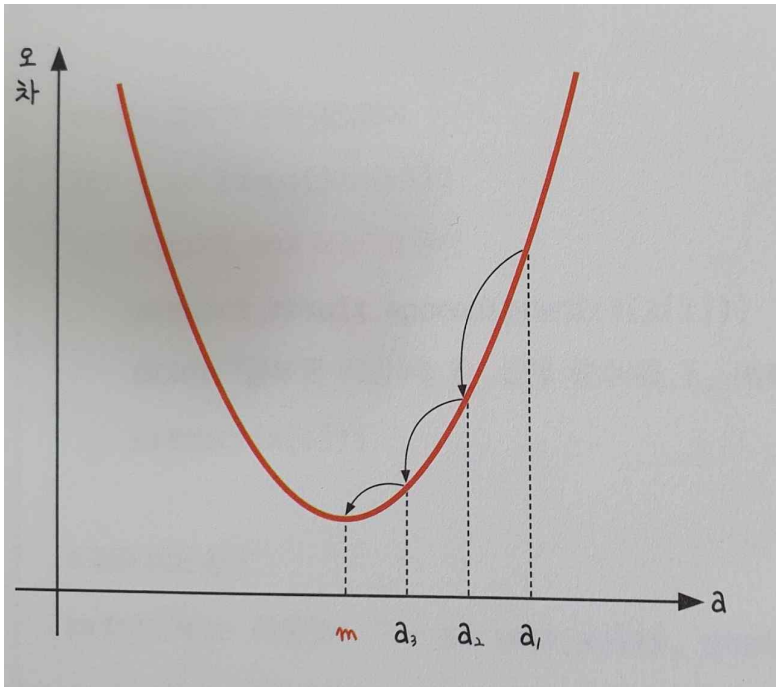
경사 하강법(gradient descent)

기울기를 너무 크게 잡거나 작게 잡으면 오차가 커짐!
기울기 a와 오차 사이에는 이렇게 상관관계가 있음

a가 무한대로 커지면 -> 오차 무한대로 커짐
a가 무한대로 작아지면 -> 오차 무한대로 커짐
-> 이러한 관계는 이차 함수 그래프로 표현됨

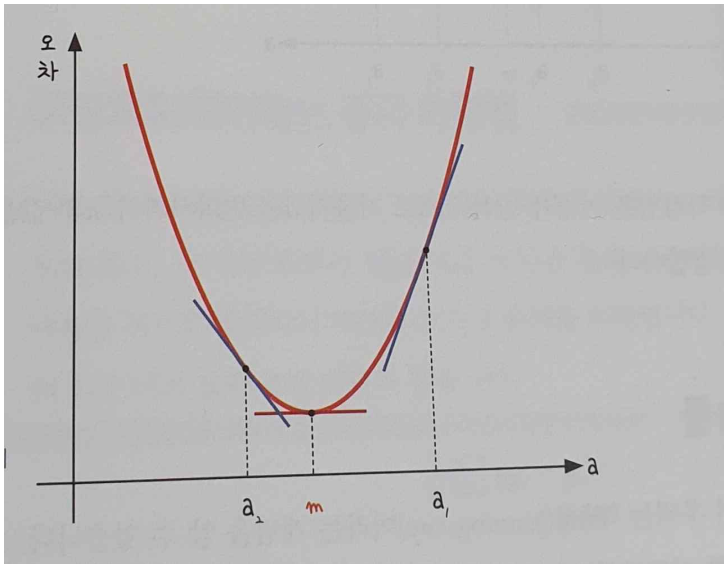


기울기 a가 m 위치에 있을 때 오차가 가장 작음
m의 값을 구하려면 임의의 한 점(a_1)을 찍고 이 점을 m에 가까운 쪽으로 점점 이동($a_1 \rightarrow a_2 \rightarrow a_3$)시키는 과정이 필요함. 그러려면 a_1의 값보다 a_2의 값이 m에 더 가깝고 a_2의 값보다 a_3가 m에 더 가깝다는 것을 컴퓨터가 알아야 함.

미분 기울기를 이용하는 **경사 하강법(gradient descent)** 이용!!

1. 경사 하강법의 개요

미분 - 한 점에서의 순간 기울기



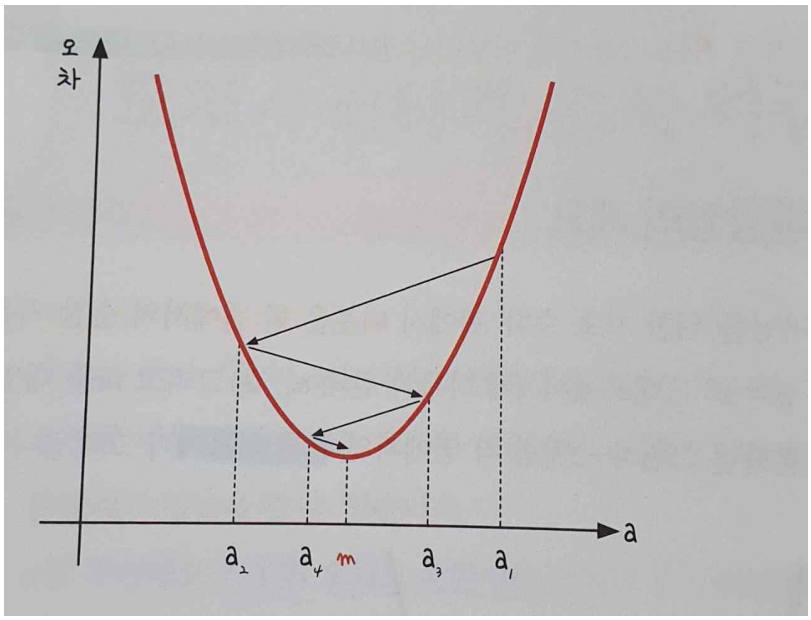
여기서 눈여겨 봐야 할 것은 우리가 찾는 최솟값 m에서의 순간 기울기

-> 미분 값(기울기)이 0인 지점 (기울기가 x축과 평행한 선)을 찾아야 함!!

① a_1에서 미분을 구한다.

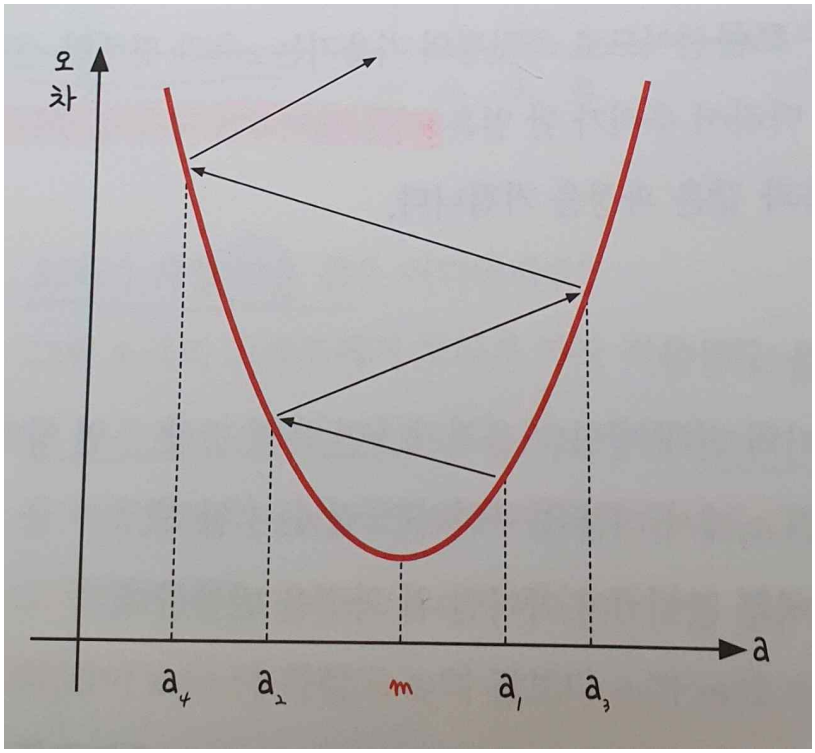
② 구해진 기울기의 반대 방향(기울기가 +면 음의 방향, -면 양의 방향)으로 얼마간 이동시킨 a_2에서 미분을 구한다.

③ 위에서 구한 미분 값이 0이 아니면 위 과정을 반복한다.



2. 학습률(learning rate)

기울기의 부호를 바꿔 이동시킬 때 적절한 거리를 찾지 못해 너무 멀리 이동시키면 a 값이 한 점으로 모이지 않고 위로 치솟아 버림



이동 거리를 정해 주는 것이 바로 학습률!
최적의 학습률을 찾는 것이 중요한 최적화 과정 중 하나!

경사 하강법 - 오차의 변화에 따라 이차 함수 그래프를 만들고 적절한 학습률을 설정해 미분값이 0인 지점을 구하는 것

y절편 b의 값도 너무 크면 오차가 커지고 너무 작아도 오차가 커지므로 최적의 b 값을 구할 때 역시 경사 하강법을 사용함.

3. 코딩으로 확인하는 경사 하강법

오차 최솟값을 구하기 위해서는 이차 함수에서 미분을 해야 하고, 그 이차 함수는 평균 제곱 오차를 통해 나온다는 것이다.

$$\text{평균 제곱 오차} = \frac{1}{n}\sum(y_i - \hat{y_i})^2$$

$\hat{y_i} = ax_i + b$ 이므로 다음과 같다.

$$\frac{1}{n}\sum(y_i - (ax_i + b))^2$$

이 값을 미분할 때 우리가 궁금한 것은 a와 b라는 것에 주의!!
식 전체를 미분하는 것이 아니라 필요한 값을 중심을 미분해야 함. -> 편미분 이용

$$a\text{로 편미분 한 결과} = \frac{2}{n}\sum(ax_i + b - y_i)x_i$$

$$b\text{로 편미분 한 결과} = \frac{2}{n}\sum(ax_i + b - y_i)$$

(실습코드는 github 참고)

기울기 a가 2.3에 수렴하고 y 절편 b의 값이 79에 수렴하는 과정을 볼 수 있음 (최소 제곱법 사용했을 때와 결과가 같음)

4. 다중 선형 회귀란

더 정확한 예측을 하려면 추가 정보를 입력해야 하며, 정보를 추가해 새로운 예측 값을 구하려면 변수의 개수를 늘려 **다중 선형 회귀**를 만들어 주어야 합니다.

ex)

공부한 시간(x_1)	2	4	6	8
과외 수업 횟수(x_2)	0	4	2	3
성적(y)	81	93	91	97

2개의 독립 변수 x_1, x_2가 생겼음. 종속 변수 y를 만들 경우 기울기를 두 개 구해야 함

$$y = a_1x_1 + a_2x_2 + b$$

-> 경사 하강법 똑같이 적용!!

5. 코딩으로 확인하는 다중 선형 회귀

(실습코드는 github 참고)