02. 도함수 , 경사도

(derivate) (기울기, gradient)

↓ ↓

일변량 다변량

최적화와 도함수

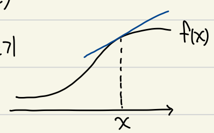
- 최적화

크게 하기 위해서는 부호만 바꿔주면 ok

- 목적함수의 값을 작게 하는 계획적 탐색 문제 → 계획점 (design point) 찾기
- 값을 변화시킬 때, 목적함수 값의 변화 (목적함수 값 적게 하는 점 택)
- 도함수 / 경사도 → x의 변화에 따른 목적함수 f의 변화 측정

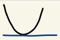
- 도함수

- 접선 (tangent line)
- 도함수 : 접선의 기울기



↓

접선의 기울기 = 0 ⇒ 최소점



- 선형근사 (linear approximation)

$$f(x+\underbrace{\Delta x}_{\text{작은 값}}) \approx f(x) + \underbrace{f'(x)}_{\text{기울기}}\Delta x$$

ex) $f(x) = x^2$, $f'(x) = 2x$, $x = -1$, $\Delta x = 0.01$

→ −1에서 −0.99로 이동할 때 함수값 변화 보고 싶음. 목적함수 값 얼만큼 변화?

$$f(-1+0.01) \approx f(-1) + f'(-1) \times 0.01$$

$$0.9801 \approx 1 + \underbrace{(-2)}_{} \times 0.01$$

↳ 기울기 0이 되면 목적함수 값이 줄어들지 X 그곳이 design point. (계획점, minimizer)

↓

기울기를 $f'(x)$ 이용해 직접 적용

선형근사

- $f(x) = x^2$
- $f(x+dx) = f(x) + f'(x)dx$
- dx는 델타 x 의미한다

```
x=-1
dx=0.001

f=x**2
df=2*x

f_dx=f+df*dx
print('f값, f_dx값')
print((f,f_dx))

print(' ')
print('dx 값 변화에 따른 f값과 f_dx값')
for dx in [1E-3,1E-4,1E-5,1E-8]:
    f_dx=f+df*dx
    print((f,f_dx))

print(' ')
print('f(-1)값')
def f(x):
    return x**2
print(f(-1))
print(' ')

print('df(-1)값')
def df(x):
    return 2*x

print(df(-1))
print(' ')
print('dx 값 변화에 따른 f(-1), f_dx(-1,dx), 두 값의 차이 절대값')
def f_dx(x,dx):
    return f(x)+df(x)*dx

for dx in [1E-3,1E-4,1E-5,1E-6]:
    print('%.4f, %.4f (%.4f)' %(f(-1),f_dx(-1,dx),abs(f(-1)-f_dx(-1,dx))))
```

df=2*x → 1차 미분한 도함수

print((f,f_dx)) → 참값 (1)에 근사함 (0.998), dx 작아질 수록 참값에 가까워짐.

for dx in [1E-3,1E-4,1E-5,1E-8] … → 수치미분의 기본이 되는 아이디어.

print(df(-1)) → 1차 미분한 $x=-1$ 에서의 도함수. 기울기 = −2

$x=-1$ 일 때 x^2 함수의 탄젠트 라인 기울기 −2

for dx in [1E-3,1E-4,1E-5,1E-6] … → 참값에 가까운 값으로 수렴.

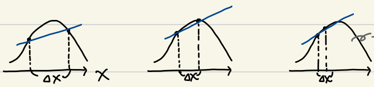
```
f값, f_dx값
(1, 0.998)

dx 값 변화에 따른 f값과 f_dx값
(1, 0.998)
(1, 0.9998)
(1, 0.99998)
(1, 0.99999998)

f(-1)값
1

df(-1)값
-2

dx 값 변화에 따른 f(-1), f_dx(-1,dx), 두 값의 차이 절대값
1.0000, 0.9980 (0.0020)
1.0000, 0.9998 (0.0002)
1.0000, 1.0000 (0.0000)
1.0000, 1.0000 (0.0000)
```

도함수 정의

1) Lagrange

: $f(x+\Delta x) \approx f(x) + f'(x)\Delta x$

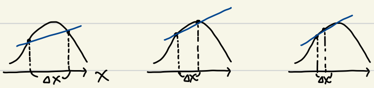
$\rightarrow f'(x) \equiv \dfrac{f(x+\Delta x) - f(x)}{\Delta x} = \dfrac{\Delta f(x)}{\Delta x}$



수렴된 극한값을 도함수라고 하자!

2) Leibniz

$f'(x) \equiv \dfrac{d f(x)}{d x}$ 단, $d = f(x+\Delta x) - f(x)$

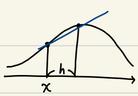


도함수의 극한 $(h \rightarrow 0)$

- h : 차분 값 (step size)

- 전방 차분 (forward difference)

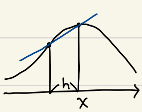
$f'(x) \equiv \lim_{h\to 0} \dfrac{f(x+h) - f(x)}{h}$



→ 계산적으로 정의식 직접 이용 무리 (why? 분모 = 0)

- 후방 차분 (backward difference)

$f'(x) \equiv \lim_{h\to 0} \dfrac{f(x) - f(x-h)}{h}$



⇓

후방 차분 값 = 전방 차분 값 → continuous function

만약, 후방 차분 ≠ 전방 차분 → 도함수 존재 X = 미분값 존재 X

⇒ 최적화에서 대부분 목적함수는 연속함수이다.

- 중심차분 (central difference)

: 양쪽에서 step-size 동일한 값으로 지정해 도함수 도출

$f'(x) \equiv \lim_{h\to 0} \dfrac{f(x+\frac{h}{2}) - f(x-\frac{h}{2})}{h}$

다차원 도함수

- 경사도 (기울기, gradient)

  - 일차원 도함수의 다차원으로의 일반화.

    : 한 점에서 어떤 방향으로 조금 변화시켰을 때, 함수값의 변화를 예측할 수 있는 함수의 국소 기울기.

  - 경사점 (gradient point)

    : 초평면 (hyperplane) 상의 가장 경사가 큰 방향에 있는 점.

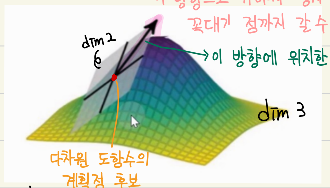
- 초평면

  - $w_1x_1 + w_2x_2 + \cdots + w_nx_n = b$

    → 변수가 3개면 4차원 공간 상에서의 hyperplane 정의

  - 임의의 vec $w = [w_1 \ \ w_2 \ \cdots \ w_n]$
  - 임의의 scalar b
  - $[a, b, c] = [a \ \ b \ \ c]^T$
  - $(n-1)$ 차원

이 방향으로 가야지 경사 가파르고, 꼭대기 점까지 갈 수 있음.



이 방향에 위치한 점은 경사점.

다차원 도함수의 계획점 후보

↓

dim 2 의 평면 어떻게 정의? → $w_1x_1 + \cdots + w_nx_n = b$로 정의

$\therefore w_1x_1 + w_2x_2 = b$

다차원 도함수

- 경사도

2차원 이면 plane으로 정의

- $\nabla f(x)$: 함수 f의 점 x에서의 경사도. (nx1) 벡터
- $\dfrac{\partial f(x)}{\partial x_T}$: 함수 f에 대한 변수 x_T의 편미분 값. ≠ 미분

$$\Rightarrow \nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \cdots, \frac{\partial f(x)}{\partial x_n} \right]$$

ex) $c = [2, 0]$ 에서의 $f(x) = x_1 \cdot \sin(x_2) + 1$ 의 경사도

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2} \right]$$

$$= [\sin(x_2), \; x_1 \cdot \cos(x_2)]$$

$$\nabla f(c) = [0, 2]$$

- 헷세 행렬 (Hessian)

: 함수의 국소 곡률 (local curvature)에 대한 정보

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix}_{n \times n}$$

→ 다변량 회귀분석에서, 최소 제곱 추정량 계산 위해 필요.

대각 원소 중심으로 비대각 원소 대칭 (대칭 행렬)

0보다 크거나 작은지 확인해 곡률 파악.

But, 최근 최적화 문제 2차 미분 사용X. 1차 미분 정보 이용.

다차원 도함수

다변량 함수에서도 gradient 존재 (초평면, 움직일 수 있는 방향 多)

- 방향 도함수 (directional derivative)

- $\nabla_s f(x)$: 점 x를 속도 s로 움직였을 때의 f(x) 변화율
- 전방 차분 : $\nabla_s f(x) \equiv \lim_{h \to 0} \dfrac{f(x+hs) - f(x)}{h}$

→ x : n개의 원소로 구성된 vec

→ s : 방향 vec , n×1

- 후방차분 : $\nabla_s f(x) \equiv \lim_{h \to 0} \dfrac{f(x) - f(x-hs)}{h}$

⇒ 전방차분 = 후방차분 이면, f(x)는 연속

- 중심 차분 : $\nabla_s f(x) \equiv \lim_{h \to 0} \dfrac{f(x+\frac{h \cdot s}{2}) - f(x-\frac{h \cdot s}{2})}{h}$

- $\nabla_s f(x) = \nabla f(x)^T \cdot s$

ex) 점 $x = [1, 0]$에서 방향 $s = [-1, -1]$로의 함수

$f(x) = x_1 \cdot x_2$ 의 방향 도함수

$$\nabla f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2} \right] = [x_2, x_1]$$

$$\nabla_s f(x) = \nabla f(x)^T s = [0 \;\; 1] \begin{bmatrix} -1 \\ -1 \end{bmatrix} = -1$$

└ 함수 f값이 적어지는 방향으로 움직인다.

*경사점

: 방향으로 가는 것 중, 기울기가 가장 가파르게 되는 방향

* 계획점 찾는 것이 목적!

다차원 도함수

- 수치 미분 (numerical differenciation)

: 매우 작은 h에 대한 유한 차분 방법

h(step size) adam으로 매우작은값 미리 정하기

- 전방 차분법 : $f'(x) \approx \dfrac{f(x+h)-f(x)}{h}$
- 후방 차분법 : $f'(x) \approx \dfrac{f(x)-f(x-h)}{h}$
- 중심 차분법 : $f'(x) \approx \dfrac{f(x+\frac{h}{2})-f(x-\frac{h}{2})}{h}$ 〈多사용〉

수치미분

- 전방차분
- 후방차분
- 중심차분

```python
# 전방차분
def forward_diff(f,x,h=1E-6):
    df_=(f(x+h)-f(x))/h
    return df_
print('전방차분시 x=-1값')
print(forward_diff(f,-1))
print(' ')

# 후방차분
def backward_diff(f,x,h=1E-6):
    df_=(f(x)-f(x-h))/h
    return df_
print('후방 차분시 x=-1값')
print(backward_diff(f,-1))
print(' ')

# 중심차분
def central_diff(f,x,h=1E-6):
    df_=(f(x+0.5*h)-f(x-0.5*h))/h
    return df_
print('중심차분시 x=-1값')
print(central_diff(f,-1))
print(' ')

print('h값 변화에 따른 df(-1)값, 중심차분 값, 차이의 절댓값')
for h in [1E-4,1E-6,1E-8]:
    cf=central_diff(f,-1,h)
    print('df: %.8f, cf: %.8f (%.8f)'%(df(-1), cf, abs(df(-1) - cf)))
print(' ')

print('x값 변화시 df값, 중심차분값, 차이의 절댓값')
for x in [-1E0,-5E-1,-2.5E-1,-1E-2,-1E-6]:
    cf=central_diff(f,x)
    print('df: %.8f, cf: %.8f (%.8f)' %(df(x), cf, abs(df(x)-cf)))
```

→ 함수 f를 -1에서 전방 차분 이용해 수치 미분 결과 참값(-2)에 근사하는 도함수 값

→ 참값(-2)에 가깝지만 전방차분과 다르게 -2보다 작은 값.

→ 전/후방 차분 보다 오차 훨씬 작아짐. 중심차분 多사용.

참값 / 근사값 / 오차

↳ step size(h) 값 작게 하기 : 1E-8에서 오차 증가. 무조건 작다고 좋은것 X

↳ x^2은 0에서 최저, 0에 가깝게!
중심차분 값 0에 가까워 지며 참값과 차이 X, 기울기=0 이 되는 점

```
전방차분시 x=-1값
-1.999999000079633

후방 차분시 x=-1값
-2.0000009999243673

중심차분시 x=-1값
-2.0000000000575113

h값 변화에 따른 df(-1)값, 중심차분 값, 차이의 절댓값
df: -2.00000000, cf: -2.00000000 (0.00000000)
df: -2.00000000, cf: -2.00000000 (0.00000000)
df: -2.00000000, cf: -1.99999999 (0.00000001)

x값 변화시 df값, 중심차분값, 차이의 절댓값
df: -2.00000000, cf: -2.00000000 (0.00000000)
df: -1.00000000, cf: -1.00000000 (0.00000000)
df: -0.50000000, cf: -0.50000000 (0.00000000)
df: -0.02000000, cf: -0.02000000 (0.00000000)
df: -0.00000200, cf: -0.00000200 (0.00000000)
```

다차원 도함수

- 자동 미분 (Automatic differentiation)

: AI, DL 알고리즘, gradient 계산, 복잡한 다변량 목적함수 → 자동미분

- 연쇄 법칙 (chain rule) ← 복잡한 함수 간단히 미분

: $\dfrac{d}{dx}f(g(x)) = \dfrac{d}{dx}(f\circ g)(x) = \dfrac{df}{dg}\cdot\dfrac{dg}{dx}$

ex) $f(a,b) = \ln(ab+\max(a,2))$

$$\frac{\partial f}{\partial a} = \frac{\partial}{\partial a}\ln(ab+\max(a,2))$$

$\ln x \xrightarrow{미분} \frac{1}{x}$

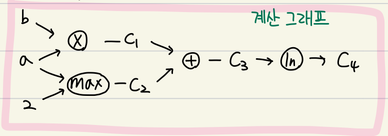
$$= \frac{1}{(ab+\max(a,2))}\cdot\frac{\partial}{\partial a}(ab+\max(a,2))$$

$$= \frac{1}{(ab+\max(a,2))}\left[\frac{\partial(ab)}{\partial a} + \frac{\partial(\max(a,2))}{\partial a}\right]$$

$$= \frac{1}{(ab+\max(a,2))}\left[\left(b\cdot\cancelto{1}{\frac{\partial a}{\partial a}} + a\cdot\cancelto{0}{\frac{\partial b}{\partial a}}\right) + \left(\cancelto{0}{\frac{\partial 2}{\partial a}}(2>a) + \cancelto{1}{\frac{\partial a}{\partial a}}(2<a)\right)\right]$$

$$= \frac{1}{(ab+\max(a,2))}\left[b+(2<a)\right]$$

) $f(a,b) = \ln(ab+\max(a,2))$

⇒ 

- 전방 누적법 (forward accumulation)

간단한 문제를 복잡한 문제로 바꿔 가는 것.

$$\frac{df}{dx} = \frac{df}{dc_4}\cdot\frac{dc_4}{dx} = \frac{df}{dc_4}\left(\frac{dc_4}{dc_3}\cdot\frac{dc_3}{dx}\right)$$

$$= \frac{df}{dc_4}\left(\frac{dc_4}{dc_3}\left(\frac{dc_3}{dc_2}\cdot\frac{dc_2}{dx} + \frac{dc_3}{dc_1}\cdot\frac{dc_1}{dx}\right)\right)$$

⇒ 계산 그래프 : $\dot{a} = \dfrac{\partial a}{\partial a}$, $\dot{b} = \dfrac{\partial b}{\partial b}$