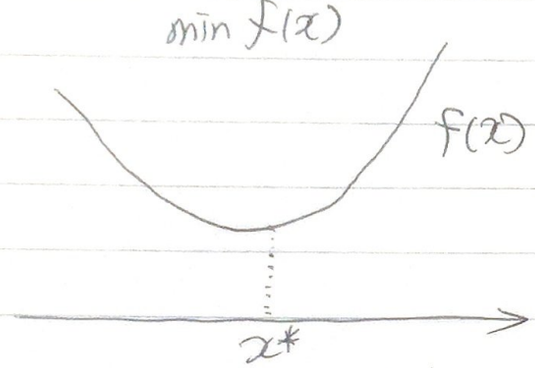
Week 9. 경사하강법과 최소제곱문제의 해

(?) 주어진 함수의 최적해 계산
① 도함수가 0이 되는 임계점(critical point)을 구한다
② 임계점이 극대/극소/최대/최소가 되는지 판단
// 함수가 복잡하면 임계점 구하기 쉽지 않다
→ 경사하강법 사용 ; 딥러닝 가중치 업데이트 핵심 알고리즘

9.1. 경사하강법 (Gradient Descent Algorithm)

— 최소제곱문제의 근사해를 수치적(numerical)으로 얻는다.

"미분가능한 일변수 함수 $f(x)$ 최솟값 구하기"



Fermat의 임계점 정리를 위해, 함수의 최솟값을 반환하는 ($=$ 최적해, optimal solution) x^*는 다음을 만족한다.

$$f'(x^*) = 0$$

→ 방정식 $f'(x) = 0$ 만족하는 해집합을 구하여 최적해를 판단
→ But 함수가 복잡하면 임계점 구하는 것도 쉽지 않다
→ 수치 최적화 방법인 경사하강법(gradient descent method)으로 임계점을 근사

기본 idea 함수의 기울기(경사)를 구하여
기울기가 낮은 쪽으로 계속 이동하면서 극값이 될 때까지 반복

경사하강법 (1) 초기 근사해 x_1, 허용오차(tolerance) $0 \le \epsilon < 1$, 학습률(learning r
를 부여한다. $k := 1$이라고 한다.

$$x_{k+1} = x_k - \eta g_k$$

(2) $g_k = f'(x_k)$를 계산한다. 만일 $|g_k| \le \epsilon$이면, 알고리즘을 멈춘
(3) $x_{k+1} = x_k - \eta g_k$, $k := k+1$이라 두고 (2)로 돌아간다.

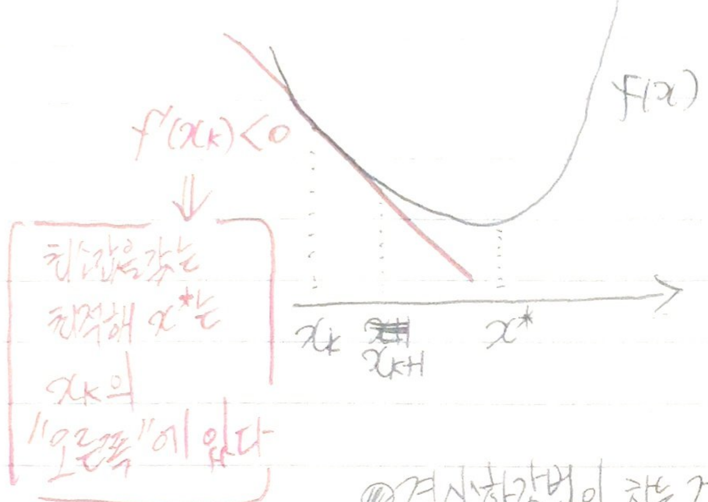
(설명)

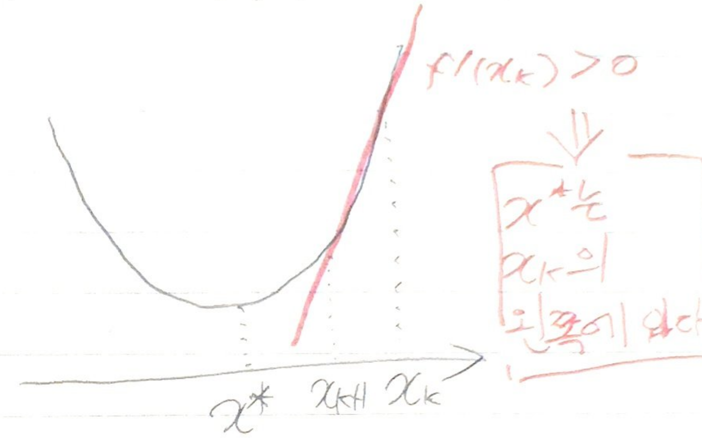
① $|g_k| \le \epsilon$ (threshold error) → $f'(x_1) \approx 0$ (으로 간주한다)

∴ 1st step에서 근사해는 x_1이 된다

② 이러한 x를 계속 찾아낸다, $x_1, x_2, x_3, \cdots x_k$

③ 우리는 이중 적절한 x_k 또는 극한값 x^*에서 $f'(x)=0$을 만족하길 기대한다





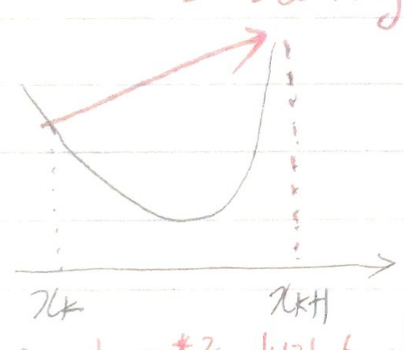
경사하강법이 하는 것

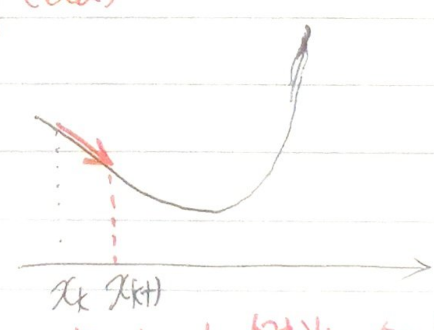
$f(x_1) > f(x_2) > f(x_3) \cdots$ 가 만족되도록

x_1, x_2, x_3을 찾아가는 과정을 수행

* 얼마만큼씩 이동하는가 — $x_k \xrightarrow{next} x_{k+?}$

= Learning Rate, η (eta)





η 너무 크면 x^*을 지나칠 수 있다

η 너무 작으면 수렴하는 속도가 너무 느려진다

"적절한 η를 이용해서 $x_{k+1} = x_k - \eta g_k$ 에 따라 x_{k+1} 결정"

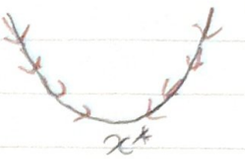
* 적절한 learning rate 결정은 중요하지만 여기서는 다루지 않음
* 학습률은 대개 10^{-6}에서 1 사이 범위에서 지정
* 초기 학습률로는 $\eta = 0.1$ 또는 $\eta = 0.01$이 주로 사용

Additional Section
: 경사하강법

Gradient Descent Algorithm

: 어떤 모델에 대한 비용(cost)을 최소화시키는 알고리즘
: 머신러닝 / 딥러닝의 가중치 (weight) 최적해 구할 때 널리 사용

비유개념) 함수의 기울기 (경사)를 구하여,
기울기가 낮은 쪽으로 계속 이동시켜서
극값에 이를 때까지 반복한다.



"제약조건이 없는 최적화 (unconstrained optimization)"

$\min_{x \in \mathbb{R}^n} f(x)$

↳ $\nabla f(x^*) = 0$

↳ $\nabla f(x) = 0$ solution set

"목적함수 f가 비선형인 경우"

iterative method

- k번째 추정해 x_k 또는 극한값 $(x_k \to) x^*$에서 $\nabla f(x) = 0$ 만족하도록

$$x_{k+1} = x_k + \alpha_k d_k$$

- d_k: 탐색방향 / search direction
- α_k: 학습률 / learning rate / step-size

"x_k 상에서 d_k 방향으로 α_k 만큼 이동하여 x_{k+1}을 생성한다"

⑩ 보통 d_k (방향)는 함수값이 감소하는 방향으로 정한다

$$D_{d_k} f(x_k) = \nabla f(x_k) \cdot d_k = d_k^T \nabla f(x_k) < 0$$

* 이러한 d_k는 하강방향 (descent direction) 이라고 함
* d_k 방향으로 움직인다 → f 감소가 보장 → α_k는 $f(x_k) > f(x_{k+1})$ 하도

step-size α_k를 선택하는 방법

① 반직선 $x = x_k + \alpha d_k \ (\alpha \geq 0)$ 상에서
함수 f의 값이 가장 작게 되는 α_k를 찾는다.

$$f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k)$$

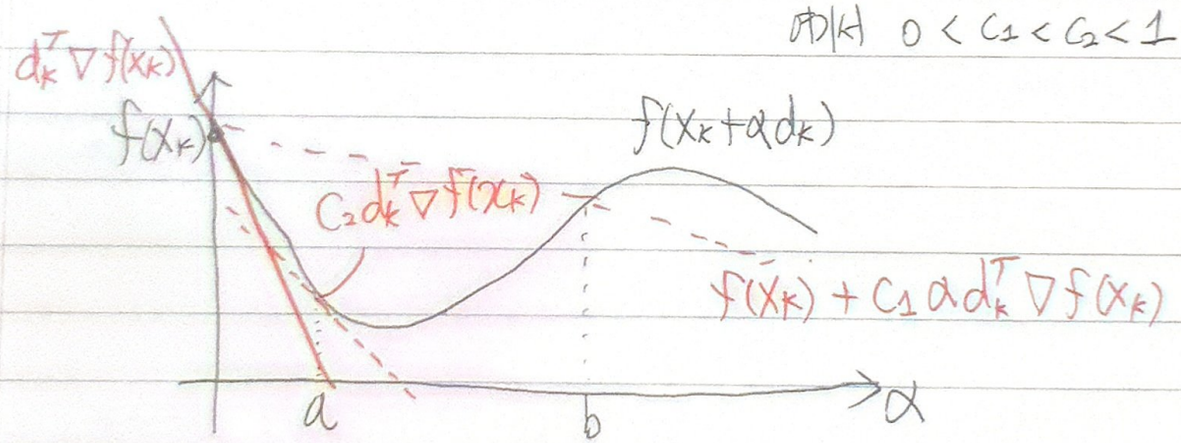
↳ **Exact Line Search**
; 이때의 α_k를 optimal step-size라 한다.
; 대체 cost가 높아서 잘 사용하지 않는다.

② 만일 $\min_{\alpha \geq 0} f(x_k + \alpha d_k)$를 정확하게 풀지는 않지만
"함수값이 충분히 감소한다"를 보장하는 α_k를 선택하는 방법이 있다면,
exact line search를 피하여 cost를 줄일 수 있다.
즉, 다음 조건을 만족하는 α_k를 찾는다.

Wolfe condition ← $f(x_k + \alpha d_k) \leq f(x_k) + c_1 \alpha d_k^T \nabla f(x_k)$

Armijo condition ← $d_k^T \nabla f(x_k + \alpha d_k) \geq c_2 d_k^T \nabla f(x_k)$

여기서 $0 < c_1 < c_2 < 1$이다.



여기서 위의 부등식을 만족하는 α_k는 폐구간 $[a, b]$에서 택하면 된다.

* 관련 알고리즘 — Backtracking Line Search 등.
* 관련 서적 (강력)
(S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004
(http://stanford.edu/~boyd/cvxbook/)

9.2 응용 (최소제곱문제)

최소제곱문제도 역시 경사하강법으로 해결할 수 있다.

$\min E(u)$

// 다만 앞서 소개한 경사하강법은 독립변수가 하나인 일변수함수 $f(x)$에 대하여 구성
// 최소제곱문제는 독립변수가 최소 2개인 다변수함수 $E(u)$이므로 다음과 같이 변형

① 초기근사해 $u_1 \in R^n$, 허용오차 $0 \le \epsilon < 1$, 학습률 η 부여
$k := 1$
② $g_k = \nabla E(u_k)$ 계산한다. 만일 $\|g_k\| \le \epsilon$ 이면 stop
③ $u_{k+1} = u_k - \eta g_k$, $k := k+1$이라 두고 ②로 돌아간다.

* 스칼라 $x \longrightarrow$ 벡터 u
절대값 $|g| \longrightarrow$ 벡터 norm $\|g\|$
도함수 $f'(x) \longrightarrow$ (도함수역할) gradient $\nabla E(u)$

● 다변수함수 $z = f(x, y)$
좌표공간에서 점 (x, y, z)을 상정하고 (x, y)를 움직이면 → 하나의 곡면형성
이 곡면을 함수 $z = f(x, y)$의 graph라 부른다.

(일변수 f : 도함수 f' = 다변수 f : gradient)

n변수함수 $f(x_1, x_2, \cdots x_n)$에 대하여 gradient는 다음과 같다.

$$\text{grad} f(x_1, x_2, \cdots x_k) = \nabla f(x_1, x_2, \cdots x_n) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

* $\frac{\partial f}{\partial x}$ — "f를 변수 x에 대하여 편미분한다."
(x를 제외한 나머지를 상수취급)