# 학습(Learning)

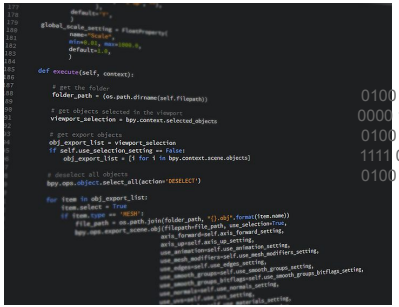
# 전통적인 프로그래밍

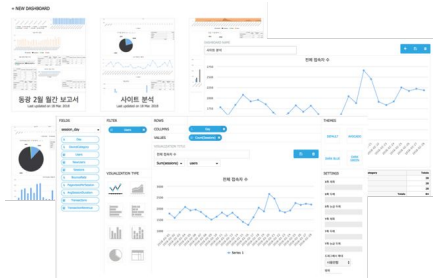
기계(혹은 컴퓨터)를 실행하기 위해서 기계가 이해할 수 있는 프로그래밍 언어로 명령을 내리고, 그 결과를 사용자에게 전달

프로그래밍 언어
(Programming Language)

기계
(Computing Machine)

사용자
(User)

# 머신러닝

데이터로 부터 학습하도록 컴퓨터를 프로그래밍 하는 과학

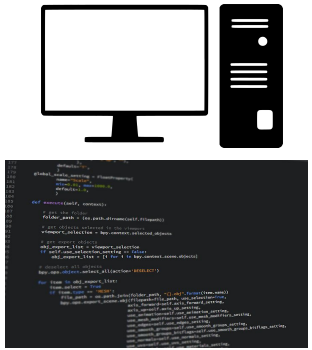
명시적인 프로그래밍 없이 컴퓨터가 스스로 학습하는 능력을 갖게 하는 연구 분야 (Arthur Samuel)

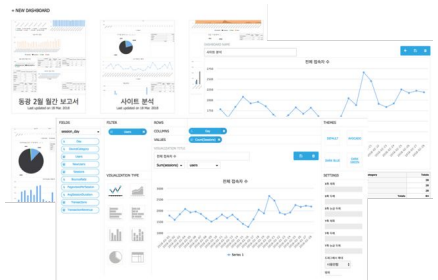
데이터
(Data)

기계
(Computing Machine)

사용자
(User)

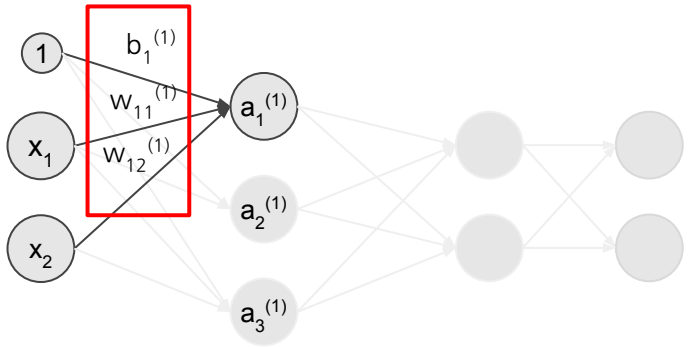
프로그래밍 언어
(Programming Language)



# What?

- 파라미터(매개변수) : 학습의 대상이 되는 변수




$$a_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + b_1^{(1)}$$
$$a_2^{(1)} = w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + b_2^{(1)}$$
$$a_3^{(1)} = w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + b_1^{(1)}$$

$$A^{(1)} = XW^{(1)} + B^{(1)}$$

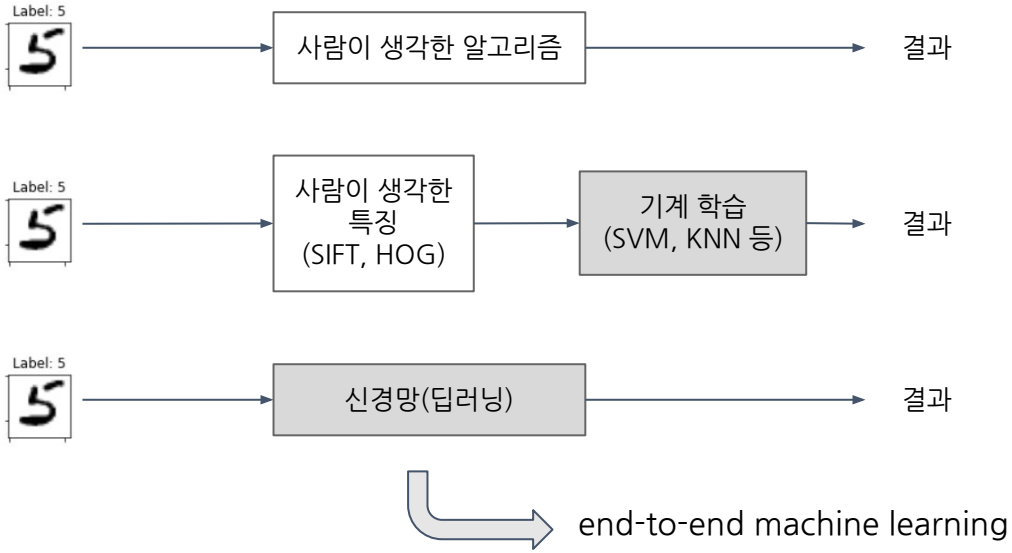
$$A^{(1)} = (a_1^{(1)} \; a_2^{(1)} \; a_3^{(1)}) \;,\; X = (x_1 \; x_2)$$

$$B^{(1)} = (b_1^{(1)} \; b_2^{(1)} \; b_2^{(1)})$$

$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} \end{pmatrix}$$

# How

- 데이터에서 학습한다 ➡️ 기계 학습



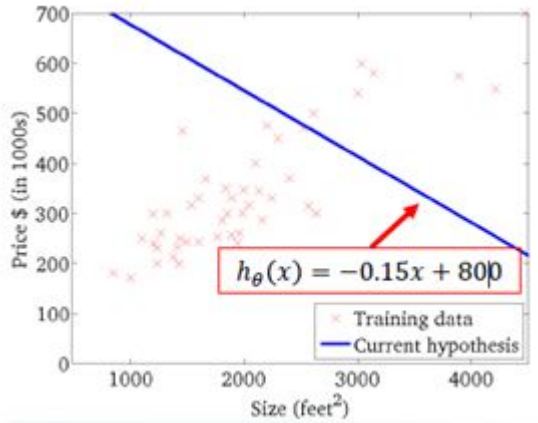
# 딥러닝 학습 방법

- Hypothesis 함수(H) : 머신러닝의 목적이 되는 모델
- Cost 함수(J) : Hypothesis 함수로 인해 찾아지는 예측값과 실제 값의 차이를 Cost라고 한다

➡️ 어떻게 정확한 Hypothesis를 찾아가는가?

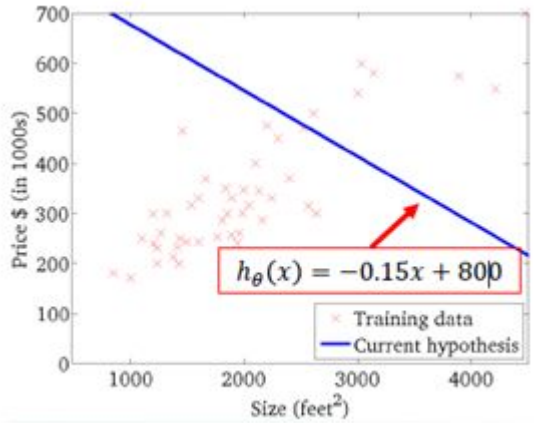
**Hypothesis 함수 정리**

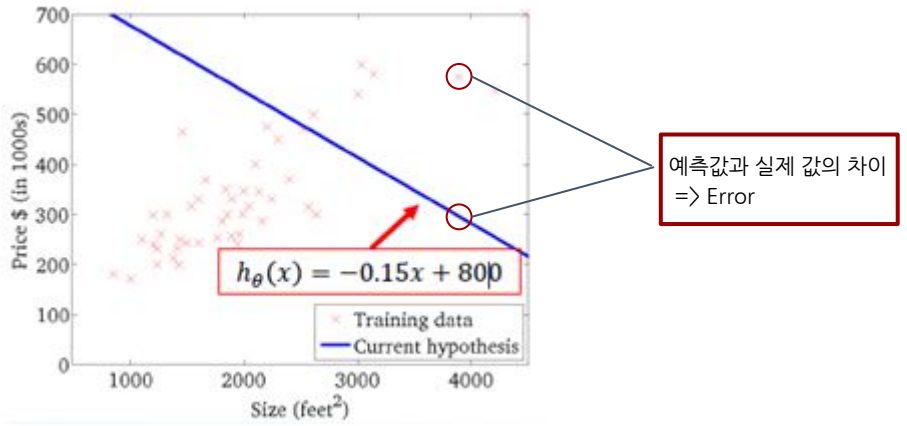
1. Hypothesis 함수는 입력 값 X 가 출력 값 Y에 영향을 미치는 정도를 의미하는 Weight 값 W(혹은 $\theta$ )로 이루어져 있다.
2. 출력값 Y는 결국 모델의 예측값이다.
3. 입력값 X는 주어지는 데이터이므로, 정확한 Hypothesis를 찾는다는 말은 Weight 값들을 찾는다는 것을 의미한다



# 정확한 Hypothesis를 찾는 과정

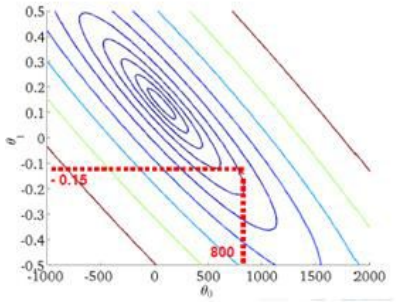
1. Hypothesis 함수를 구성하는 weight 값들에 임의의 초기값을 대입하여 첫번째 Hypothesis를 찾는다
2. 찾은 첫 번째 Hypothesis에 의한 예측값과 실제 데이터의 값의 차이를 계산한다

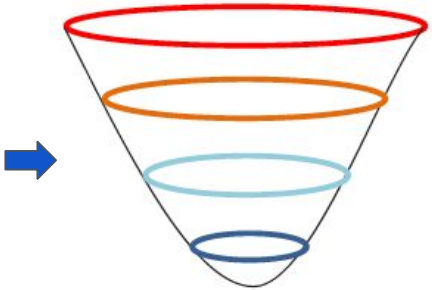


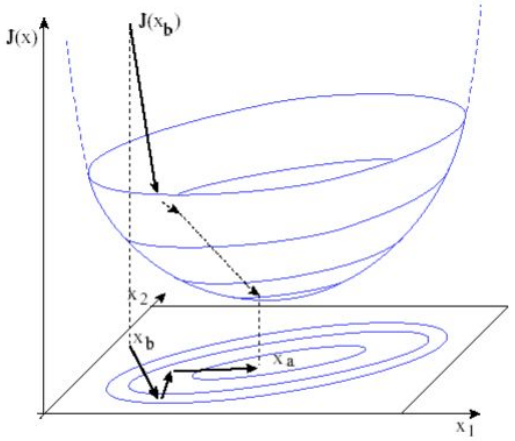



# Cost 함수

1. Cost 함수 : $(error)^2$ 값의 평균 (RSS : Residual Sum of Square)
2. Hypothesis 함수에 따라, 즉 Weight 값들에 따라 Cost는 달라진다. Weight 값들에 따른 Cost값들의 집합이 Cost 함수이다.





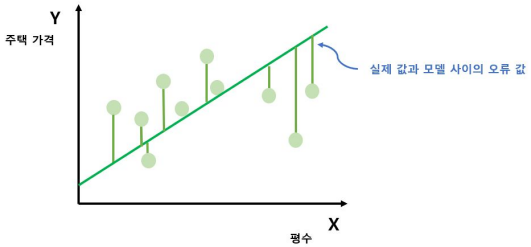
# Cost 함수

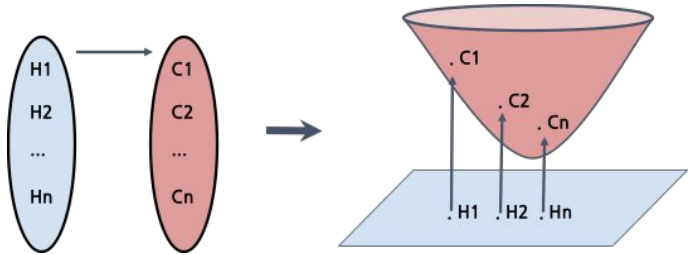
- 실제 값과 모델 사이의 오류 값(잔차)들의 합을 최소화

  = Cost 함수를 최소화

  = 최적의 회귀 계수를 찾는 과정

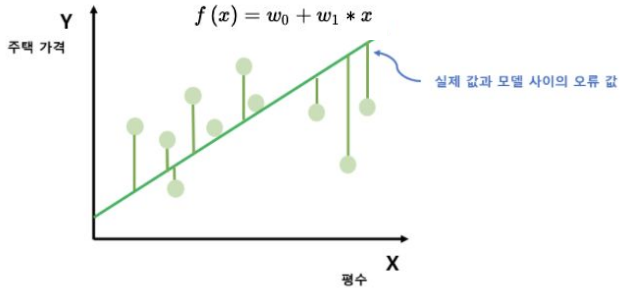
- 각각의 Hypothesis는 자신의 Cost를 가지고 있다



# 회귀에서 Cost 함수

- RSS (Residual Sum of Square) : 오류 값의 제곱을 구해서 더하는 방식



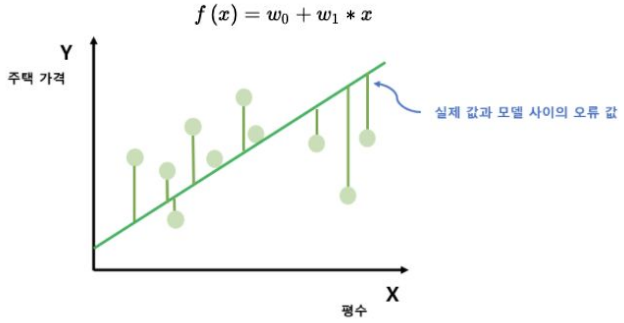

RSS = (#1 주택가격 -($w_0$+$w_1$*#1주택 크기))$^2$
+ (#2 주택가격 -($w_0$+$w_1$*#2주택 크기))$^2$
+ (#3 주택가격 -($w_0$+$w_1$*#3주택 크기))$^2$
+ …
+ (#n 주택가격 -($w_0$+$w_1$*#n주택 크기))$^2$

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$

↳ w변수(회귀 계수) 의 함수

# Cost 함수 - 오차제곱합

- SSE (Sum of Square Error) : 오류 값의 제곱을 구해서 더하는 방식
- 개념은 같은데 표현만 다르다




SSE = (#1 주택가격 -($w_0$+$w_1$*#1주택 크기))$^2$
+ (#2 주택가격 -($w_0$+$w_1$*#2주택 크기))$^2$
+ (#3 주택가격 -($w_0$+$w_1$*#3주택 크기))$^2$
+ …
+ (#n 주택가격 -($w_0$+$w_1$*#n주택 크기))$^2$

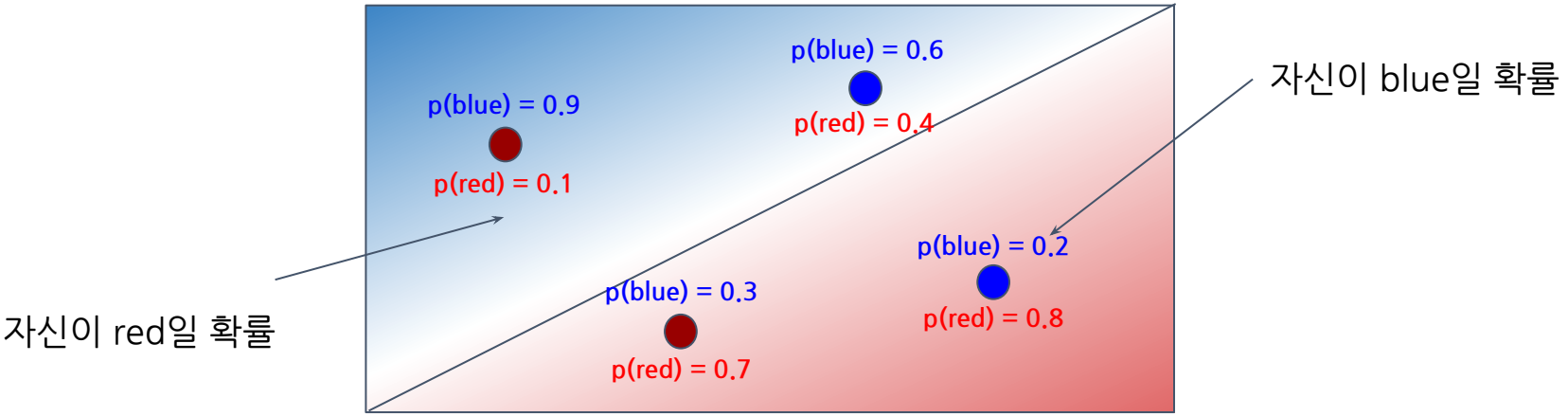
$$E = \frac{1}{2}\sum_{k}\left(y_k - t_k\right)^2$$

# Cost 함수 - 평균 제곱 오차

- MSE (Mean Squared Error) : 오류 값의 제곱을 구해서 더하는 방식

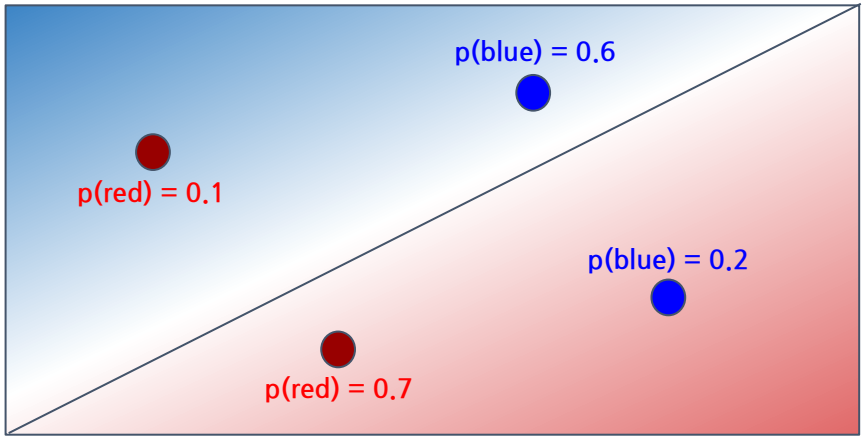
$$E = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - t_i\right)^2$$

# Cost 함수 - 교차 엔트로피

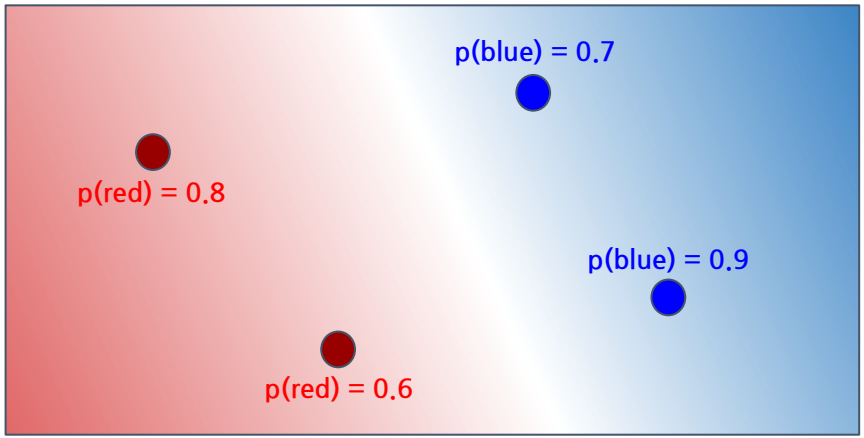


# Maximum Likelihood

- 올바르게 구별이 될 확률 : 모든 데이터의 올바르게 구별 될 확률의 합




0.6 * 0.2 * 0.1* 0.7 = 0.0084




0.7 * 0.9 * 0.8* 0.6 = 0.3024

# Cost 함수 - 교차 엔트로피

- 정답 확률에 로그를 취한 후 음수(-)를 붙인 값
  - 데이터가 많으면 곱셈의 연산량이 커진다 ➡️ Log
  - 로그를 붙이면 음수로 결과가 나온다 ➡️ minus(-)
- 즉 정답 확률을 최대화 하는 것(maximum likelihood)은 cross entropy를 최소화하는 것과 같고 error 함수를 최소화하는 것과 같다

0.6 * 0.2 * 0.1 * 0.7 = 0.0084

ln(0.6) + ln(0.2) + ln(0.1) + ln(0.7)

-0.51 -1.61 -2.3 -0.36

-ln(0.6) - ln(0.2) - ln(0.1) - ln(0.7)

= 4.78

0.7 * 0.9 * 0.8* 0.6 = 0.3024

ln(0.7) + ln(0.9) + ln(0.8)+ ln(0.6)

-0.36 -0.1 -0.22 -0.51

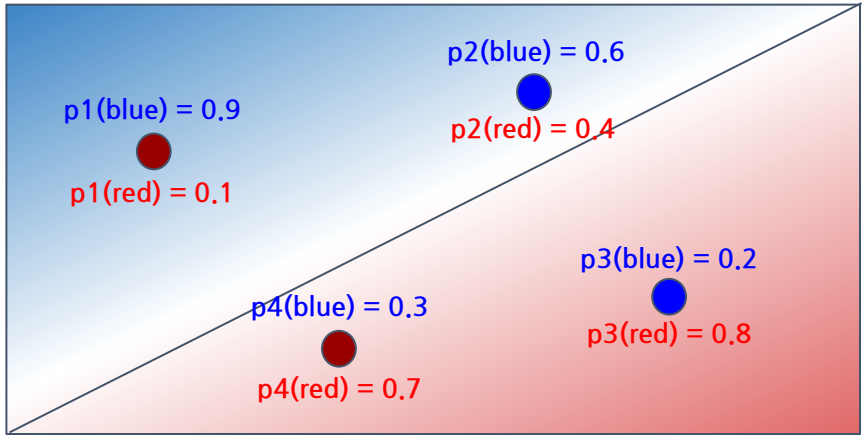
-ln(0.7) - ln(0.9) - ln(0.8) - ln(0.6)

= 1.19

# Cost 함수 - 교차 엔트로피

- 파란색을 1, 빨간색을 0이라고 한다면
- 파란색의 확률은

p1 = 0.9, p2 = 0.6, p3 = 0.2, p4 = 0.3

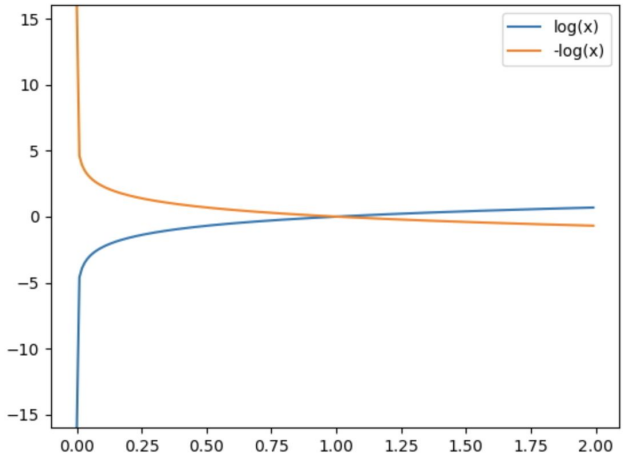



| p1 | p2 | 1 - p3 | 1 - p4 |
|---|---|---|---|
| 0.9 | 0.6 | 0.8 | 0.7 |
| y1 = 1 | y1 = 1 | y1 = 0 | y1 = 0 |

$$-\sum_{i=1}^{m} y_i \ln(p_i) + (1-y_i)\ln(1-p_i)$$

⇨ $$E = -\sum_{k} t_k \ln y_k$$

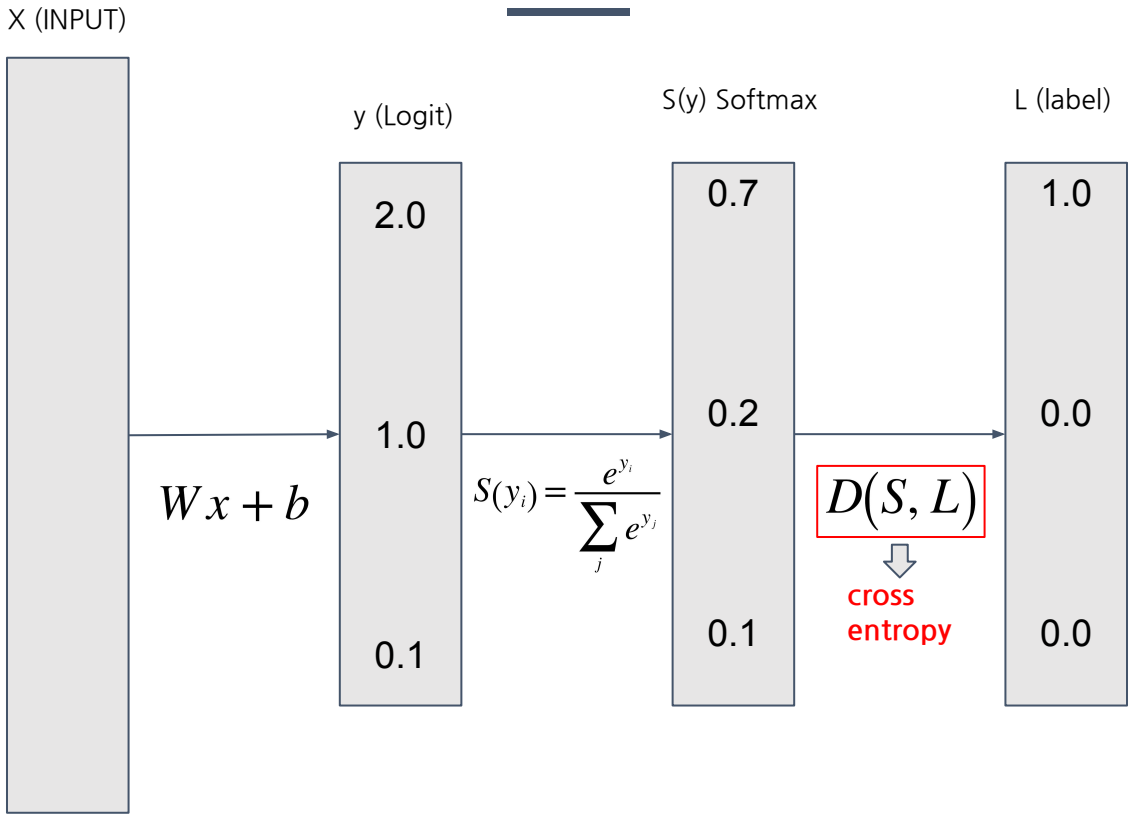
# Cost 함수 - 교차 엔트로피




$$-\sum_{i=1}^{m} y_i \ln(p_i) + (1-y_i)\ln(1-p_i)$$

- y = 1 일때 p=1에 가까우면?
  y = 1 일때 p=0에 가까우면?
  y = 0 일때 p=1에 가까우면?
  y = 0 일때 p=0에 가까우면?

# Cost 함수 - 교차 엔트로피



# 교차 엔트로피

실습

# 미니 배치

- cross entropy 손실함수

$$E = -\frac{1}{N}\sum_{n}\sum_{k} t_{nk}\log y_{nk}$$

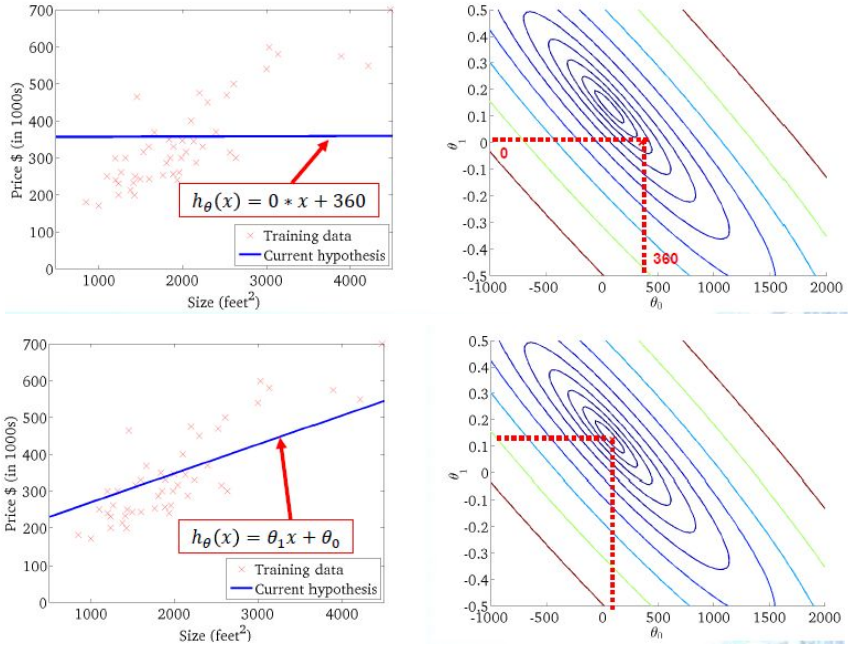
n : n번째 데이터, k : k번째 값, N : 데이터 사이즈

➡️ 평균 손실 함수

➡️ 데이터 양이 많기 때문에 전체 평균 손실 함수를 구하기 어렵다. 그래서 데이터 일부를 추려 전체의 '근사치'로 이용하여 학습하는 것이 **미니배치 학습**

# 정확한 Hypothesis를 찾는 과정

3. Cost 값이 낮아지는 방향으로 Weight 값들을 업데이트 하다가 Cost 값이 가장 낮을때 멈춘다.



# Cost 함수 (비용 함수) 최소화

- 머신러닝 회귀 알고리즘은 데이터를 계속 학습하면서 이 비용 함수가 반환하는 값(즉, 오류 값)을 지속해서 감소시키고 더 이상 감소하지 않는 최소의 오류 값을 구하는 것
- 손실 함수(Loss function) :

Mean Square Error(연속)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - t_i)^2$$

Cross Entropy(분류)

$$-\sum_{i=1}^{n}\ln(y_i)\times t_i$$

- w 파라미터의 개수가 적다면 고차원 방정식으로 비용 함수가 최소가 되는 w 변수값을 도출하겠지만 그 개수가 많다면 고차원 방정식으로 풀기 어려움

➡️ **경사 하강법 (Gradient Descent) !**

# 미분(derivate)

- 도함수, 순간 변화량

$$\frac{df(x)}{dx} = \lim_{n \to 0} \frac{f(x+h) - f(x)}{h}$$

- 편미분 : 변수가 여러개 일때 하나의 변수에 대해서 미분

$$f(x_a, x_b) = x_a^2 + x_b^2$$

$x_a$ 에 대한 편미분 : $\frac{\partial f}{\partial x_a} = 2x_a$

$x_b$ 에 대한 편미분 : $\frac{\partial f}{\partial x_b} = 2x_b$

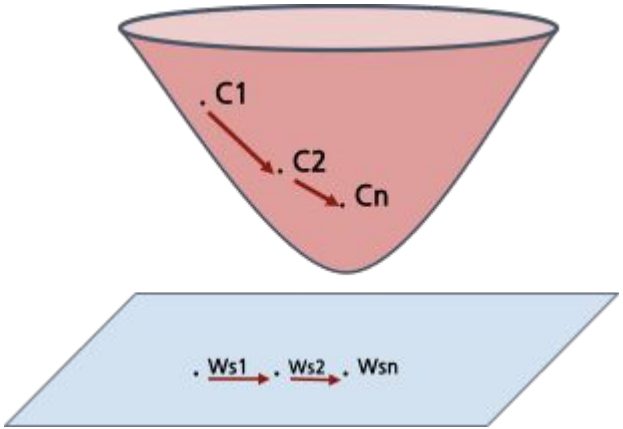
# 미분

## 실습

# 경사 하강법 (Gradient Descent)

- Gradient : 기울기
- Descent : 하강

➡️ 점진적으로 반복적인 계산을 통해서 W 파라미터 값을 업데이트 해가면서 (기울기가 감소하는 방향으로 이동하면서) 오류 값이 최소가 되는 W 파라미터를 구하는 방식입니다

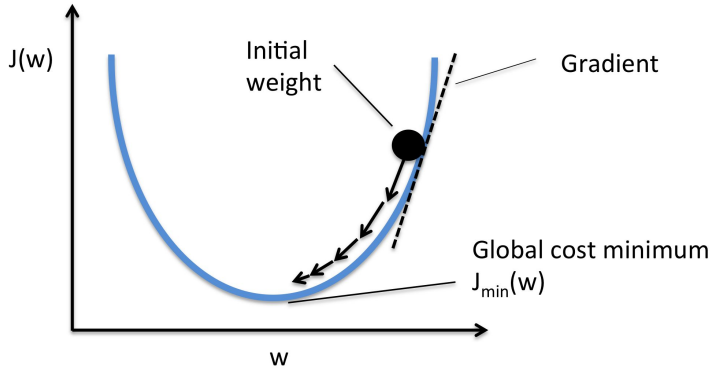


# 기울기와 경사 하강법

실습

# 최적화 (Optimization)

- 어떻게 하면 오류가 작아지는 방향으로 w값을 보정할 수 있을까?



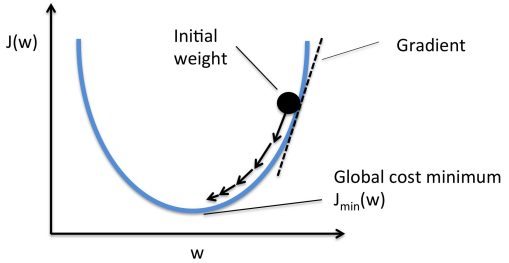

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$

➡ w에서 부터 미분을 적용한 뒤 이 미분 값이 감소하는 방향으로 순차적으로 w를 업데이트하면서 기울기가 0일때 멈춘다.

# 최적화 (Optimization)

- 비용함수를 $w_0$, $w_1$에 대해 편미분

$$RSS(w_0, w_1) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(w_0 + w_1 * x_i\right)\right)^2$$




$$\frac{\partial R(\omega)}{\partial \omega_1} = \frac{2}{N}\sum_{i=1}^{N} - x_i * \left(y_i - \left(w_0 + w_1 x_1\right)\right)$$

실제값$_i$ 예측값$_i$

$$\frac{\partial R(\omega)}{\partial \omega_0} = \frac{2}{N}\sum_{i=1}^{N} - \left(y_i - \left(w_0 + w_1 x_1\right)\right)$$

즉, 비용함수 RSS(w0, w1) 이 최소가 되는 w1, w0를 구할 수 있다.

$$W_1 = w_1 - \left(-\frac{2}{N}\sum_{i=1}^{N} x_i \times \left(y_i - \hat{h}_i\right)\right) \qquad \hat{h}_i = w_0 + w_1 x_i$$

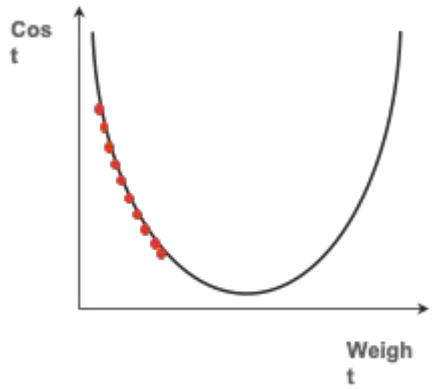
편미분 값이 클 수 있기 때문에 보정 $\eta$ 수 를 곱하는데 이를 '학습률'이라고 한다.

$$W_1 = W_1 - \eta\left(-\frac{2}{N}\sum_{i=1}^{N} x_i \times \left(y_i - \hat{h}_i\right)\right)$$

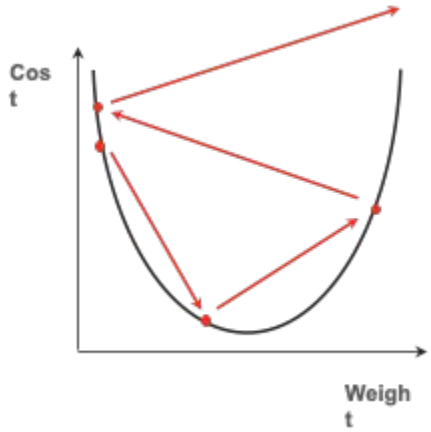
# 학습률 (Learning Rate)

- 한번의 학습으로 얼마만큼 학습해야 할지, 즉 매개변수 w값을 얼마나 갱신하느냐를 정하는 것

- too small ⇒ 너무 오래 걸린다




- too large ⇒ 학습이 안된다



# 최적화 (Optimization) 프로세스

✓ Step 1 : $w_1, w_0$를 임의의 값으로 설정하고 첫 비용 함수의 값을 계산

✓ Step 2 : $w_1$을 $w_1 - \eta\frac{2}{N}\sum_{i=1}^{N} x_i * \left(y_i - \hat{h}_i\right)$ , $w_0$을 $w_0 - \eta\frac{2}{N}\sum_{i=1}^{N} \left(y_i - \hat{h}_i\right)$ 으로 업데이트 한 후 다시 비용 함수의 값을 계산

✓ Step 3 : 비용함수의 값이 감소했으면 다시 Step 2를 반복합니다. 더 이상 비용 함수의 값이 감소하지 않으면 그 때의 $w_1, w_0$를 구하고 반복을 중지합니다.

# 학습 알고리즘 구현하기

- 신경망에는 적응 가능한 가중치화 편향이 있고, 이 가중치와 편향을 훈련 데이터에 적응하도록 조정하는 과정을 '학습'이라고 한다
  1. 훈련 데이터 중 일부를 무작위로 가져온다. 이렇게 선별한 데이터를 미니배치라 하며, 그 미니배치의 손실 함수 값을 줄이는 것이 목표 (**미니배치**)
  2. 미니배치의 손실 함수 값을 줄이기 위해 각 가중치 매개변수의 기울기를 구한다. 기울기는 손실 함수의 값을 가장 작게 하는 방향을 제시(**기울기 산출**)
  3. 가중치 매개변수를 기울기 방향으로 아주 조금 갱신한다(**매개변수 갱신**)
  4. 1~3 단계 반복

➡️ 확률적 경사 하강법 (SGD, stochastic gradient descent)

# 단층 신경망 클래스 구현

실습

# 2층 신경망 클래스 구현

실습

# 연쇄 법칙

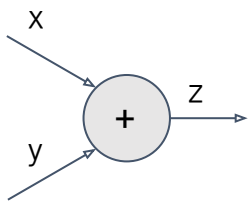
- 합성 함수
  - $z = (x + y)^2$ 의 식은 다음과 같은 두 개의 식으로 구성
    - z = t2
    - t = x + y
  - 합성함수의 미분은 합성함수를 구성하는 각 함수의 미분의 곱, 즉 위의 식을 x에 대해 미분하면

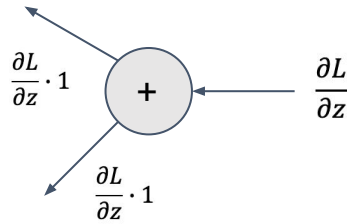
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t} \times \frac{\partial t}{\partial x} = 2t \times 1 = 2 \times (x + y)$$

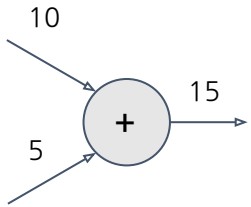
➡️ 연쇄 법칙(Chain Rule) : 해당 노드의 국소적 계산을 전달함으로써 복잡한 계산을 처리

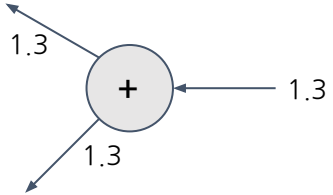
# 덧셈 노드의 역전파

- z = x + y 의 미분
  - $\frac{\partial z}{\partial x} = 1 \qquad \frac{\partial z}{\partial y} = 1$



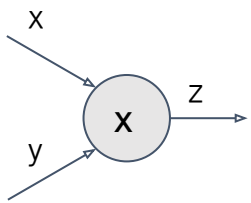



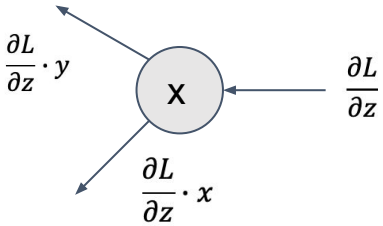



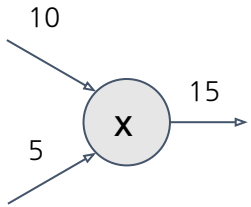



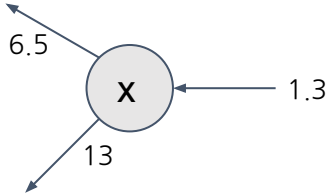
# 곱셈 노드의 역전파

- z = xy 의 미분
  - $\frac{\partial z}{\partial x} = y \qquad \frac{\partial z}{\partial y} = x$



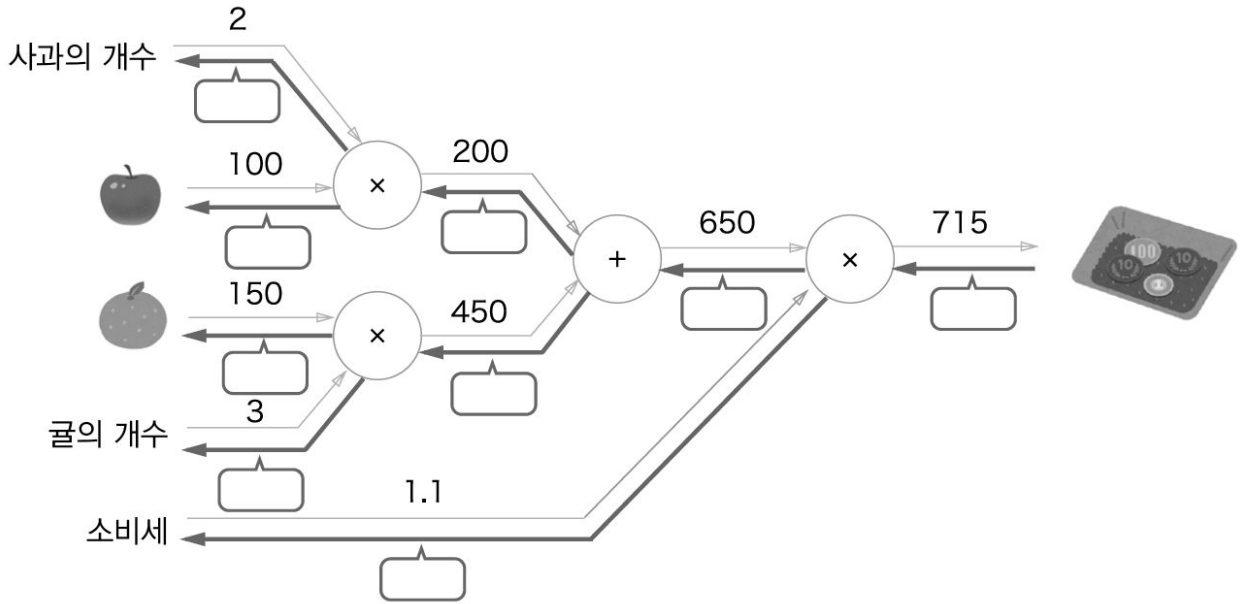



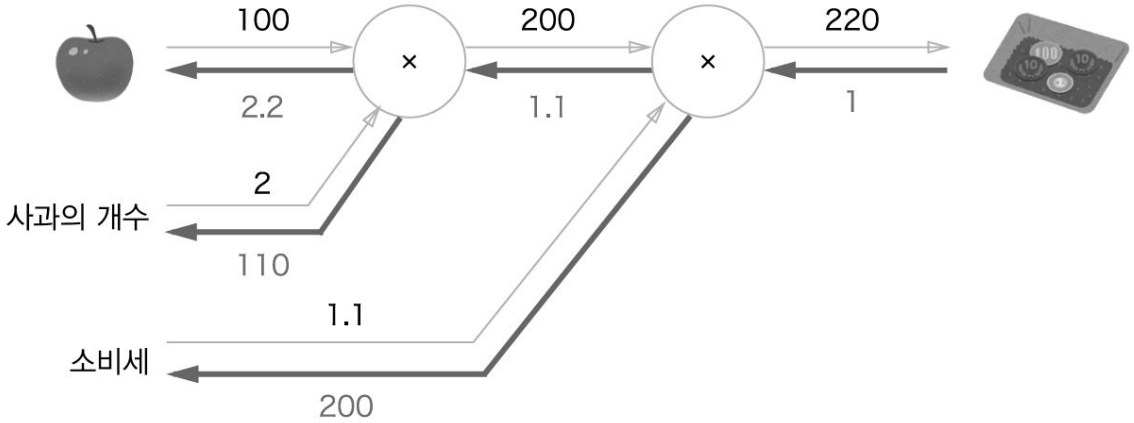
# 역전파

- 계산 그래프의 예
  - (사과의 가격 * 사과의 개수 +귤의 가격 * 귤의 개수) * 소비세



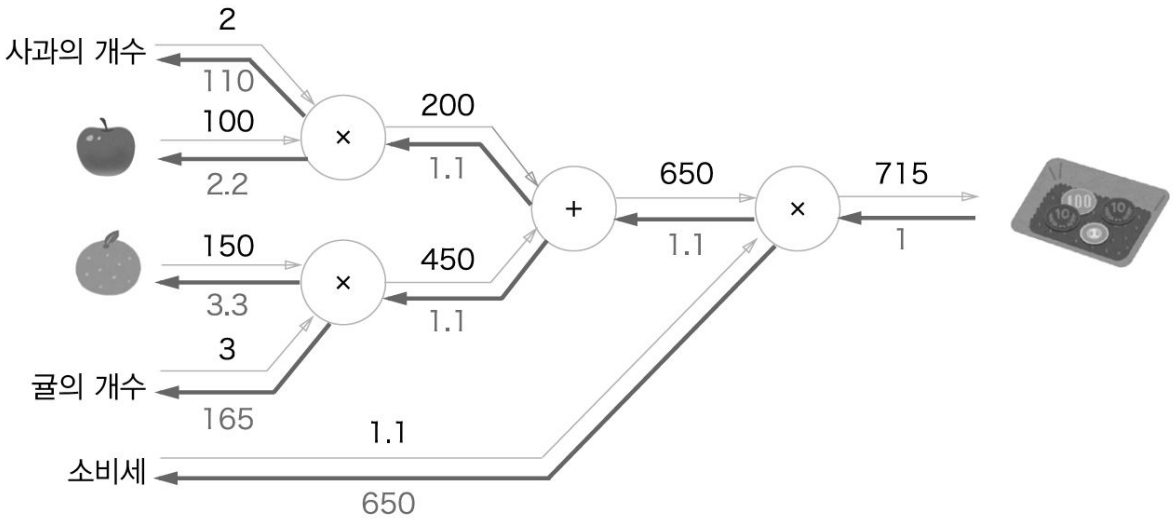
# 곱셈 계층 역전파

## 실습



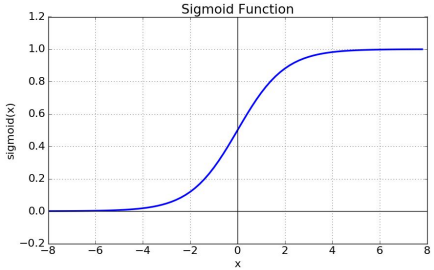
# 역전파

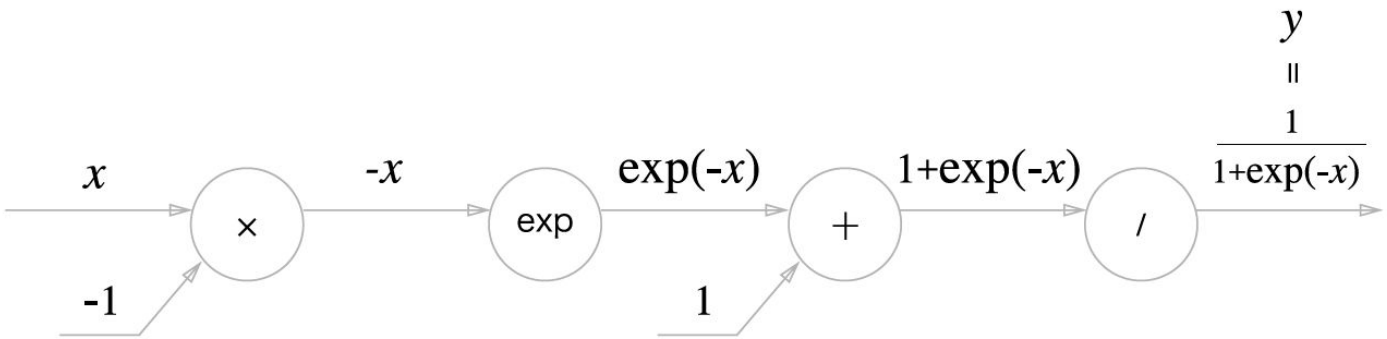
# 실습



# Sigmoid 노드의 역전파

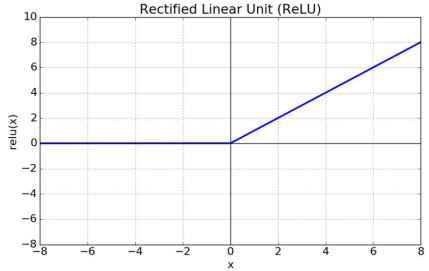
- 시그모이드 함수
  - $$y = \frac{1}{1 + \exp(-x)}$$



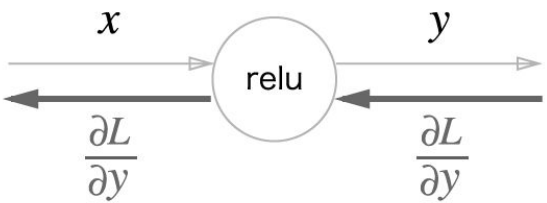



# ReLU 노드의 역전파

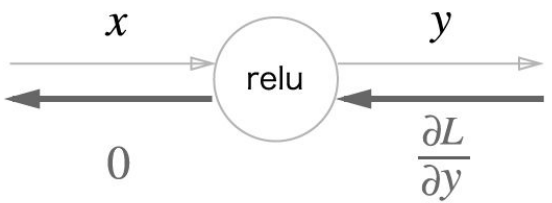
- ReLU 함수
  - $$y = \begin{cases} x\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$$




$x > 0$




$x \leq 0$

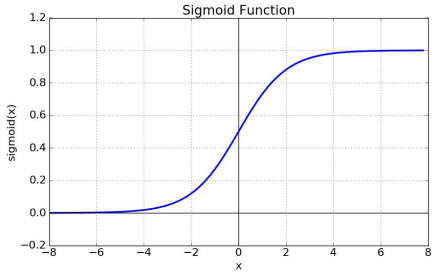


# Relu 노드의 역전파

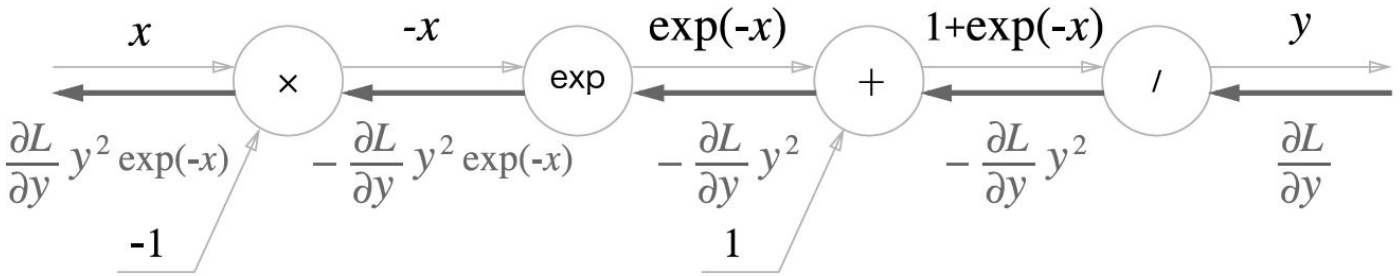
실습

# Sigmoid 노드의 역전파

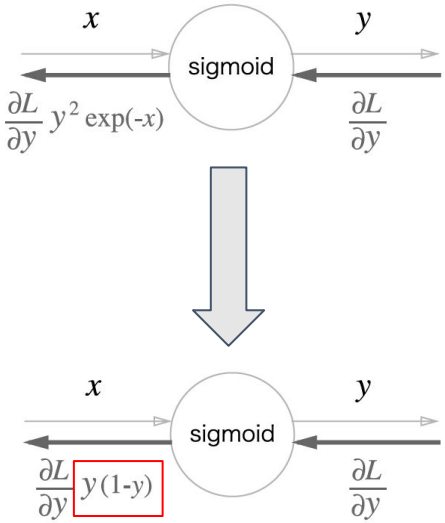
- 시그모이드 함수
  - $$y = \frac{1}{1 + \exp(-x)}$$




| ※ 나눗셈 미분 : y = 1/x | ※ 지수 미분 : y = exp(x) |
|---|---|
| $$\frac{\partial y}{\partial x} = -\frac{1}{x^2}$$ $$= -y^2$$ | $$\frac{\partial y}{\partial x} = \exp(x)$$ |



# Sigmoid 노드의 역전파




$$\frac{\partial L}{\partial y} y^2 \exp(-x) = \frac{\partial L}{\partial y} \frac{1}{(1+\exp(-x))^2} \exp(-x)$$

$$= \frac{\partial L}{\partial y} \frac{1}{1+\exp(-x)} \frac{\exp(-x)}{1+\exp(-x)}$$

$$= \frac{\partial L}{\partial y} y(1-y)$$

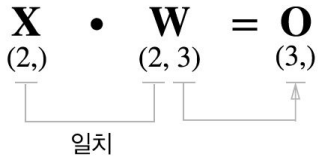
➡️ 시그모이드 함수 미분 : $\sigma' = \sigma(1-\sigma)$

# Sigmoid 노드의 역전파

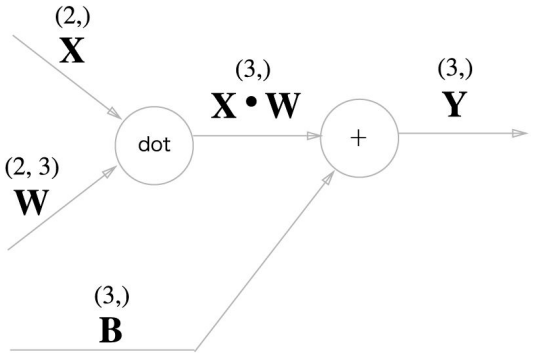
실습

# 행렬 노드의 계산 그래프

- 행렬의 곱에서는 대응하는 차원의 원소 수를 일치 시킨다



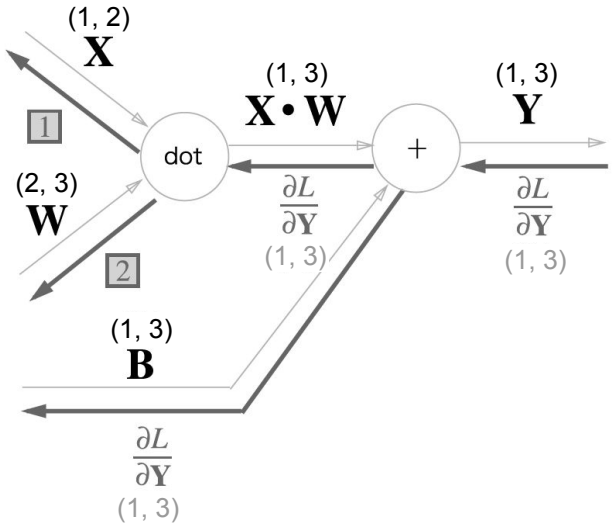

- $X \cdot W + B$ (Affine 행렬)



# Affine 계층의 역전파

$$\boxed{1}\ \frac{\partial L}{\partial \mathbf{X}} = \frac{\partial L}{\partial \mathbf{Y}}\ \mathbf{W}^{\mathrm{T}}$$
(1, 2) (1, 3) (3, 2)

$$\boxed{2}\ \frac{\partial L}{\partial \mathbf{W}} = \mathbf{X}^{\mathrm{T}}\ \frac{\partial L}{\partial \mathbf{Y}}$$
(2, 3) (2, 1) (1, 3)

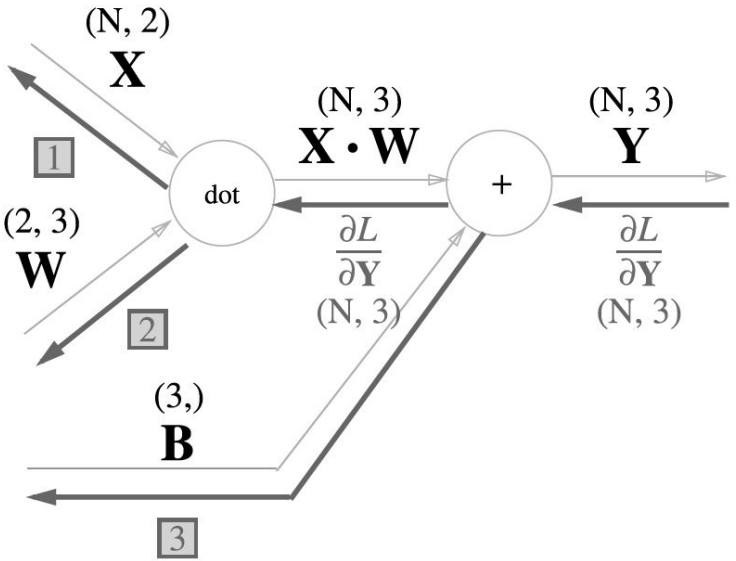



- 다차원 행렬의 미분 (행렬의 미분)
- 차원을 맞추는 것만 기억하자. 예) $\boldsymbol{w = w - \mu \cdot \frac{\partial L}{\partial w}}$ 에서 w 와 dw 의 차원이 같아야 한다

# 배치용 Affine 계층의 역전파

1 $\underset{(N,\,2)}{\frac{\partial L}{\partial \mathbf{X}}} = \underset{(N,\,3)}{\frac{\partial L}{\partial \mathbf{Y}}} \cdot \underset{(3,\,2)}{\mathbf{W}^{\mathrm{T}}}$

2 $\underset{(2,\,3)}{\frac{\partial L}{\partial \mathbf{W}}} = \underset{(2,\,N)}{\mathbf{X}^{\mathrm{T}}} \cdot \underset{(N,\,3)}{\frac{\partial L}{\partial \mathbf{Y}}}$




3 $\underset{(3)}{\frac{\partial L}{\partial \mathbf{B}}} = \underset{(N,\,3)}{\frac{\partial L}{\partial \mathbf{Y}}}$ 의 첫 번째 축(0축, 열방향)의 합

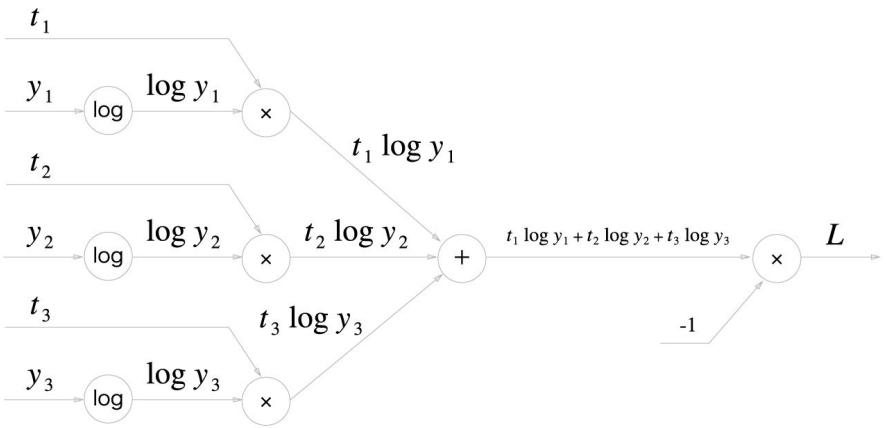
# 배치용 Affine 역전파

실습

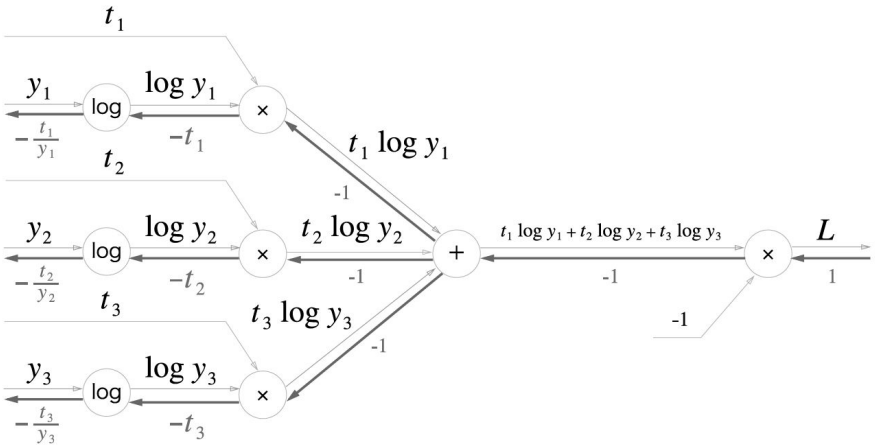
# Cross Entropy 역전파

$$L = -\sum_k t_k \log y_k$$

※ 로그 미분 : $y = \log x$ ➡️ $\frac{\partial y}{\partial x} = \frac{1}{x}$



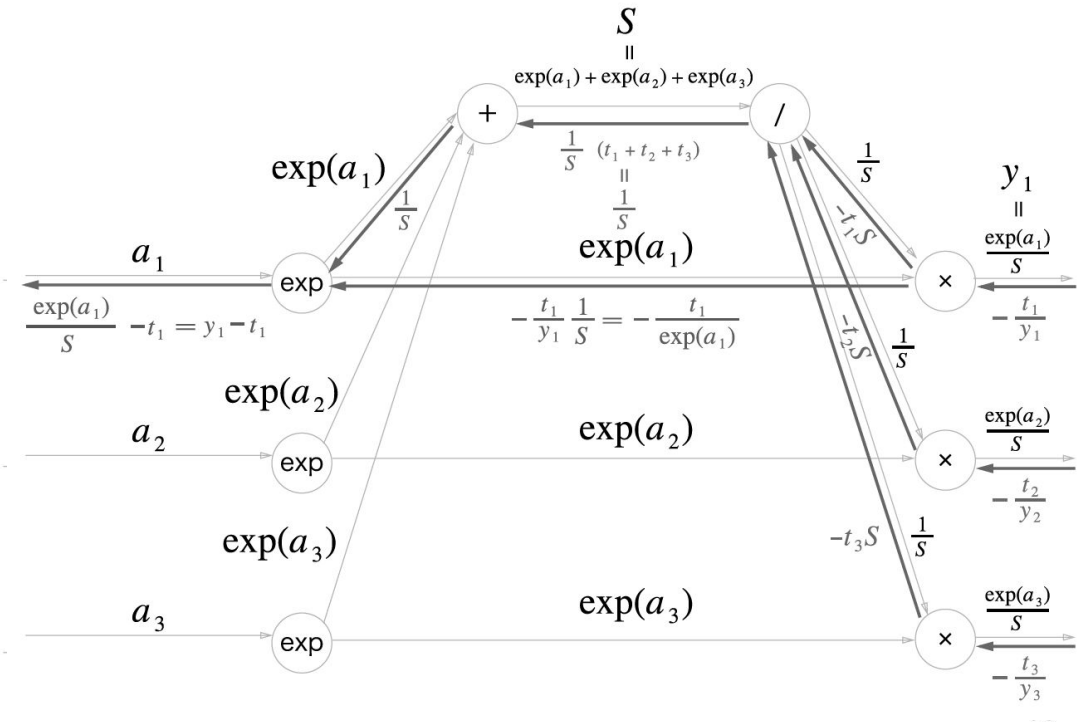

Cross Entropy Error 계층의 순전파




Cross Entropy Error 계층의 역전파

# Softmax 역전파

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$




$$y = exp(x)$$

$$\frac{\partial y}{\partial x} = exp(x)$$

$$(\frac{1}{S} - \frac{t_1}{exp(a_1)})exp(a_1)$$

# Softmax with Loss 역전파

실습