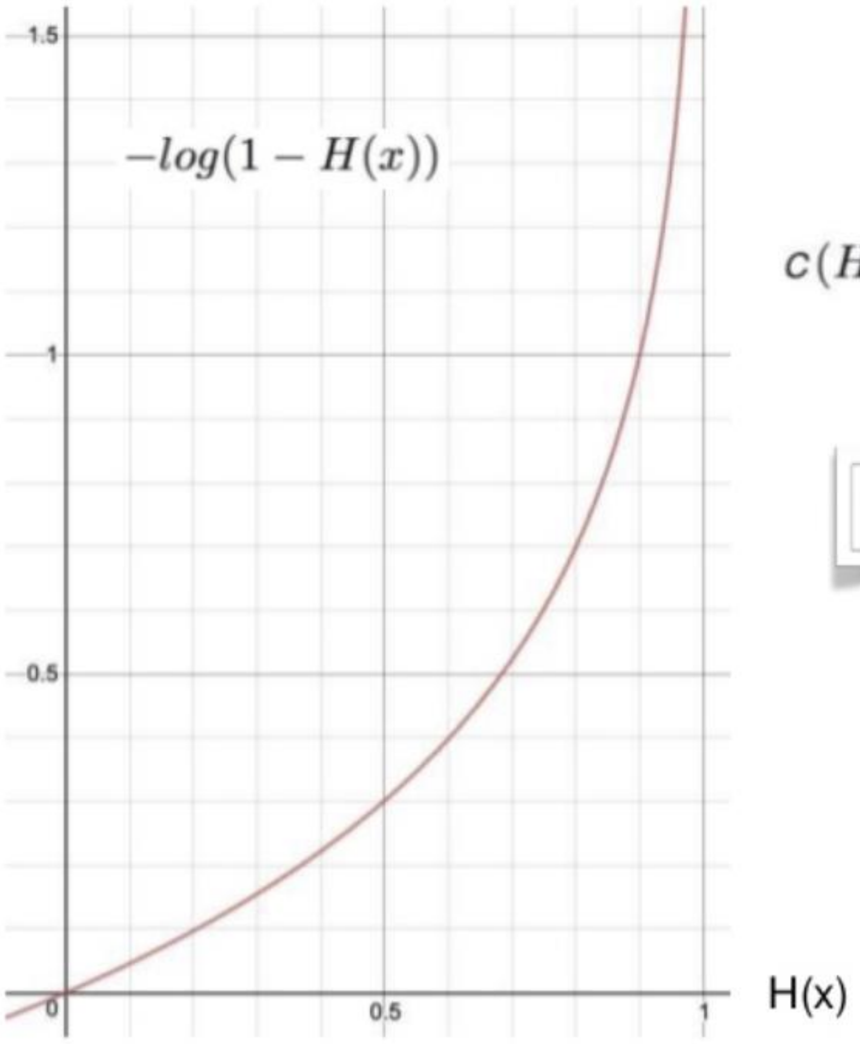
$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$, y = 실제값

$$\boxed{y = 1}$$

$H(x) = 1 \longrightarrow Cost(1,1) = 0$

$H(x) = 0 \longrightarrow Cost(0,1) = \infty$

Cost Function – Logistic Classification



$$cost(W) = \frac{1}{m} \sum \quad c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$, y = 실제값

$$\boxed{y = 0}$$

H(x) = 1 ⟶ Cost(1,0) = ∞

H(x) = 0 ⟶ Cost(0,0) = 0

Cost Function – Logistic Classification

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

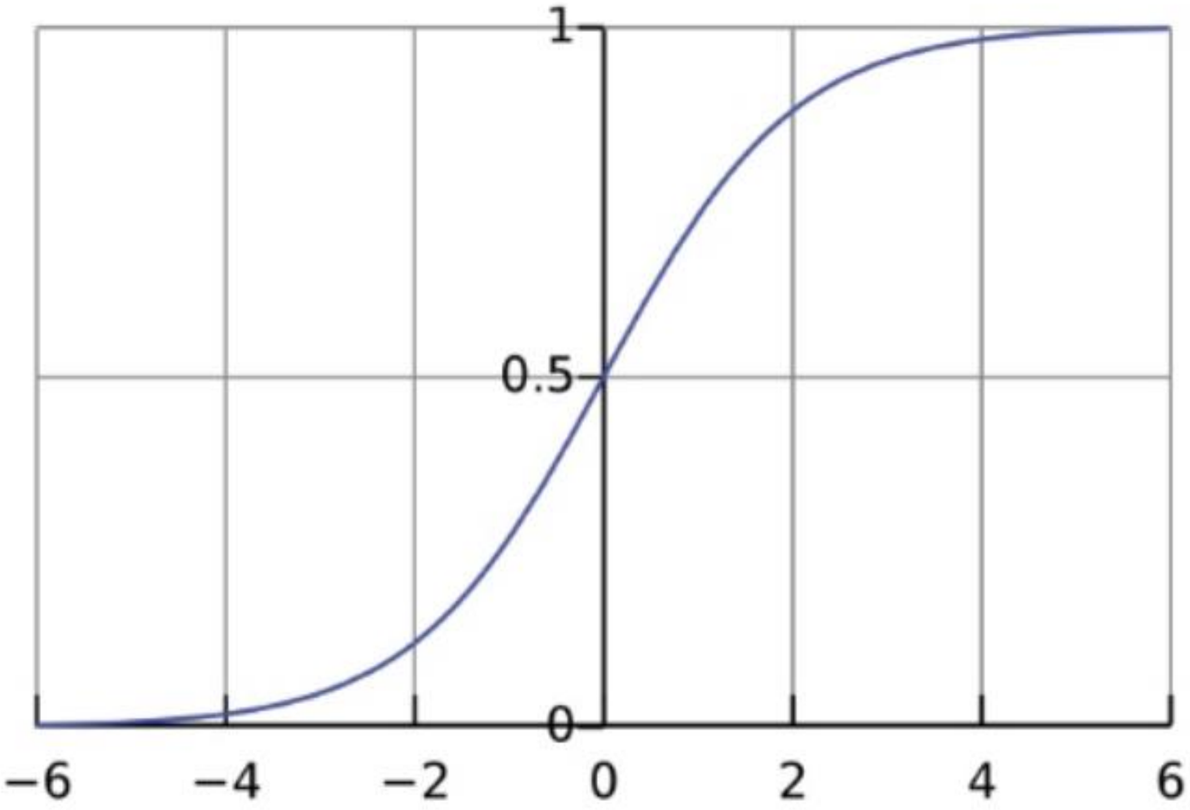
$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

$$cost(W) = -\frac{1}{m} \sum ylog(H(x)) + (1 - y)log(1 - H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

Cost Function – Logistic Classification



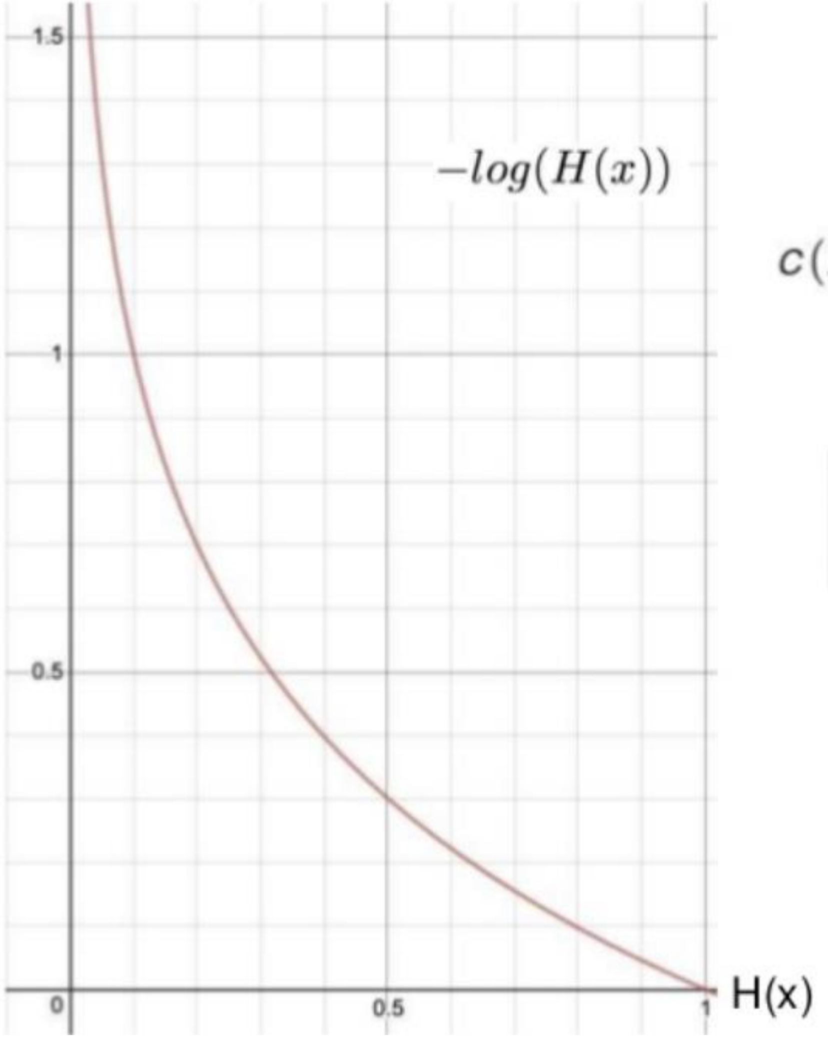
linear hypothesis

(e.g. logistic hypothesis = sigmoid function)

$$H(X) = \frac{1}{1+e^{-W^T X}}$$

(WX = linear hypothesis)

Cost Function – Logistic Classification



$$cost(W) = \frac{1}{m}\sum c(H(x), y)$$

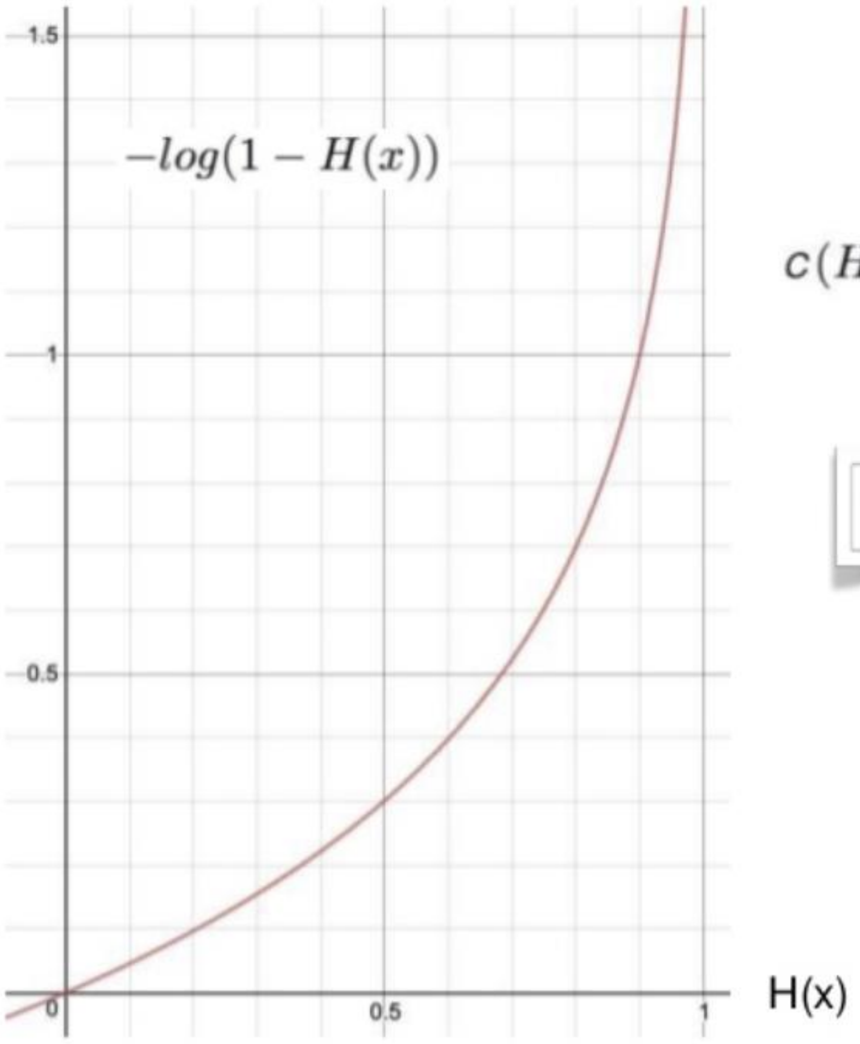
$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$, y = 실제값

$$\boxed{y = 1}$$

$H(x) = 1 \longrightarrow Cost(1,1) = 0$

$H(x) = 0 \longrightarrow Cost(0,1) = \infty$

Cost Function – Logistic Classification



$$cost(W) = \frac{1}{m} \sum \quad c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$, y = 실제값

$$\boxed{y = 0}$$

$H(x) = 1 \longrightarrow Cost(1,0) = \infty$

$H(x) = 0 \longrightarrow Cost(0,0) = 0$

Cost Function – Logistic Classification

KOSA 한국소프트웨어산업협회

$$c(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

$$cost(W) = \frac{1}{m} \sum c(H(x), y)$$

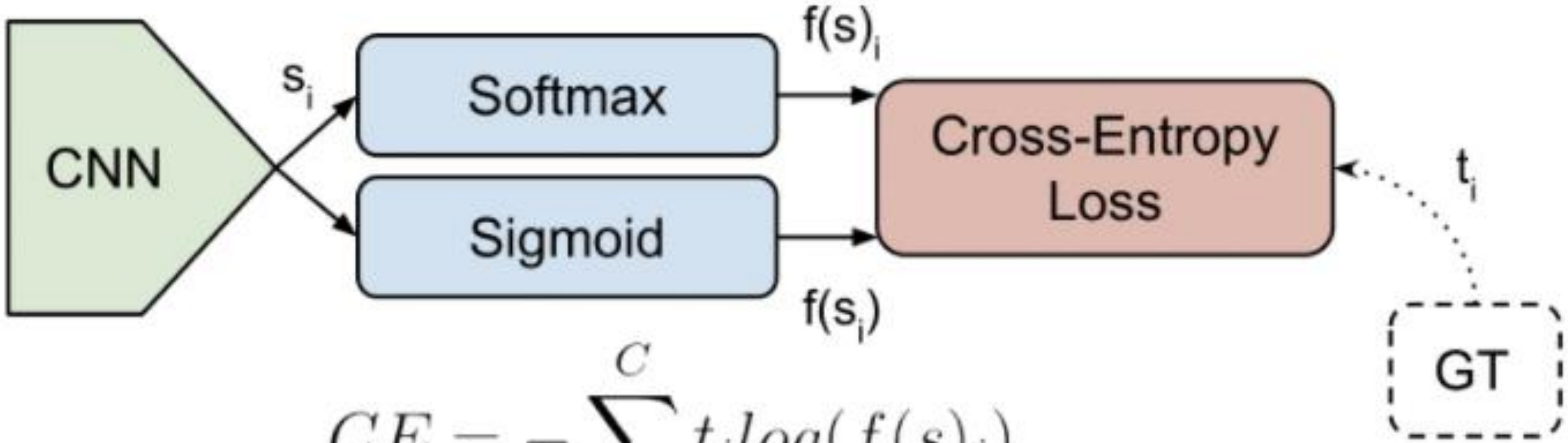
$$c(H(x), y) = -ylog(H(x)) - (1 - y)log(1 - H(x))$$

$$cost(W) = -\frac{1}{m} \sum ylog(H(x)) + (1 - y)log(1 - H(x))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

Categorical/Binary CrossEntropy - Softmax/Logistic Loss

- **Task**
  - **Multi-Class Classification , Multi-Label Classification**



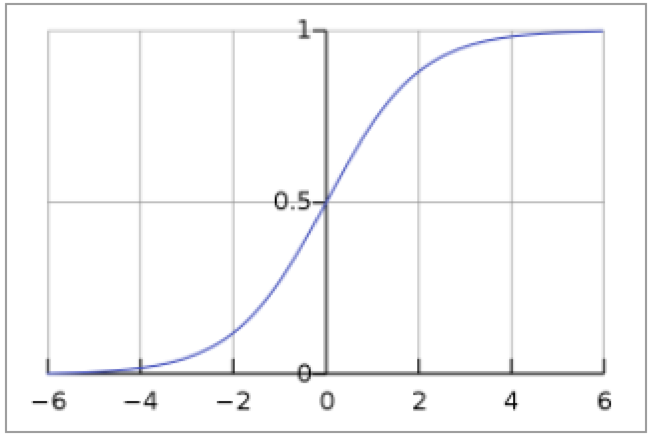
$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

[그림출처] https://gombru.github.io/2018/05/23/cross_entropy_loss/

KOSA 한국소프트웨어산업협회

- **Output Activation Functions**
  - Sigmoid vs Somtmax



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

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

- **Losses**
  - Cross-Entropy Loss
  - binary classification problem

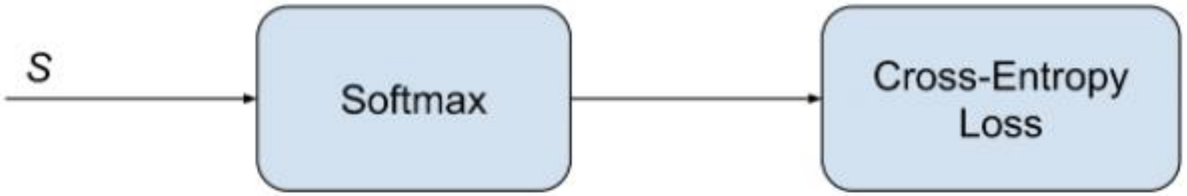
$$CE = -\sum_i^C t_i log(s_i)$$

Where t_i and s_i are the groundtruth and the CNN score for each class i in C. As **usually an activation function (Sigmoid / Softmax) is applied to the scores before the CE Loss computation**, we write $f(s_i)$ to refer to the activations.

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1)$$

Categorical/Binary CrossEntropy - Softmax/Logistic Loss

- **Categorical Cross-Entropy loss**



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

- **Binary Cross-Entropy Loss**

$$CE = \begin{cases} -log(f(s_1)) & if \quad t_1 = 1 \\ -log(1 - f(s_1)) & if \quad t_1 = 0 \end{cases}$$

$$CE = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

$$f(s_i) = \frac{1}{1 + e^{-s_i}} \qquad CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$