Linear Regression

Unsupervised	Supervised
• Clustering & Dimensionality Reduction o SVD o PCA o K-means	• Regression o Linear o Polynomial • Decision Trees • Random Forests
• Association Analysis o Apriori o FP-Growth • Hidden Markov Model	• Classification o KNN o Trees o Logistic Regression o Naive-Bayes o SVM

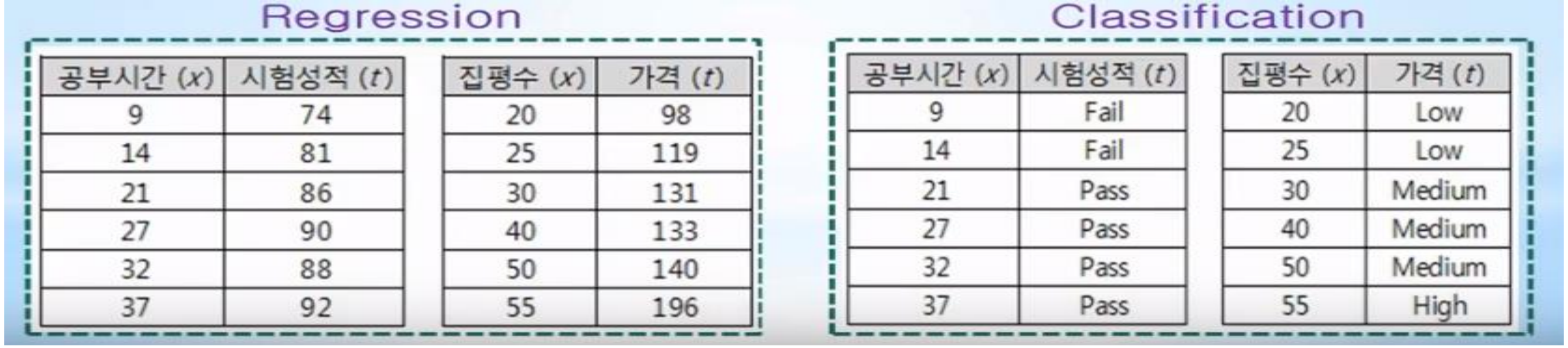
Linear Regression

지도학습(Supervised Learning)은 입력 값(x)과 정답(t, label)을 포함하는 Training Data를 이용하여 학습하고, 그 학습된 결과를 바탕으로 미지의 데이터(Test Data)에 대해 미래 값을 예측(predict) 하는 방법 ⇒ 대부분 머신러닝 문제는 지도학습에 해당됨

[예1] 시험공부 시간(입력)과 Pass/Fail (정답)을 이용하여 당락 여부 예측

[예2] 집 평수(입력)와 가격 데이터(정답) 이용하여 임의의 평수 가격 예측

Linear Regression | Logistic Regression



Regression

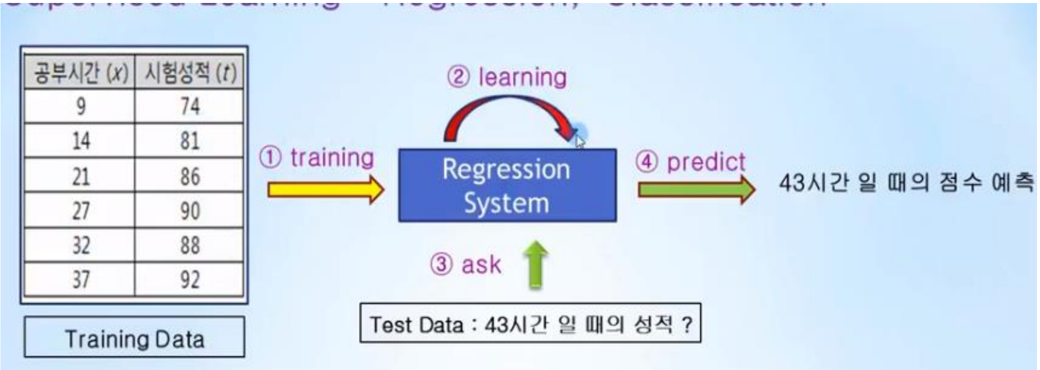
공부시간 (x)	시험성적 (t)
9	74
14	81
21	86
27	90
32	88
37	92

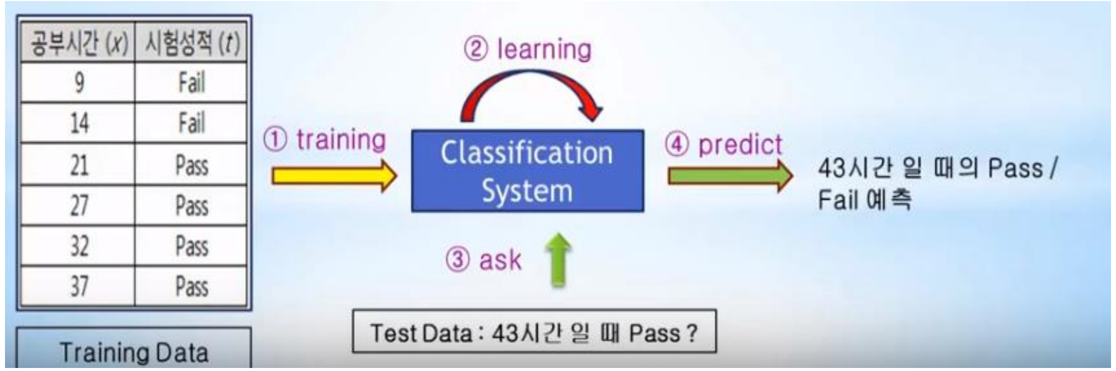
집평수 (x)	가격 (t)
20	98
25	119
30	131
40	133
50	140
55	196

Classification

공부시간 (x)	시험성적 (t)
9	Fail
14	Fail
21	Pass
27	Pass
32	Pass
37	Pass

집평수 (x)	가격 (t)
20	Low
25	Low
30	Medium
40	Medium
50	Medium
55	High





회귀분석에서는 Training Data에서 보여지듯 공부시간에 대한 값입력에 대해서 결과값인 시험성적이 연속적인 반면, Classification에서는 입력값은 회귀분석과 동일한 값이지만 결과를 이상적인 분류값으로 나태내주고 있다.

Linear Regression

- Linear Regression
  - 독립변수와 종속변수의 관계를 분석함
  - 데이터의 분포경향을 학습하여 새로운 데이터가 들어왔을 때 결과값을 예측하는 것
  - 결과값이 연속적인 수로 나타난다

  - 독립변수 1개와 종속변수 1개
  - https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-gradient-descent-fcd5e0fc077d

Linear Regression

Linear Regression

학생들과 성적의 관계는 학생들마다 다양한 성적 분포를 가지는데, 여기에 어떤 연관이 있는지 알아내고 그 연관 관계를 이용해서 결국에는 특정학생의 성적을 예측할 수 있다.

학생들의 기말고사 성적은 [ ]에 따라 다르다
[ ]안에 시험성적을 좌우할 만한 요소들로 무엇이 있을까?

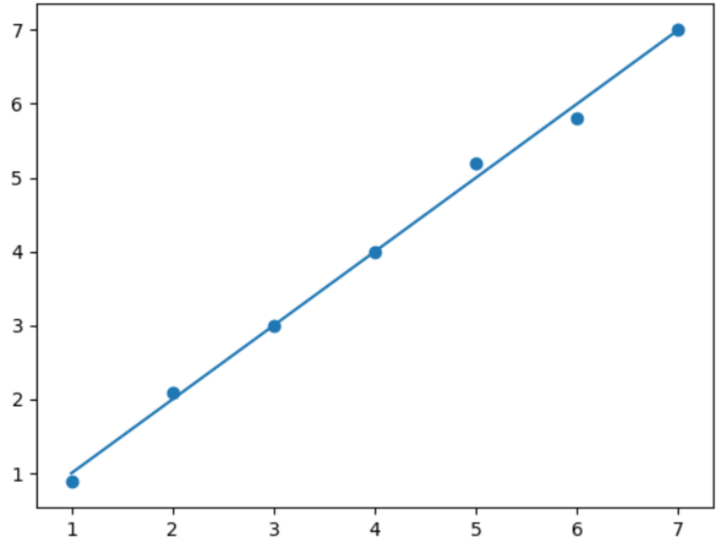
여기서 [ ]안에 들어갈 내용을 '정보'라 한다.
머신러닝과 딥러닝은 이 정보가 필요하다.
정보를 정확히 준비해 놓기만 하면 성적을 예측하는 방정식을 만들 수 있다.

이것을 수학적으로 정의하면, 성적을 변하게 하는 '정보' 요소를 X라 하고,
이 값에 따라 변하는 '성적'을 Y라 한다.

'X값이 변함에 따라 Y값도 변한다'는 정의 안에서 독립적으로 변할 수 있는 값 X를 독립변수라 한다.
또한, 이 독립 변수에 따라 종속적으로 변하는 Y를 종속변수라 한다.

선형회귀는 독립변수 X를 이용해서 종속변수 Y를 예측하고 설명하는 작업을 말한다.

Linear Regression



Linear Regression 은 Training Data를 나타내는 하나의 직선을 찾아내는 것이 핵심
어느 정도의 기울기를 나타내는 직선을 찾아내는가의 문제이다.

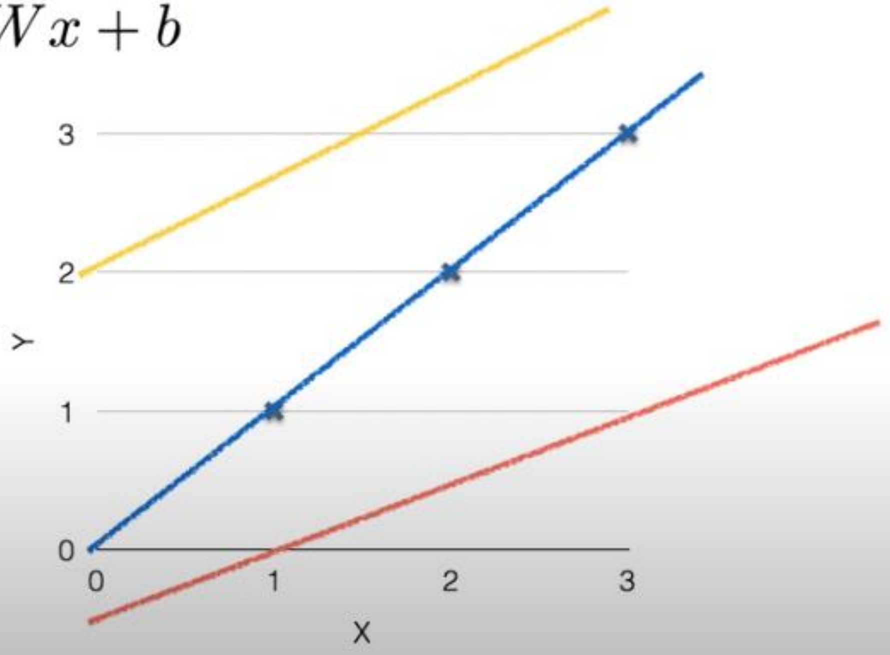
'데이터 들을 표현할 수 있는 직선이 존재한다 ' 라는 Hypothesis를 만들어 낼 수 있으며 공식은 다음과 같다.

좌표에서 직선은 W(Weight, 기울기)와 b(bias,절편) 을 갖는 함수로 표현할 수 있다.
즉 학습 데이터를 잘 표현할 수 있는 W와 b를 잘 찾아내는 것이 학습의 목표이다.

Linear Regression

(Linear) Hypothesis

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

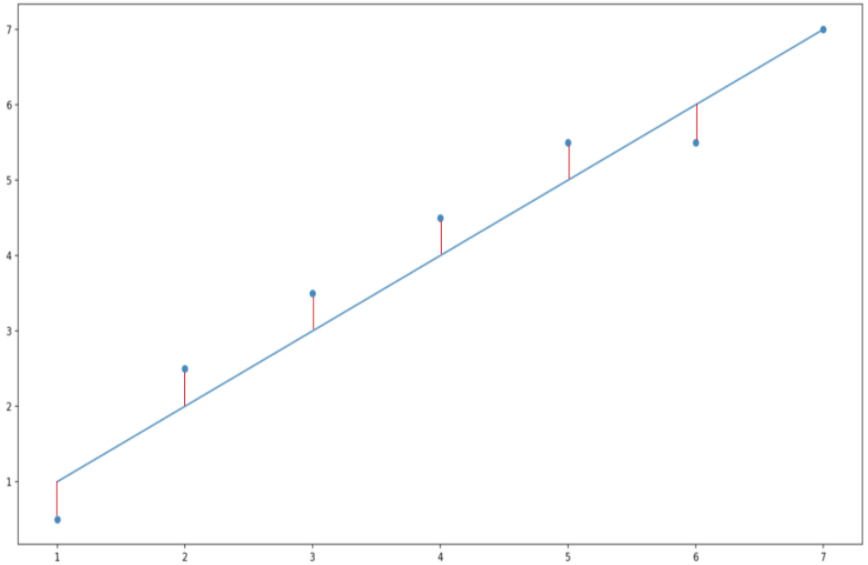


직선들의 모양은 W와 b에 따라서 달라진다.
서로 다른 직선들 중에서 우리의 데이타셋을 가장 잘 표현한 선은 어떤 선일까?

Linear Regression

Cost Function

그렇다면 직선이 학습 데이타를 잘 표현하고 있는지를 어떤 방법으로 알 수 있을까?



직선이 예상한 값과 실제 데이터 값이 얼마나 차이가 있는지를 확인하면 된다. 그림에서 파란 직선 위 값들이 예상결과이고 파란색 점들이 실제 데이터다. 직선 위 점과 실제 점 사이의 거리를 측정해서 거리 차가 좁을수록 정확도가 높아진다.

이 거리를 측정하는 것이 Cost(Loss) Function이다.

H(x) – y → 가설의 값과 실제 데이터 값 사이의 차가 손실. Loss
거리를 측정하는 것이기에 음수 값이 나오면 안됨.
(H(x) – y)2 → 제곱을 함으로써 차이가 커지면 손실치를 더 부가

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

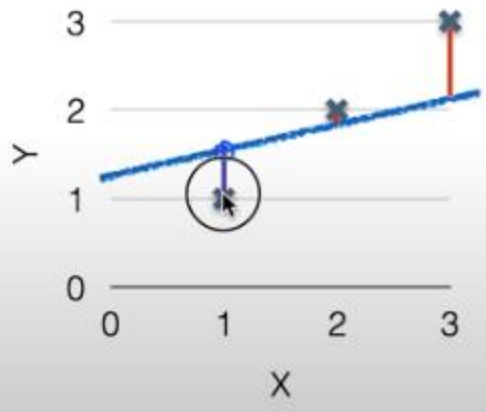
Linear Regression

- Cost Function

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

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

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



Linear Regression

- Cost Function

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

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

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

Cost Function은 실제적으로 W와 b에 대한 Function이다.
Linear Regression의 목적은 가장 작은 값을 가지는 W와 b를 구하는 것이며
이것이 Linear Regression 학습이다.

Linear Regression

Simplified hypothesis

$$H(x) = Wx$$

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

What *cost(W)* looks like?

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

X	Y
1	1
2	2
3	3

- W=1, *cost(W)*=?

Linear Regression

What cost(W) looks like?

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

x	Y
1	1
2	2
3	3

- W=1, cost(W)=0

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

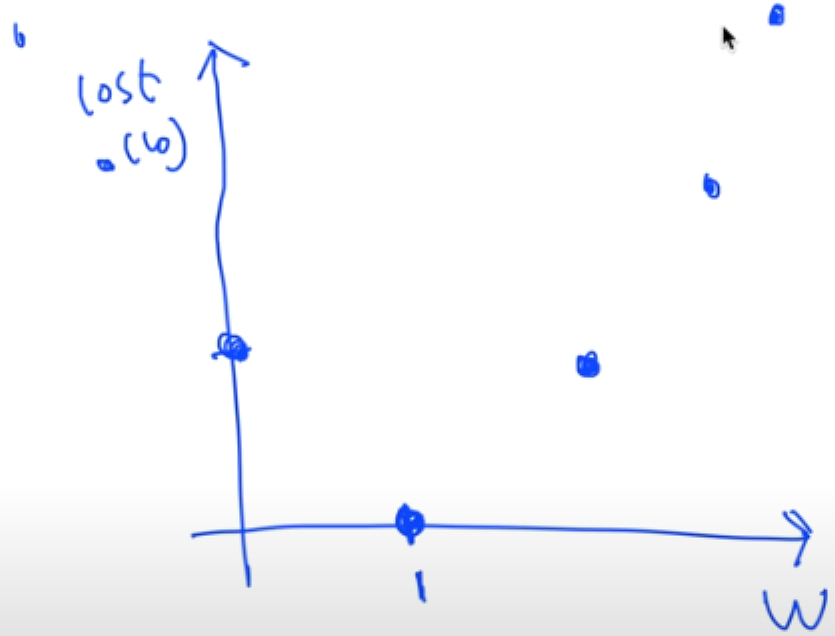
- W=0, cost(W)=4.67

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

- W=2, cost(W)=?

What *cost(W)* looks like?

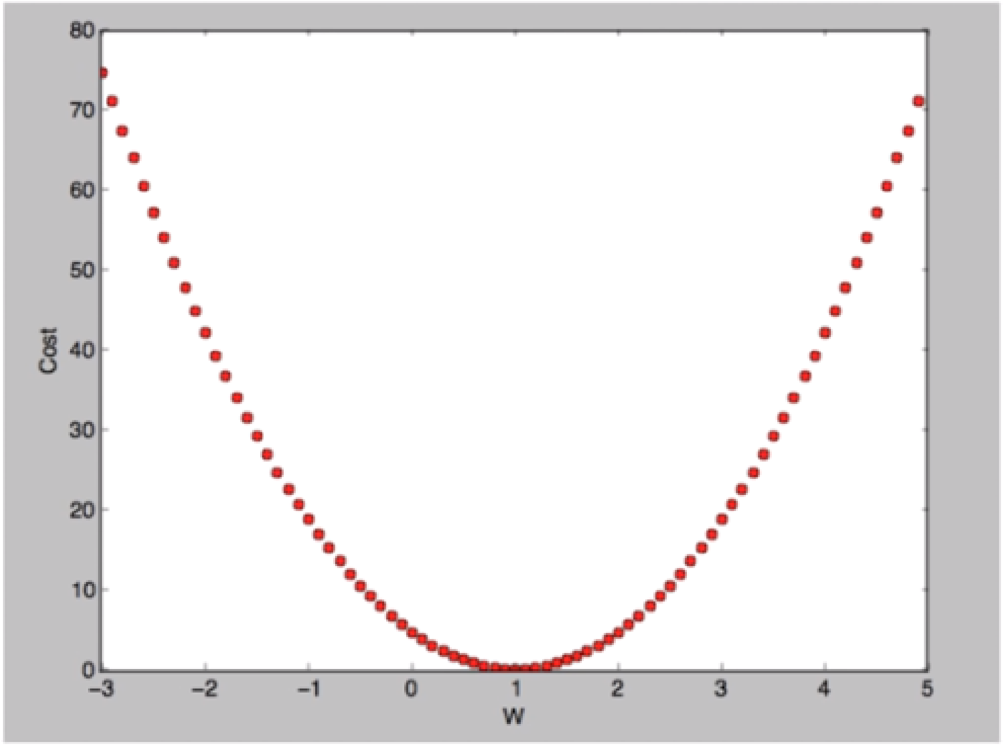
- W=1, cost(W)=0
- W=0, cost(W)=4.67
- W=2, cost(W)=4.67



Linear Regression

What *cost(W)* looks like?

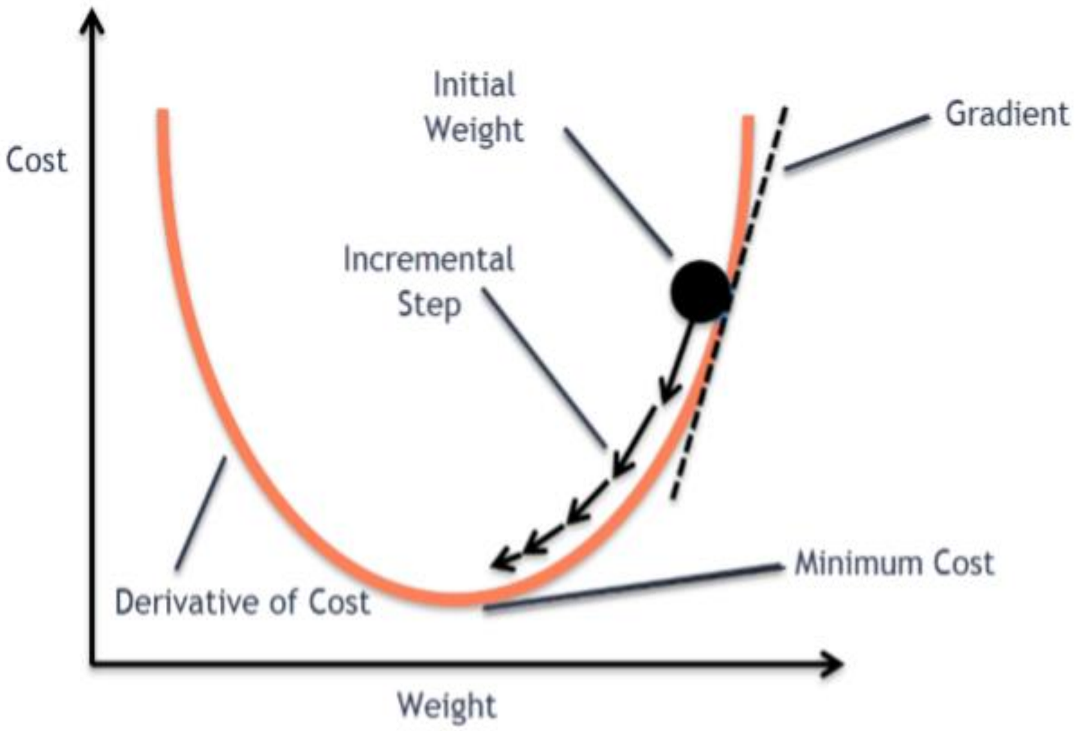
$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

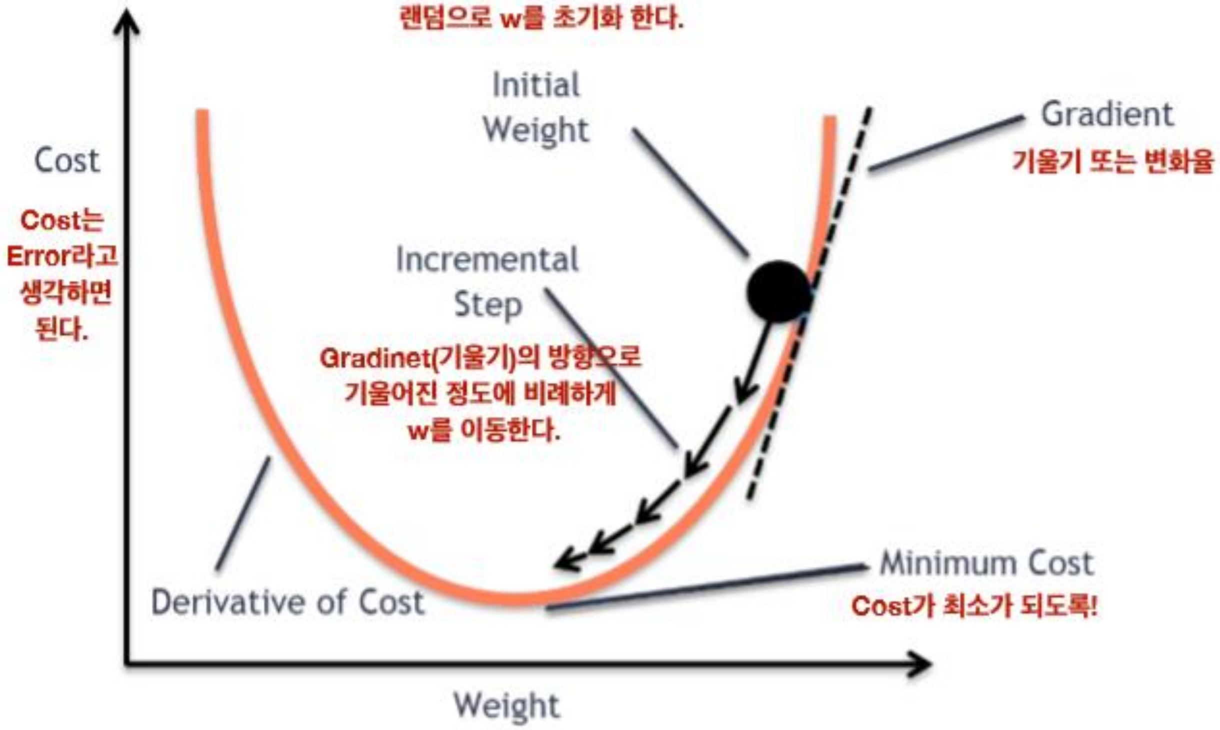


Linear Regression

- Gradient Descent

Gradient Descent는 학습 알고리즘 중 하나.
학습이라 하면 머신러닝 알고리즘의 결과가 좋아지도록 파라미터(parameter)를 조정하는 것.
가중치(weight), 편향(bias)이 파라미터에 포함된다.





Linear Regression

- Hypothesis

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

- Cost function

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

- Gradient descent algorithm

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

Linear Regression

- Gradient Descent

섭씨(Celcious)를 화씨(Farenhight)로 변환하는 간단한 문제에 Gradient Descent를 적용하며 알아보도록 한다.
단 온도 변환에 대한 아무런 사전 지식 없이, 인공지능 알고리즘이 스스로 변환 공식을 찾도록 유도한다.

섭씨-화씨 변환 공식

$$F(Farenhight) = C(Celcious) \times 1.8 + 32$$

섭씨	화씨
20	68
22	71.6
14	57.2
36	96.8

공식을 바탕으로 데이터셋을 생성. (약 100여 개)

Linear Regression

- Gradient Descent

m = The number of data

$x^{(i)}$ = The feature of i'th data

$y^{(i)}$ = The label of i'th data

w = The weight

b = The bias

m = The number of data

데이터의 개수, 즉 데이터셋의 크기

$x^{(i)}$ = The feature of i'th data

섭씨(celsius, C) i 번째 feature(x)

$y^{(i)}$ = The label of i'th data

화씨(fahrenheit, F) i 번째 label(y)

w = The weight

가중치

b = The bias

편향

Linear Regression

- Gradient Descent

수식을 다루게 되므로 용어 표기(Notation)를 정리합니다.
우리는 Loss Function과 Cost Function을 정의함으로써 주어진 문제를 잘 맞췄는지 판단할 수 있다.

m = The number of data
데이터의 개수, 즉 데이터셋의 크기

$x^{(i)}$ = The feature of i'th data
섭씨(celsius, C) i 번째 feature(x)

$y^{(i)}$ = The label of i'th data
화씨(fahrenheit, F) i 번째 label(y)

w = The weight
가중치

b = The bias
편향

Hypothesis Function y와 x의 관계를 추정하는 함수

$$h(x) = wx + b$$

섭씨-화씨 변환 공식

$$F(Farenhight) = C(Celcious) \times 1.8 + 32$$

Loss Function 데이터 한 개의 Error

$$L(y, h(x)) = \frac{1}{2}\left(h(x) - y\right)^2$$

Cost Function 모든 Loss의 평균

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

Linear Regression

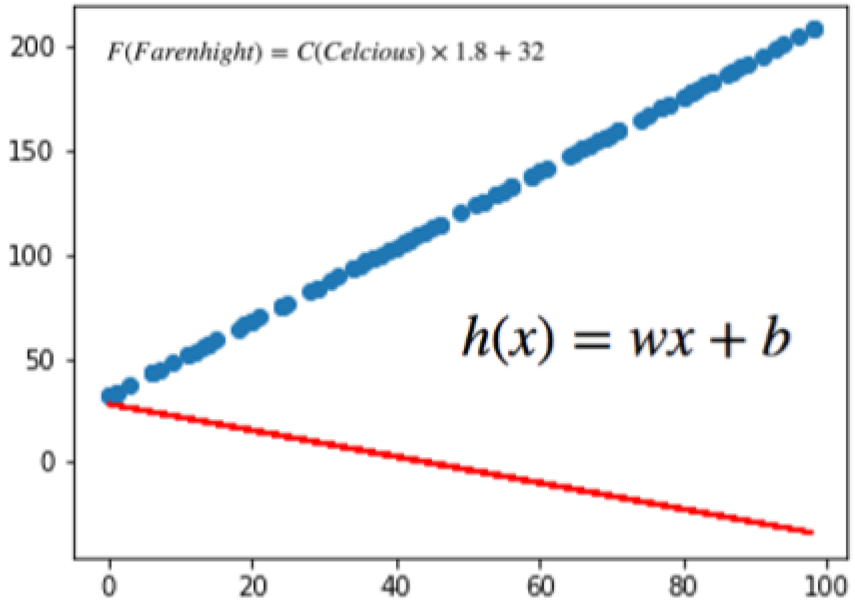
- Gradient Descent

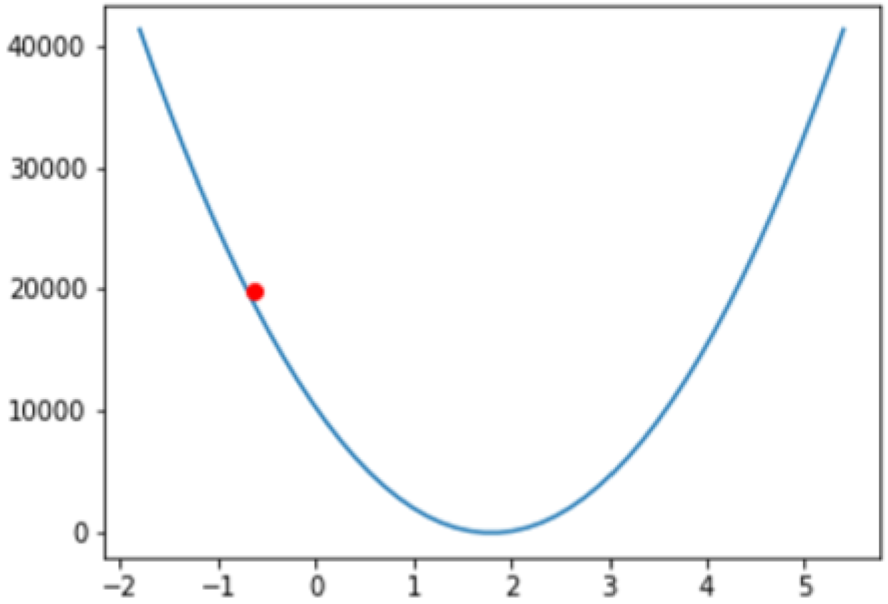
Cost Function를 타고 내려가다보면
결국에는 최적의 w를 찾을 수 있다

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

화씨(fahrenheit, F)





섭씨(celsius, C)

w = The weight

Linear Regression

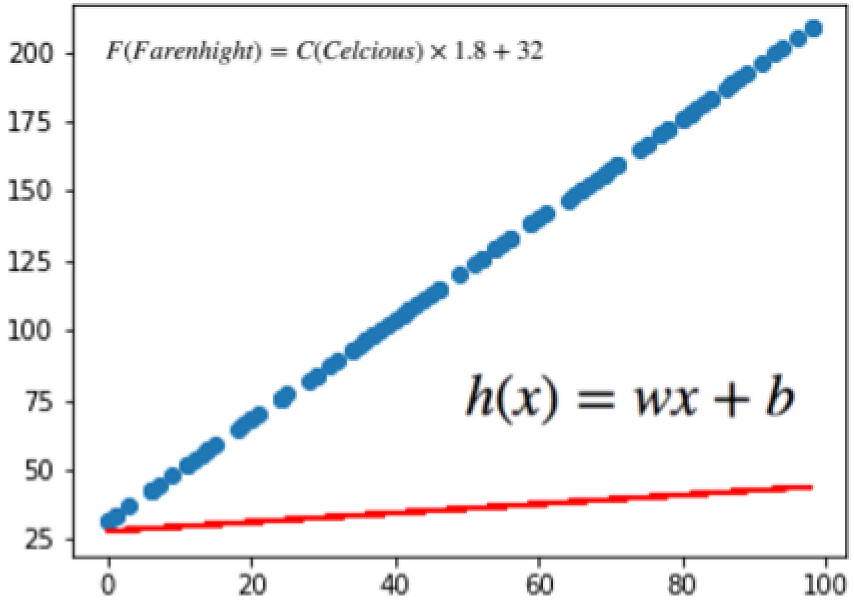
Gradient Descent

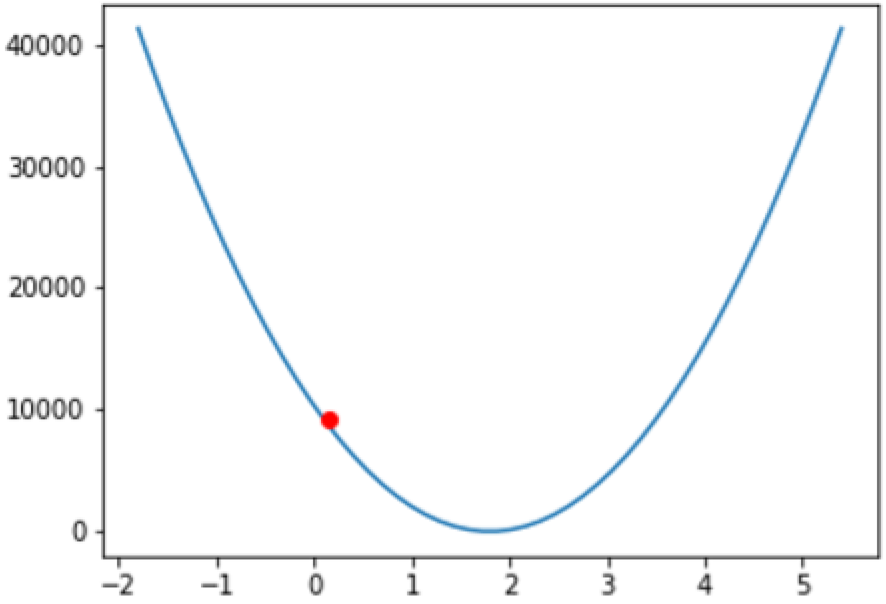
Cost Function를 타고 내려가다보면 결국에는 최적의 w를 찾을 수 있다

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

화씨(fahrenheit, F)





섭씨(celsius, C)

w = The weight

Linear Regression

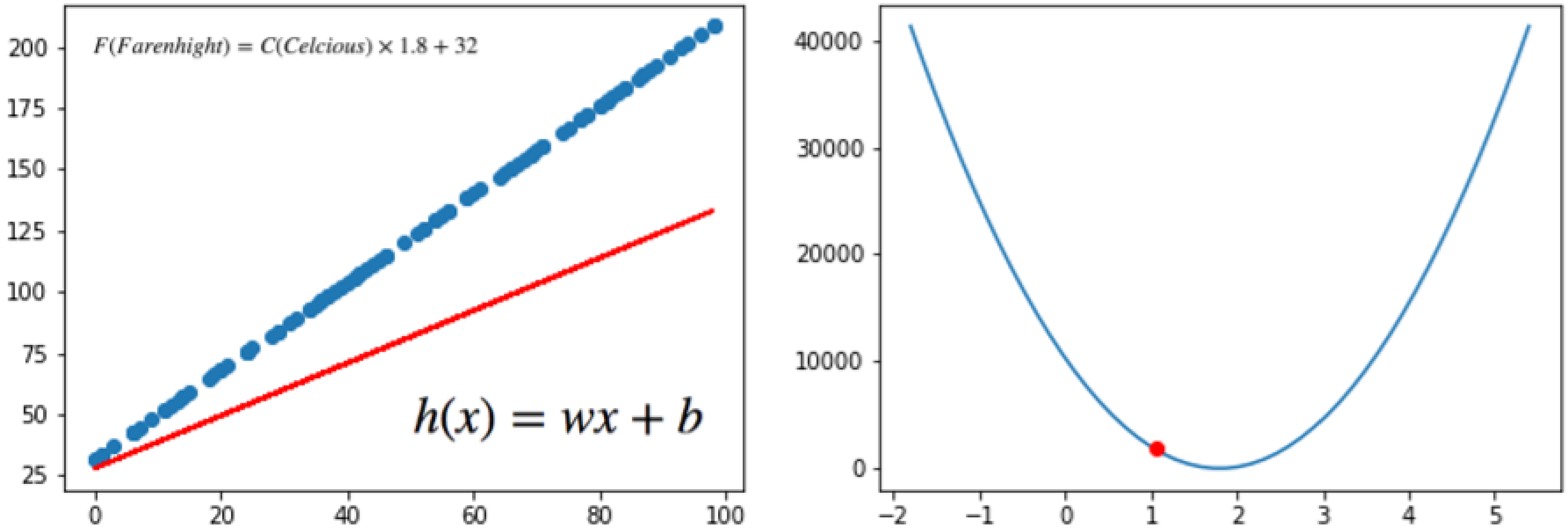
Gradient Descent

Cost Function를 타고 내려가다보면
결국에는 최적의 w를 찾을 수 있다

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

화씨(fahrenheit, F)



섭씨(celsius, C)

w = The weight

Linear Regression

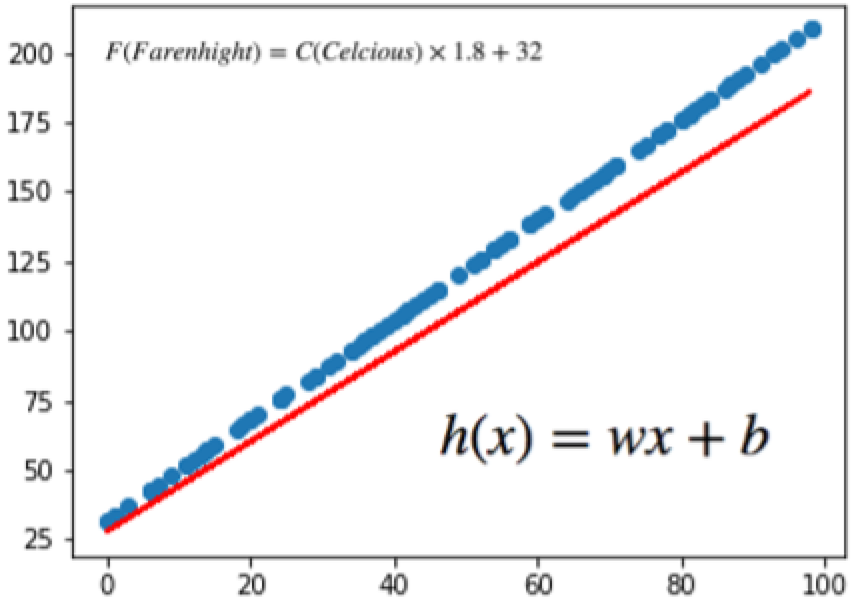
- Gradient Descent

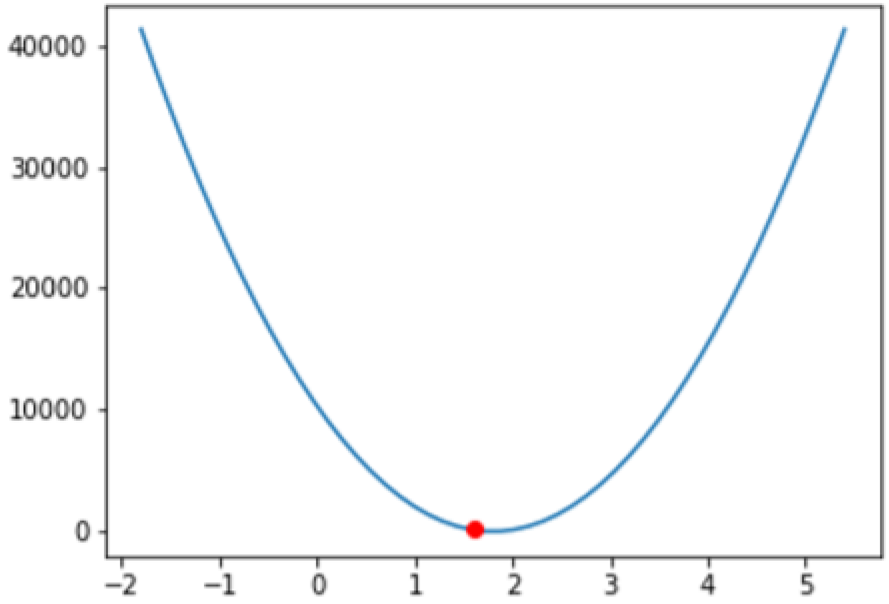
Cost Function를 타고 내려가다보면
결국에는 최적의 w를 찾을 수 있다

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

화씨(fahrenheit, F)





섭씨(celsius, C)

w = The weight

Linear Regression

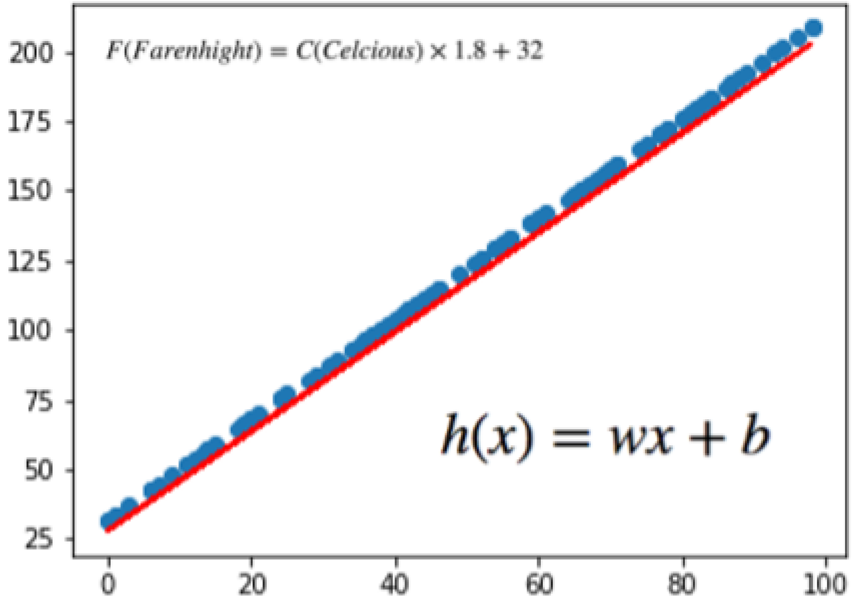
- Gradient Descent

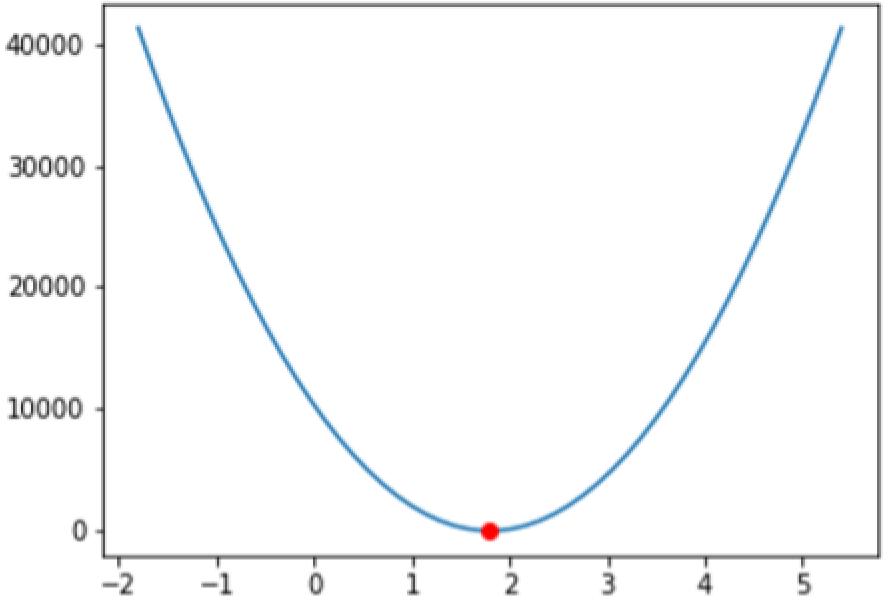
Cost Function를 타고 내려가다보면
결국에는 최적의 w를 찾을 수 있다

Cost Function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, h(x^{(i)})\right)$$

화씨(fahrenheit, F)





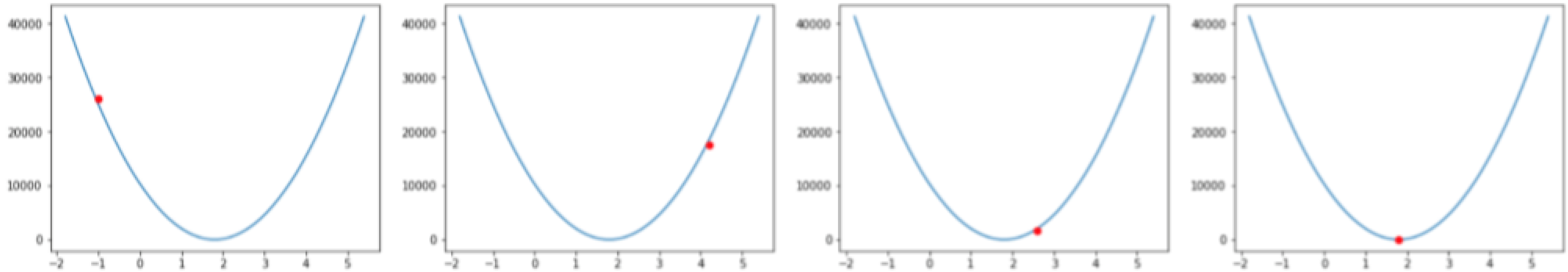
섭씨(celsius, C)

w = The weight

Linear Regression

Gradient Descent

여기서 중요한 건 Cost Function의 기울기이다
이를 구하기 위해서는 Cost Function을 미분할 수 있어야한다

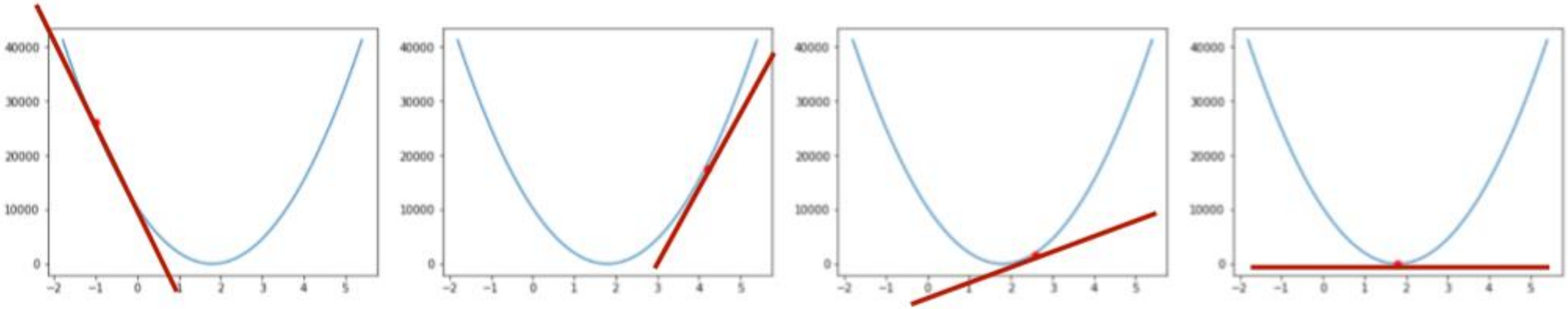


	w = -1.0	w = 4.2	w = 2.6	w = 1.8
속도	기울기가 가파르므로 w가 더 많이 이동해야 한다	마찬가지로 기울기가 가파르므로 w가 더 많이 이동해야 한다	기울기가 완만하므로 w가 적게 이동해야 한다	기울기가 평평하므로 w를 바꿔줄 필요가 없다

Linear Regression

Gradient Descent

여기서 중요한 건 Cost Function의 기울기이다
이를 구하기 위해서는 Cost Function을 미분할 수 있어야한다

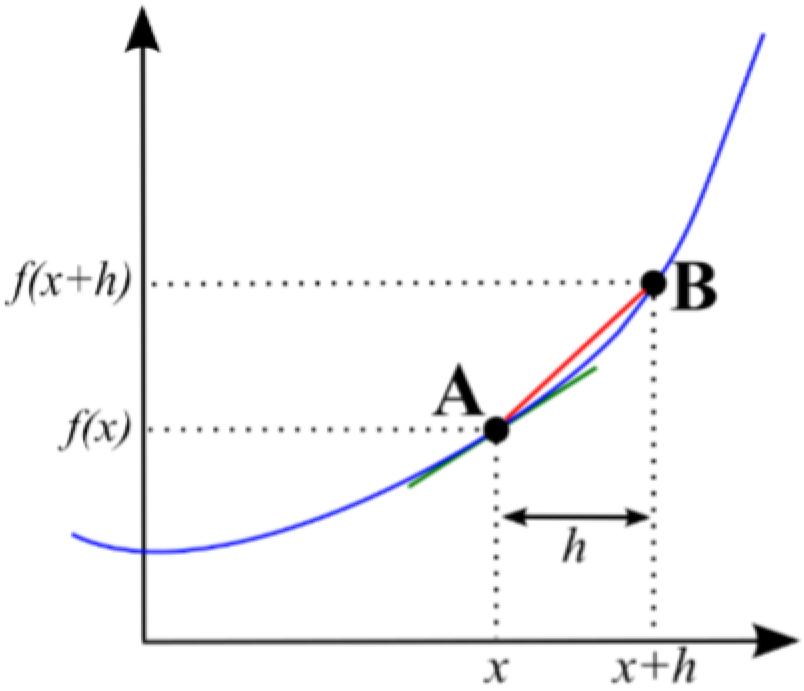


	w = -1.0	w = 4.2	w = 2.6	w = 1.8
속도	기울기가 가파르므로 w가 더 많이 이동해야 한다	마찬가지로 기울기가 가파르므로 w가 더 많이 이동해야 한다	기울기가 완만하므로 w가 적게 이동해야 한다	기울기가 평평하므로 w를 바꿔줄 필요가 없다
방향	빨간 공이 오른쪽으로 내려가므로 w를 늘려줘야 한다	빨간 공이 왼쪽으로 내려가므로 w를 줄여줘야 한다	빨간 공이 왼쪽으로 내려가므로 w를 줄여줘야 한다	빨간 공이 평평한 위치에 있으므로 w를 바꿔줄 필요가 없다

Linear Regression

- Gradient Descent

미분(Differential)이란 순간 변화율(또는 순간 기울기)을 뜻합니다.
순간 변화율은 x의 변화량이 아주 작을 때(0에 극한으로 가까워 질 때)의 평균 변화율을 의미합니다.
또는 접선의 기울기를 의미합니다.



$$f'(x) = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \frac{dy}{dx} = \frac{d}{dx} f(x)$$

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

Linear Regression

- Gradient Descent

기본적인 미분 공식을 알아봅니다.

$$(1)\ \ f(x) = C \ \rightarrow \ f'(x) = 0$$

$$f(x) = 1 \ \rightarrow \ f'(x) = 0$$

$$(2)\ \ f(x) = x^n \ \rightarrow \ f'(x) = nx^{n-1}$$

$$f(x) = x^2 \ \rightarrow \ f'(x) = 2x$$

$$(3)\ \ f(x) = nx^m \ \rightarrow \ f'(x) = nmx^{m-1}$$

$$f(x) = 3x^{100} \ \rightarrow \ f'(x) = 300x^{99}$$

Linear Regression

- Gradient Descent

기본적인 미분 공식을 알아봅니다.

$$(1)\quad f(x) = C \;\rightarrow\; f'(x) = 0$$

$$f(x) = 1 \;\rightarrow\; f'(x) = 0$$

$$(2)\quad f(x) = x^n \;\rightarrow\; f'(x) = nx^{n-1}$$

$$f(x) = x^2 \;\rightarrow\; f'(x) = 2x$$

$$(3)\quad f(x) = nx^m \;\rightarrow\; f'(x) = nmx^{m-1}$$

$$f(x) = 3x^{100} \;\rightarrow\; f'(x) = 300x^{99}$$

$$(4)\quad \frac{d}{dx}(f(x) \pm g(x)) = \frac{d}{dx}f(x) \pm \frac{d}{dx}g(x) = f'(x) \pm g'(x)$$

$$f(x) = 3x^{100},\; g(x) = -5x$$

$$\frac{d}{dx}(f(x) + g(x)) = 300x^{99} - 5$$

Linear Regression

- Gradient Descent

두 개 이상의 함수로 구성된 합성함수에 대한 미분 공식을 연쇄 법칙(Chain Rule)이라고 합니다.
합성함수는 여러 개의 함수를 합성하는 것을 의미합니다.

$$f \circ g = (f \circ g)(x) = f(g(x))$$

$$f(x) = x^3, \quad g(x) = 2x + 1$$

$$\rightarrow \ (f \circ g)(x) = f(g(x)) = f(2x+1) = (2x+1)^3$$

Linear Regression

- Gradient Descent

두 개 이상의 함수로 구성된 합성함수에 대한 미분 공식을 연쇄 법칙(Chain Rule)이라고 합니다.
합성함수의 미분은 각 함수 미분의 곱으로 이루어 진다는 특징이 있습니다. 합성함수 미분은 아래와 같습니다.

$$(f \circ g)' = (f' \circ g) \cdot g' = f'(g(x))g'(x)$$

$$f(x) = x^3, \quad g(x) = 2x + 1$$

$$\rightarrow (f \circ g)' = f'(g(x))g'(x)$$
$$= 3(2x+1)^2 * 2 = 6(2x+1)^2$$

Linear Regression

- Gradient Descent

편미분(Partial Derivative)은 미분하는 변수를 제외한 나머지 변수를 상수로 취급하여 미분하는 것입니다.

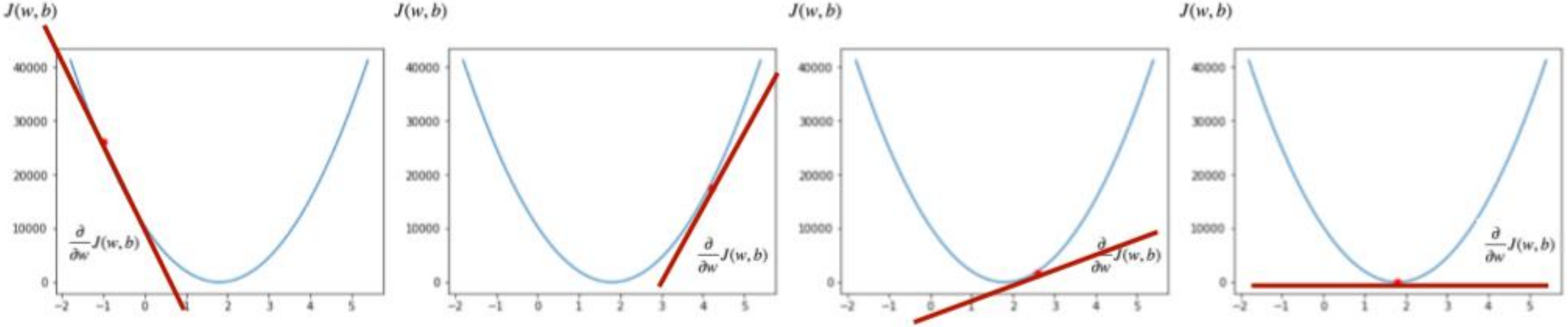
$$f',\ \frac{dy}{dx} \quad \rightarrow \quad f'_x,\ f'_y,\ \frac{\partial f}{\partial x},\ \frac{\partial f}{\partial y}$$

$$f(x, y) = x^2 - xy + y^2$$

$$\frac{\partial f}{\partial x} = 2x - y$$
$$\frac{\partial f}{\partial y} = -x + 2y$$

Linear Regression

1) Cost Function을 파라미터(w, b)로 편미분하여 Gradient를 구하고,
2) 파라미터를 업데이트 한다. 4) 1~2번을 반복하면 언젠가는 Loss Function이 0에 근접해지면서
적합한 weight를 찾을 수 있다



	w = -1.0	w = 4.2	w = 2.6	w = 1.8
속도	기울기가 가파르므로 w를 크게 업데이트 해야한다	마찬가지로 기울기가 가파르므로 w를 크게 업데이트 해야한다	기울기가 완만하므로 w를 작게 업데이트 해야한다	기울기가 평평하므로 w를 업데이트할 필요가 없다
방향	기울기가 우하단으로 내려가므로 (=미분하면 마이너스로 나온다) w를 늘려줘야 한다	기울기가 좌상단으로 올라가므로 (=미분하면 플러스가 나온다) w를 줄여줘야 한다	기울기가 좌상단으로 올라가므로 (=미분하면 플러스가 나온다) w를 줄여줘야 한다	기울기가 평평하므로 (=미분하면 0이 나온다) w를 바꿔줄 필요가 없다

결론 -> $w = w - \lambda \frac{\partial}{\partial w} J(w,b)$

(λ = learning_rate)

Linear Regression

$$L(y, h(x)) = \frac{1}{2}\left(h(x) - y\right)^2$$

$$\frac{\partial}{\partial w} L(y, h(x)) = ?$$

$$= \frac{\partial}{\partial w}\left(\frac{1}{2}\left(h(x) - y\right)^2\right)$$

$$= \frac{1}{2}\frac{\partial}{\partial w}\left(h(x) - y\right)^2$$

$$= \boxed{\frac{1}{2}2}(h(x) - y)\frac{\partial}{\partial w}\left(h(x) - y\right)$$

일단 합성함수의 바깥 부분을 먼저 편미분해준다.
편미분 후에 나오는 2를 통해 1/2를 없앨 수 있다.

이후에는 간단한 미분 공식과
합성함수의 미분(chain rule)을 이용하면 간단하다

합성함수의 미분(chain rule)

$y = f(g(x))$ 일때

$y' = f'(g(x))g'(x)$

참고자료 https://goo.gl/P7kFWW

그러므로

$f(x) = x^2$

$g(x) = (h(x) - y)$ 라고 가정하면

$(f \circ g)(x) = (h(x) - y)^2$

합성 함수 미분을 사용할 수 있다.

Linear Regression

$$L(y, h(x)) = \frac{1}{2}\left(h(x) - y\right)^2$$

$$\frac{\partial}{\partial w} L(y, h(x)) = ?$$

$$= \frac{\partial}{\partial w}\left(\frac{1}{2}\left(h(x) - y\right)^2\right)$$

$$= \frac{1}{2}\frac{\partial}{\partial w}\left(h(x) - y\right)^2$$

$$= \frac{1}{2} 2(h(x) - y)\frac{\partial}{\partial w}\left(h(x) - y\right)$$

$$= (h(x) - y)\frac{\partial}{\partial w}\left(h(x) - y\right)$$

$$= (h(x) - y)\boxed{\frac{\partial}{\partial w}\left(wx + b - y\right)}$$

편미분은 자기 자신을 제외한 나머지는 상수로 가정한다.
그러므로 w를 제외한 나머지는 상수이며, wx + b - y를 w로 편미분하면 x만 남는다.

이후에는 간단한 미분 공식과
합성함수의 미분(chain rule)을 이용하면 간단하다

합성함수의 미분(chain rule)

$y = f(g(x))$ 일때

$y' = f'(g(x))g'(x)$

참고자료 https://goo.gl/P7kFWW

그러므로

$f(x) = x^2$

$g(x) = (h(x) - y)$ 라고 가정하면

$(f \circ g)(x) = (h(x) - y)^2$

합성 함수 미분을 사용할 수 있다.

Linear Regression

이후에는 간단한 미분 공식과
합성함수의 미분(chain rule)을 이용하면 간단하다

$$L(y, h(x)) = \frac{1}{2}\left(h(x) - y\right)^2$$

$$\frac{\partial}{\partial w} L(y, h(x)) = ?$$

$$\begin{aligned}
&= \frac{\partial}{\partial w}\left(\frac{1}{2}\left(h(x) - y\right)^2\right) \\
&= \frac{1}{2}\frac{\partial}{\partial w}\left(h(x) - y\right)^2 \\
&= \frac{1}{2}2(h(x) - y)\frac{\partial}{\partial w}\left(h(x) - y\right) \\
&= (h(x) - y)\frac{\partial}{\partial w}\left(h(x) - y\right) \\
&= (h(x) - y)\frac{\partial}{\partial w}\left(wx + b - y\right) \\
&= (h(x) - y)x
\end{aligned}$$

결론

$$\boxed{\frac{\partial}{\partial w} L(y, h(x)) = (h(x) - y)x}$$

$$\begin{aligned}
\frac{\partial}{\partial w} J(w, b) &= \frac{\partial}{\partial w}\left(\frac{1}{m}\sum_{i=1}^{m} L(y^{(i)}, h(x^{(i)}))\right) \\
&= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w} L(y^{(i)}, h(x^{(i)})) \\
&= \boxed{\frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})x^{(i)}}
\end{aligned}$$

$$\boxed{\frac{\partial}{\partial b} J(w, b) = \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})}$$

Linear Regression

우리가 지금까지 작성한 코드를 살펴보면
위 Loss Function의 편미분 버전이 포함되어 있다는 것을 알 수 있다

```
num_epoch = 100000
learning_rate = 0.0003

w = np.random.uniform(low=-1.0, high=1.0)
b = np.random.uniform(low=-1.0, high=1.0)

for epoch in range(num_epoch):
    y_predict = w * X + b

    w = w - learning_rate * ((y_predict - y) * X).mean()
    b = b - learning_rate * (y_predict - y).mean()
```

$$h(x) = wx + b$$

Linear Regression

우리가 지금까지 작성한 코드를 살펴보면
위 Loss Function의 편미분 버전이 포함되어 있다는 것을 알 수 있다

```
num_epoch = 100000
learning_rate = 0.0003

w = np.random.uniform(low=-1.0, high=1.0)
b = np.random.uniform(low=-1.0, high=1.0)

for epoch in range(num_epoch):

    y_predict = w * X + b

    w = w - learning_rate * ((y_predict - y) * X).mean()
    b = b - learning_rate * (y_predict - y).mean()
```

$$\frac{\partial}{\partial w} L(y, h(x)) = (h(x) - y)x$$

Linear Regression

우리가 지금까지 작성한 코드를 살펴보면
위 Loss Function의 편미분 버전이 포함되어 있다는 것을 알 수 있다

```
num_epoch = 100000
learning_rate = 0.0003

w = np.random.uniform(low=-1.0, high=1.0)
b = np.random.uniform(low=-1.0, high=1.0)

for epoch in range(num_epoch):

    y_predict = w * X + b
                         λ
    w = w - learning_rate * ((y_predict - y) * X).mean()
    b = b - learning_rate * (y_predict - y).mean()
```

Linear Regression

우리가 지금까지 작성한 코드를 살펴보면
위 Loss Function의 편미분 버전이 포함되어 있다는 것을 알 수 있다

```
num_epoch = 100000
learning_rate = 0.0003

w = np.random.uniform(low=-1.0, high=1.0)
b = np.random.uniform(low=-1.0, high=1.0)

for epoch in range(num_epoch):

    y_predict = w * X + b

    w = w - learning_rate * ((y_predict - y) * X).mean()
    b = b - learning_rate * (y_predict - y).mean()
```

$$\frac{\partial}{\partial b} J(w,b) = \frac{1}{m}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})$$

Linear Regression

우리가 지금까지 작성한 코드를 살펴보면
위 Loss Function의 편미분 버전이 포함되어 있다는 것을 알 수 있다

```
num_epoch = 100000
learning_rate = 0.0003

w = np.random.uniform(low=-1.0, high=1.0)
b = np.random.uniform(low=-1.0, high=1.0)

for epoch in range(num_epoch):

    y_predict = w * X + b

    w = w - learning_rate * ((y_predict - y) * X).mean()
    b = b - learning_rate * (y_predict - y).mean()
```

$$w = w - \lambda \frac{\partial}{\partial w} J(w, b)$$