머신 러닝·딥러닝에 필요한 수학 기초 with 파이썬

최적화 수법

Optimization

조준우
metamath@gmail.com

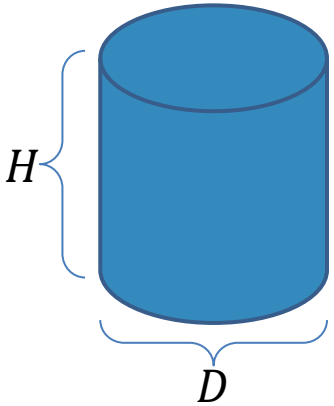
학습 개요

- 경사하강법
  - 목적함수
  - 경사도 벡터
  - 이동거리
  - 최속강하법, 공액경사법

- 경사하강법 코딩
  - scipy.optimize.minimize와 비교

캔 만들기

- 캔 설계 문제[arora]
  - D: 캔의 직경(cm), H: 캔의 높이(cm)
  - 캔의 용적은 최소 $600cm^3$
  - 직경 범위 : $3 \leq D \leq 8$, 높이 범위 : $8.5 \leq H \leq 17.5$
  - 표면적 → 재료비 → 표면적 최소 → 원가 최소
  - 주어진 조건을 만족하면서 표면적 최소가 되는 D, H는 얼마인가?



원둘레 : $2\pi r$

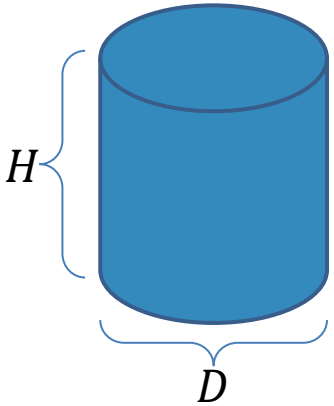
캔표면적 : $J = 2\pi\left(\frac{D}{2}\right)H + 2\left(\pi\frac{D^2}{4}\right) = \pi DH + \frac{\pi D^2}{2}$

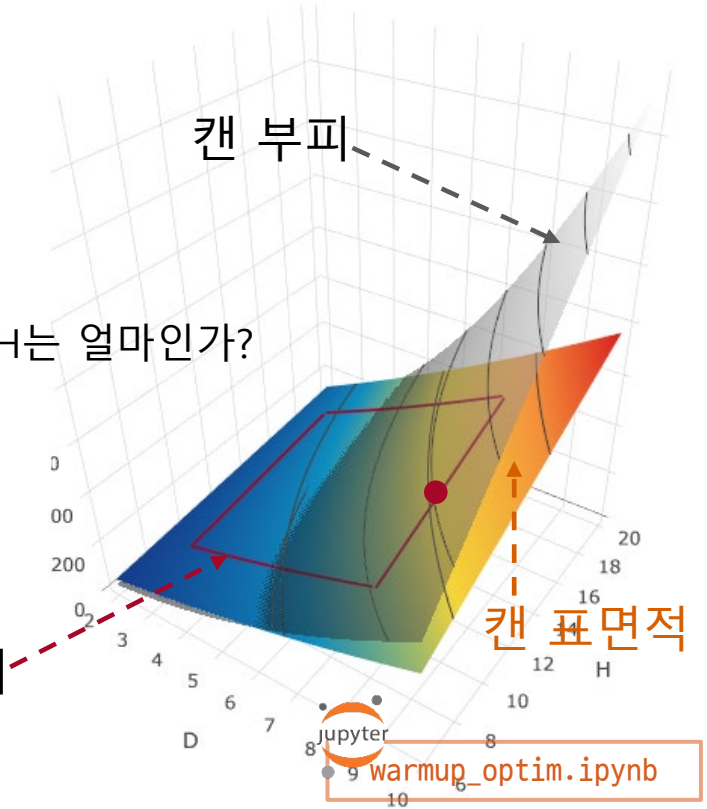
캔부피 : $\frac{\pi}{4}D^2H \geq 600$

D,H의 제약 $3 \leq D \leq 8,\ 8.5 \leq H \leq 17.5$

캔 만들기

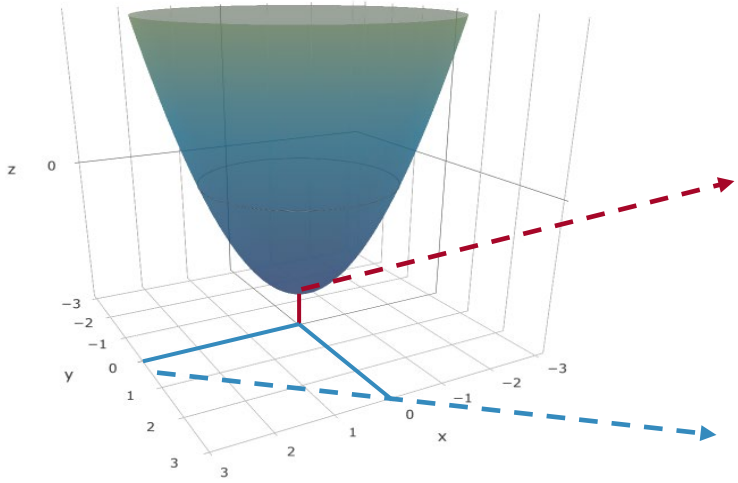
- 캔 설계 문제
  - D: 캔의 직경(cm), H: 캔의 높이(cm)
  - 캔의 용적은 최소 $600cm^3$
  - 직경 범위 : $3 \leq D \leq 8$, 높이 범위 : $8.5 \leq H \leq 17.5$
  - 표면적 → 재료비 → 표면적 최소 → 원가 최소
  - 주어진 조건을 만족하면서 표면적 최소가 되는 D, H는 얼마인가?





목적함수Objective function

- 최솟값을 찾으면 좋은 결과가 나오도록 목적에 맞게 설계된 함수
- 주어진 함수의 값을 제일 작게 하는 독립변수를 구하고 싶다!
- 목적에 맞게 설계된 함수
  - 목적함수objective, 비용함수cost, 손실함수loss

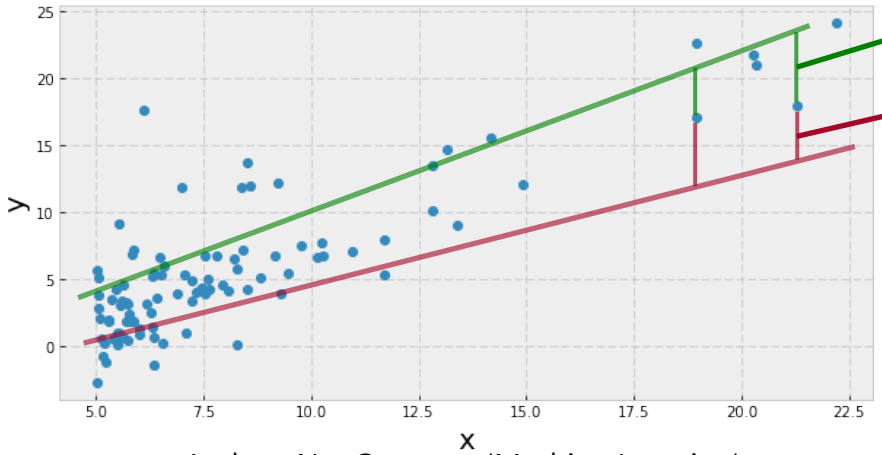


함숫값이 제일 작은 여기를 결과로 주는

x, y 값을 알고 싶다.

선형회귀와 목적함수

- 주어진 점(학습 데이터)와 임의의 선($\mathbf{w}$)의 간격(error)를 모두 더함
  - 작아지면 작아질 수 록 좋다.

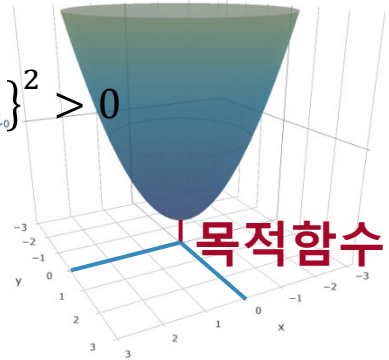


$$\frac{1}{2N}\sum_{n=1}^{N}\{h(x_n, \mathbf{w_g}) - y_n\}^2 > 0$$

ERROR

ERROR

$$\frac{1}{2N}\sum_{n=1}^{N}\{h(x_n, \mathbf{w_r}) - y_n\}^2 > 0$$



Andrew Ng, Coursera, 'Machine Learning'

실험용 목적함수

$$f_1(x_1, x_2) = x_1^2 + x_2^2$$

```
def f_1(x) :
    return   x[0]**2 + x[1]**2
```

$$f_2(x_1, x_2; a) = \frac{1}{20}(x_1 \cos a - x_2 \sin a)^2 + \frac{1}{2}(x_1 \sin a + x_2 \cos a)^2$$

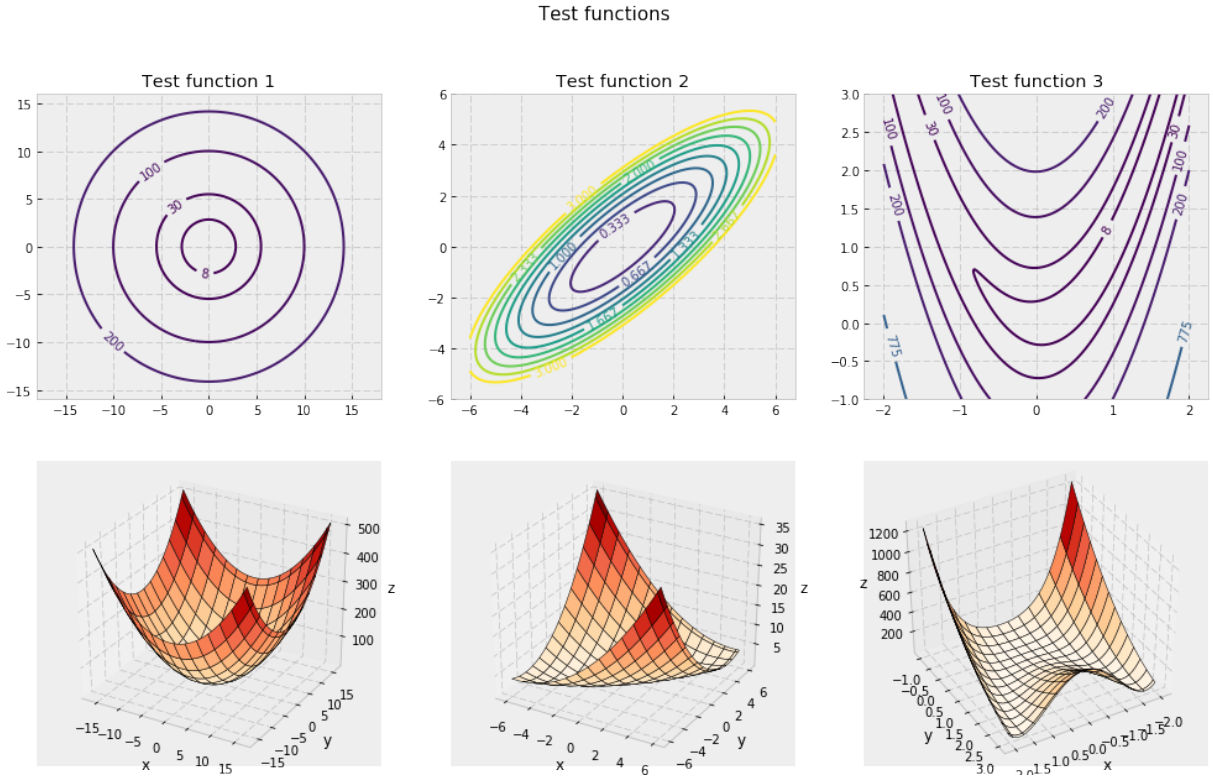
```
def f_2(x, a):
    return ( (x[0]*np.cos(a) - x[1]*np.sin(a))**2 / 20
           + (x[0]*np.sin(a) + x[1]*np.cos(a))**2 / 2 )
```

$$f_3(x_1, x_2) = 50(x_2 - x_1^2)^2 + (2 - x_1)^2$$

```
def f_3(x):
    return 50*(x[1]-x[0]**2)**2 + (2-x[0])**2
```

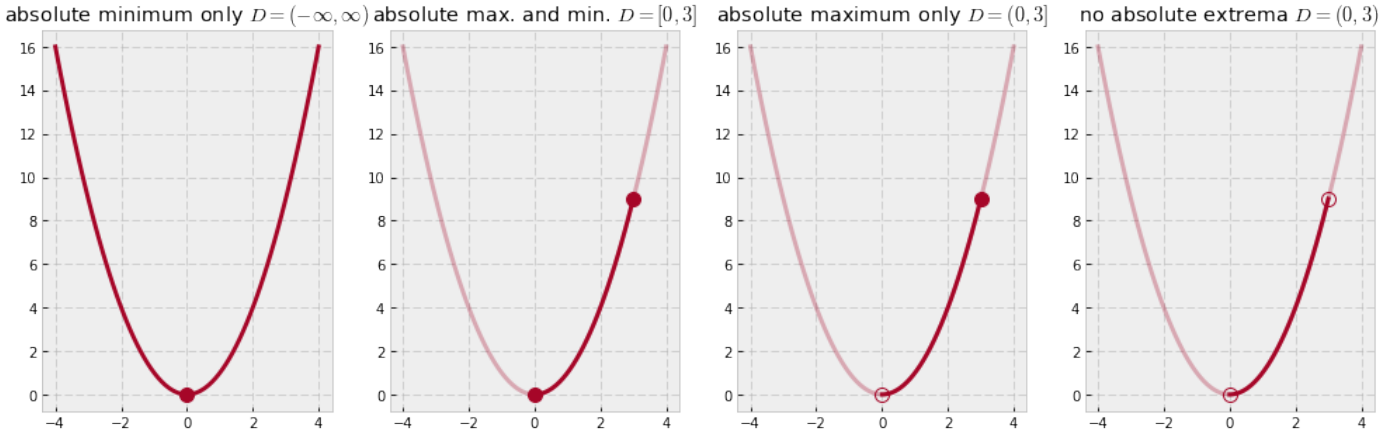
jupyter testfuncs.ipynb

실험용 목적함수


Test functions
Test function 1
Test function 2
Test function 3
x
y
z

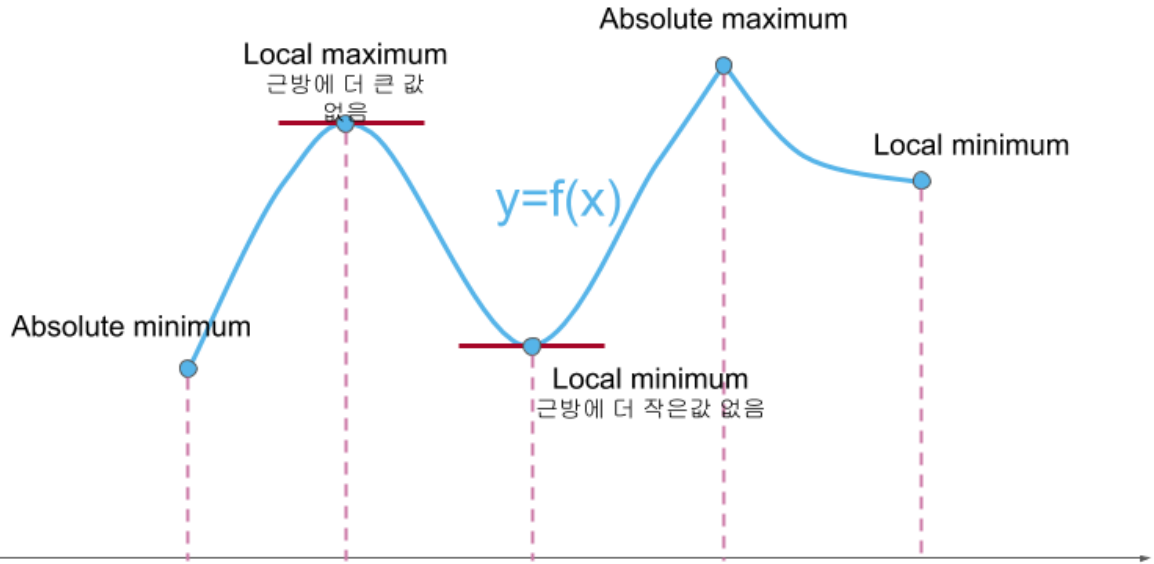
함수의 최대, 최소Global min, max.

함수	도메인 D	최대, 최소
$y = x^2$	$(-\infty, \infty)$	최대값 없음, $x = 0$에서 최솟값 0
$y = x^2$	$[0,3]$	$x = 3$에서 최댓값 9 , $x = 0$에서 최솟값 0
$y = x^2$	$(0,3]$	$x = 3$에서 최댓값 9, 최솟값 없음
$y = x^2$	$(0,3)$	최댓값, 최솟값 없음



함수의 극대, 극소Local min, max.

- 어떤 점 근방에서 제일 크거나 작은 값
- 국소 최소, 최대 또는 지역 최소, 최대
- 극댓값과 극솟값을 통틀어 극점local extrema이라 함.



간단한 목적함수의 최솟값

- 필요조건 : 최솟값이 되기 위해 반드시 만족 되어야 하는 조건
  - 한번 미분=0, $f'(x) = 0$, $\nabla f(\mathbf{x}) = \mathbf{0}$

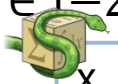
- 함수에서 y의 최솟값은 0, y 가 최솟값이 되도록 하는 x는 10

$$y = x^2 - 20x + 100 \rightarrow y = (x-10)^2, \qquad \frac{dy}{dx} = 2x - 20 \triangleq 0$$

- 울퉁불퉁(multimodal) 함수 → 미분해서 0인 지점을 다 구해서 비교

$$y = x^3 - 5x \quad x \in [-2,2]$$

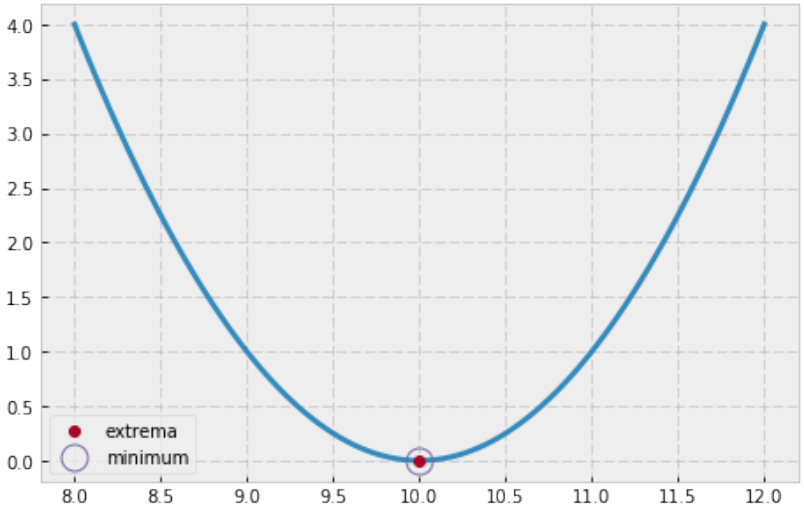
```
x = sympy.symbols('x')
sympy.solveset(sympy.diff(x**2 -
20*x + 100, x), x)
```



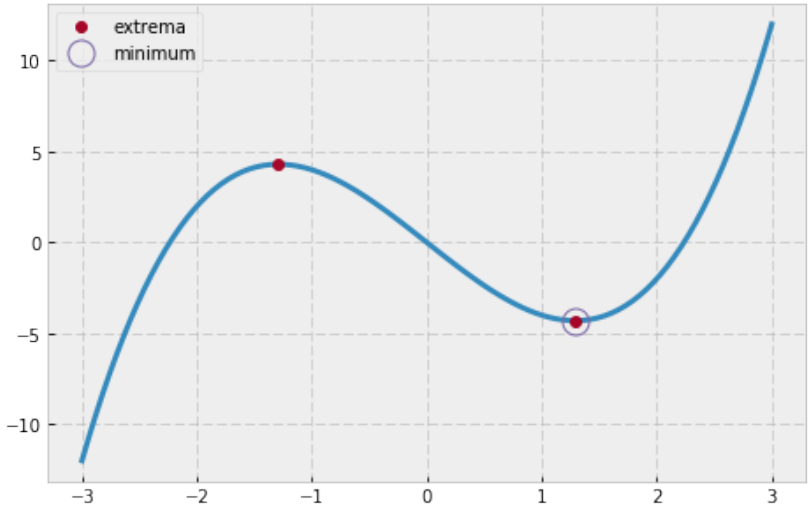
```
x = sympy.symbols('x')
sympy.solveset( sympy.diff(x**3 -
5*x, x),  x )
```

해석적 최솟값 찾기

- 극대, 극솟값을 통해 목적함수의 최솟값 찾기 가능



$y = x^2 - 20x + 100$



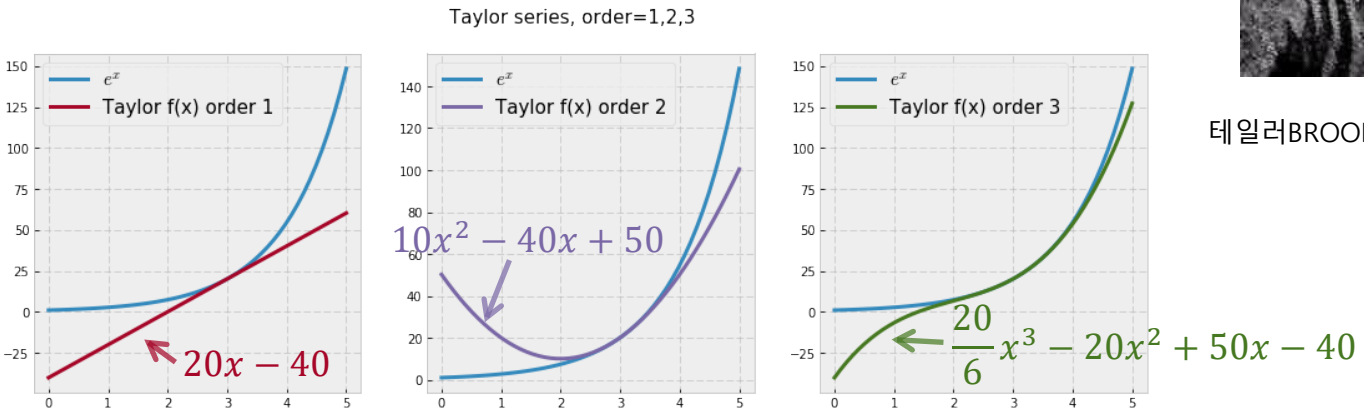
$y = x^3 - 5x \quad x \in [-2,2]$

테일러 급수Taylor series

- 미분 가능한 함수를 어느 점 근처에서 다항식으로 근사하는 방법
- 1715년 이후, 영국의 수학자 테일러Brook Taylor에 의해 널리 알려짐

$$f(x) = f(x^*) + \frac{df(x^*)}{dx}(x - x^*) + \frac{1}{2}\frac{d^2 f(x^*)}{dx^2}(x - x^*)^2 + R$$

테일러BROOK TAYLOR(1685~1731)



경사도 벡터Gradient

- n개의 변수 $x_1, x_2, \ldots, x_n$에 대한 함수 $f(\mathbf{x})$의 $\mathbf{x}^*$에서의 편미분 계수

$$c_i = \frac{\partial f(\mathbf{x}^*)}{\partial x_i}, \qquad i = 1, \ldots, n$$

- 이를 열 벡터로 모으면 경사도 벡터

$$\nabla f(\mathbf{x}^*) = \begin{bmatrix} \dfrac{\partial f(\mathbf{x}^*)}{\partial x_1} \\ \dfrac{\partial f(\mathbf{x}^*)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(\mathbf{x}^*)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f(\mathbf{x}^*)}{\partial x_1} & \dfrac{\partial f(\mathbf{x}^*)}{\partial x_2} & \cdots & \dfrac{\partial f(\mathbf{x}^*)}{\partial x_n} \end{bmatrix}^{\mathrm{T}}$$

경사도 벡터 확인

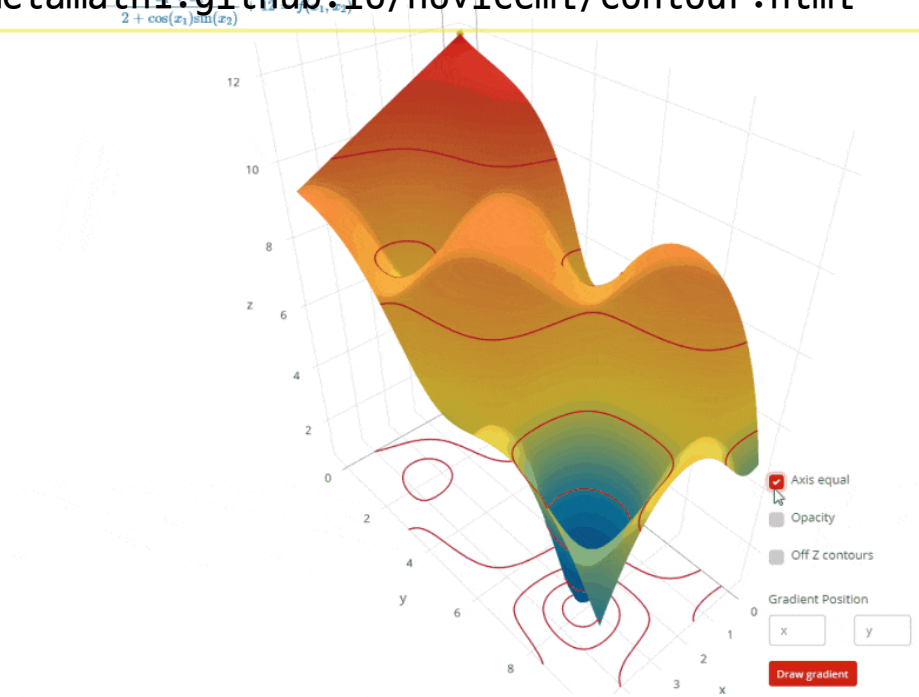
- 변수 2개에 대한 함수의 경사도 벡터를 직접 확인

JS MULTIVARIABLE-SCALAR FUCNTION

metamath1.github.io/noviceml/contour.html

- 경사도 벡터를 계산할 x, y 입력 후 '**Draw gradient**' 클릭

- $f(x,y)$에 해당하는 높이에 초록색 등고선이 그려지고 등고선 상에 Cone 모양으로 경사도 벡터가 그려 짐
  - 벡터 방향 : Cone 방향
  - 벡터 크기 : Cone 크기



헤시안행렬Hessian

- 다변수 함수를 두 번 편미분한 결과
- 경사도 벡터를 편미분한 야코비안

$$\mathbf{H} = \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}\, \partial \mathbf{x}} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 x_1} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n x_n} \end{bmatrix}$$

필요조건$\nabla f(\mathbf{x}) = \mathbf{0}$

- 테일러 급수에 의해 $f(x)$는 아래처럼 전개

$$f(x) = f(x^*) + \frac{df(x^*)}{dx}(x - x^*) + \frac{1}{2}\frac{d^2 f(x^*)}{dx^2}(x - x^*)^2 + R$$

$d = x - x^*$로 두면

$$f(x^* + d) = f(x^*) + \frac{df(x^*)}{dx}d + \frac{1}{2}\frac{d^2 f(x^*)}{dx^2}d^2 + R$$

다변수에서는 다음과 같다.

$$f(\mathbf{x}^* + \mathbf{d}) = f(\mathbf{x}^*) + \nabla f^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d} + R$$

$$\underbrace{f(\mathbf{x}^* + \mathbf{d}) - f(\mathbf{x}^*)}_{\Delta f} = \overbrace{\nabla f^{\mathrm{T}}\mathbf{d}}^{=0} + \frac{1}{2}\overbrace{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}^{>0} + R$$

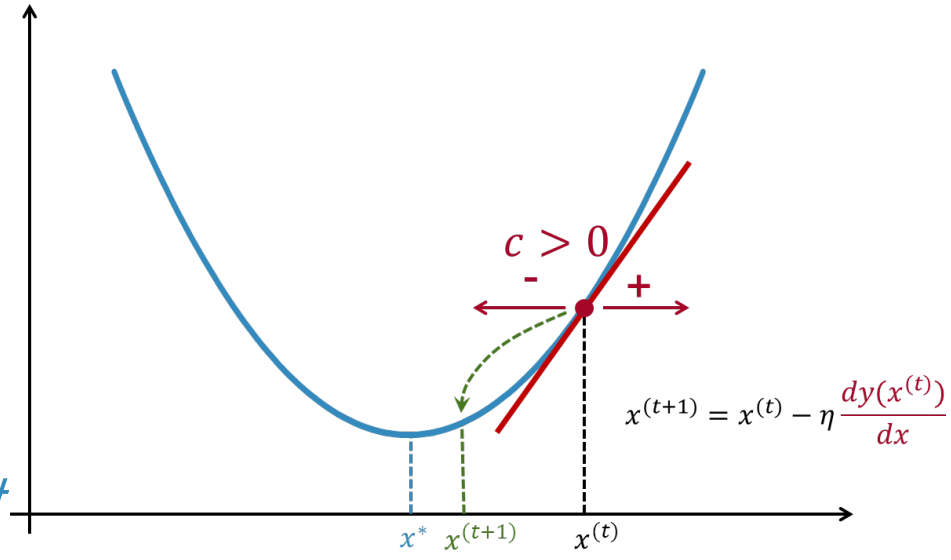
$\mathbf{x}^*$ 에서 의 작은 변화

어떤 방향으로의 작은 변화에도 상관없이

$\Delta f > 0$이 되어야 $\mathbf{x}^*$가 국부적 최소

식을 모른다면…

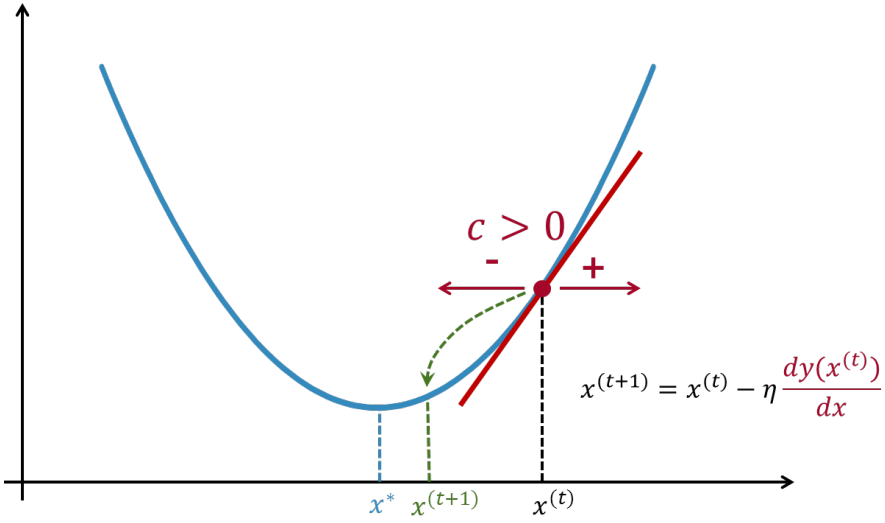
- 그림에서 현재 위치를 $x^{(t)}$라 하고 함수가 생긴 전반적인 모습은 모른다고 가정
- 양방향 다 아주 약간 움직여 보고 함수가 줄어드는 방향을 찾고 그 방향으로 어느 정도 이동
- 2변수 이상→ 이동할 수 있는 방향이 무한대
- *함수 값이 줄어드는 방향을 찾아주는 일반적인 방법이 필요하다.*

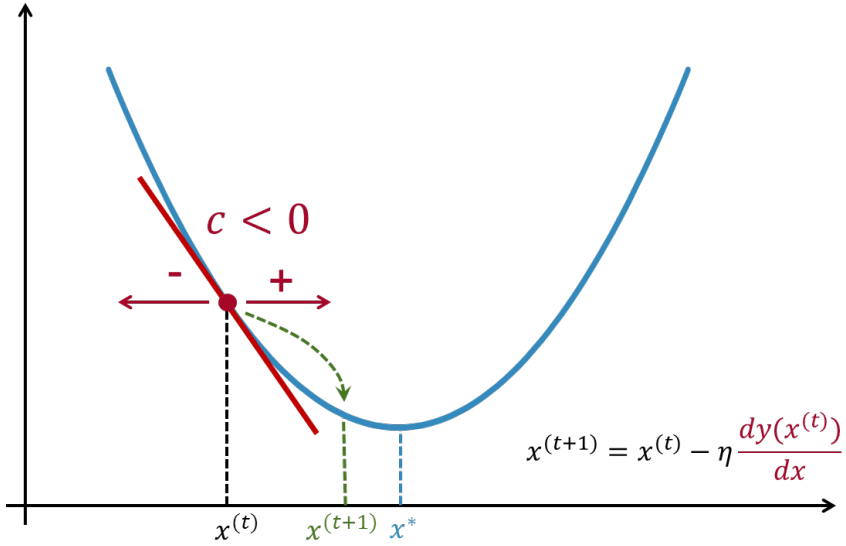


식을 모른다면…

- 경사하강

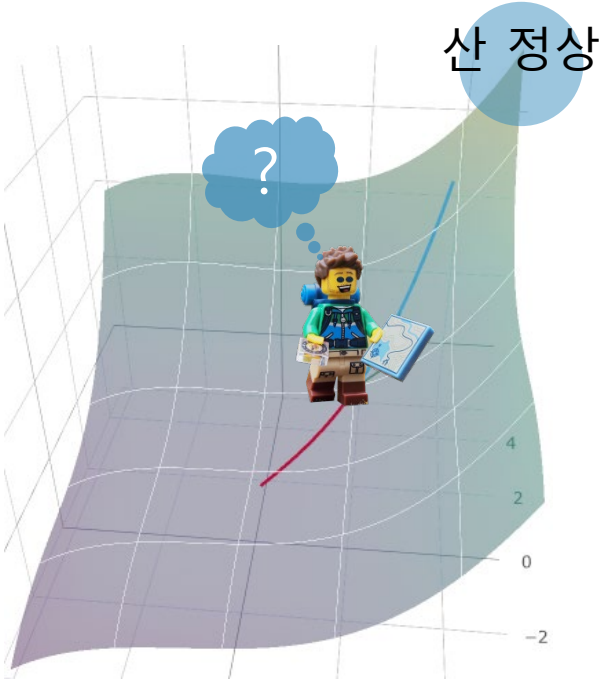
$$x^{(t+1)} = x^{(t)} - \eta \frac{dy(x^{(t)})}{dx}$$ 수치미분

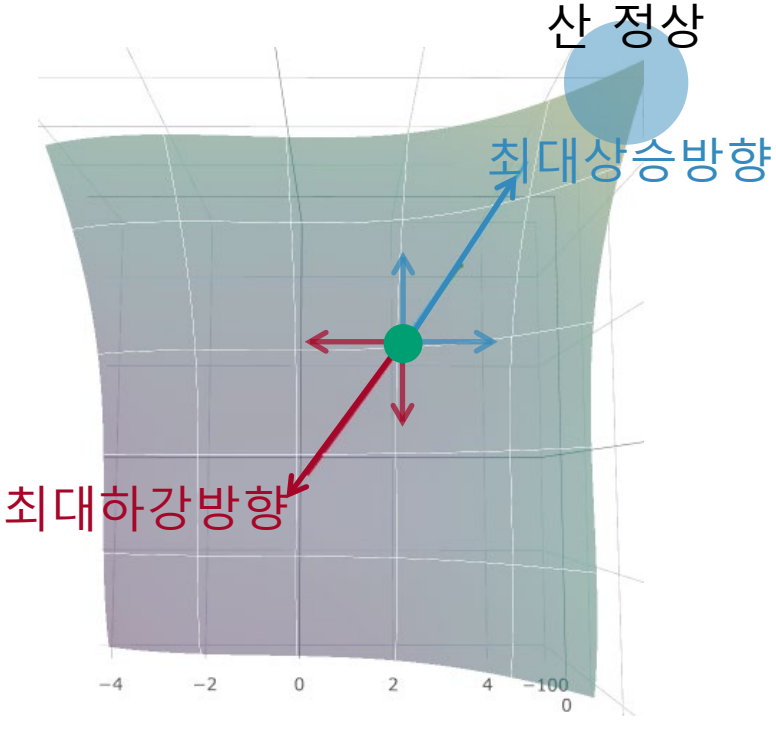




다변수 함수를 위한 눈 먼 등산객

- 눈 먼 등산객은 가장 빠른 시간에 산 정상에 도달하고 싶다.
- 어느 방향으로 가야 할 지 알 수 있을까?



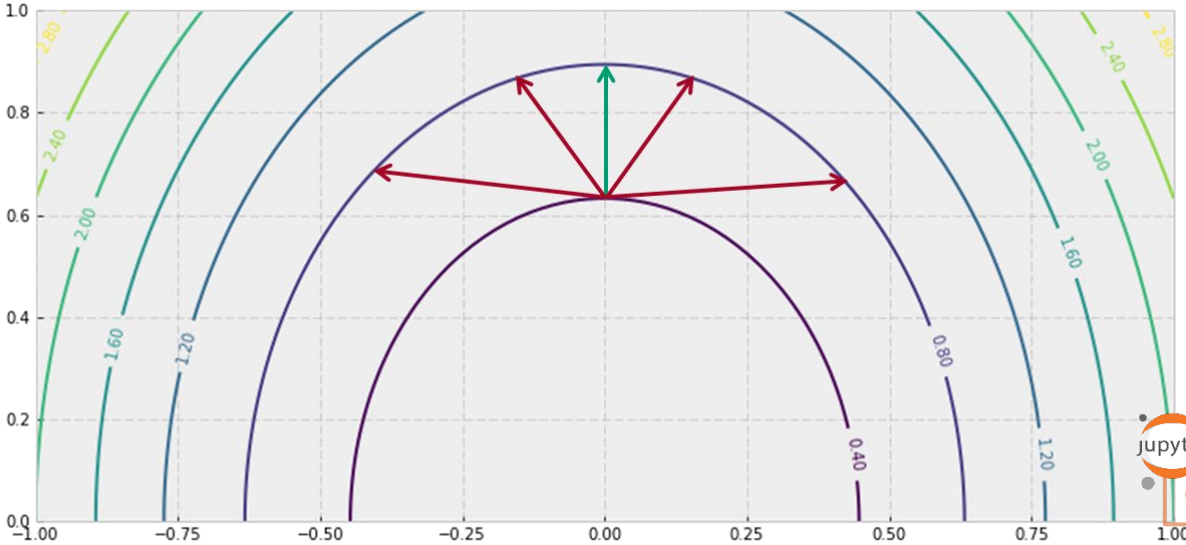


경사도 벡터의 성질

- $f(x) = 0.8$ 고지로 가기 위해 어디로 갈까?

경사도 벡터의 성질

- 함수표면의 접면에 수직
- 경사도벡터의 방향은 함수의 최대증가방향



jupyter optimum.ipynb

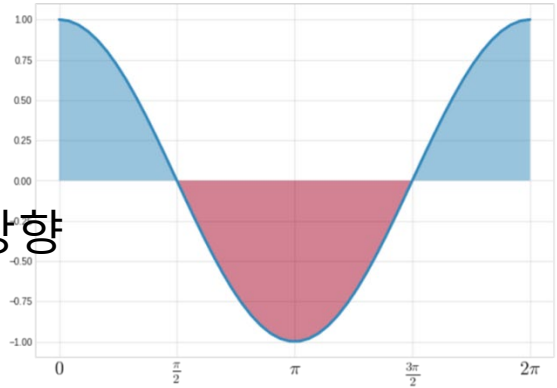
강하방향Descent Direction

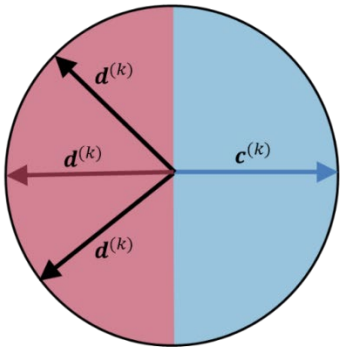
- 다음 강하조건을 만족하는 방향은 강하방향descent direction
- 강하방향이 기반한 방법을 강하법descent method이라 함

$$\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} = \left|\mathbf{c}^{(k)}\right|\left|\mathbf{d}^{(k)}\right| \cos\gamma < 0$$

경사도벡터

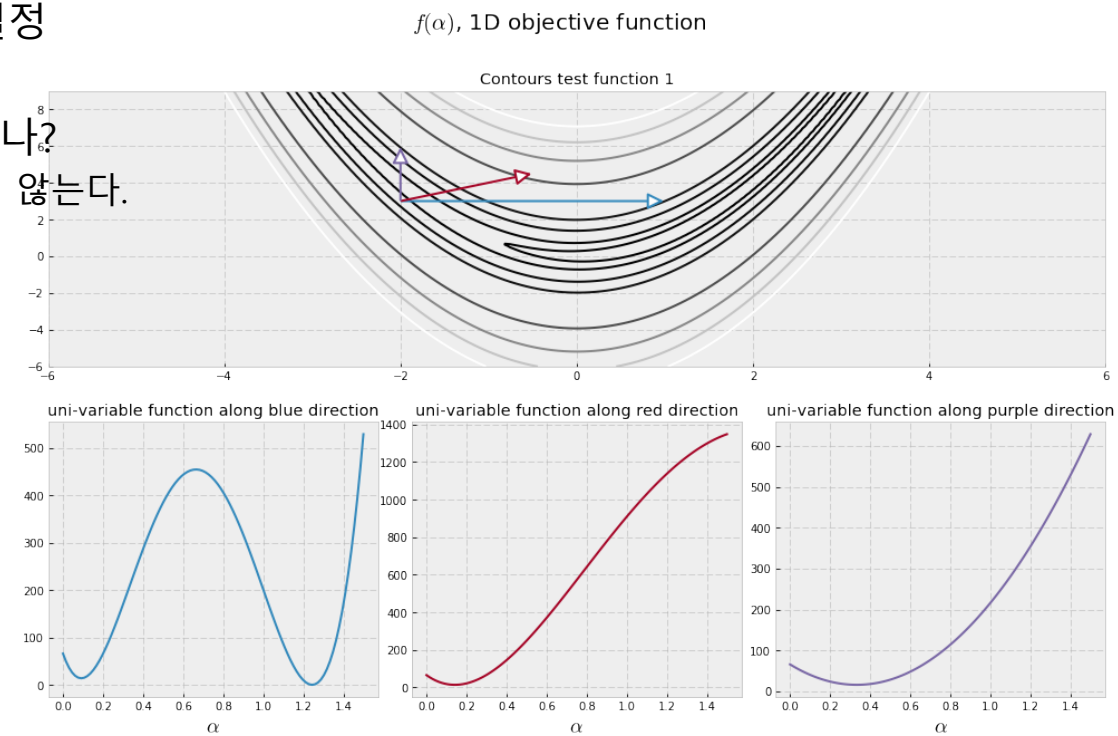
우리가 결정할 임의의 방향





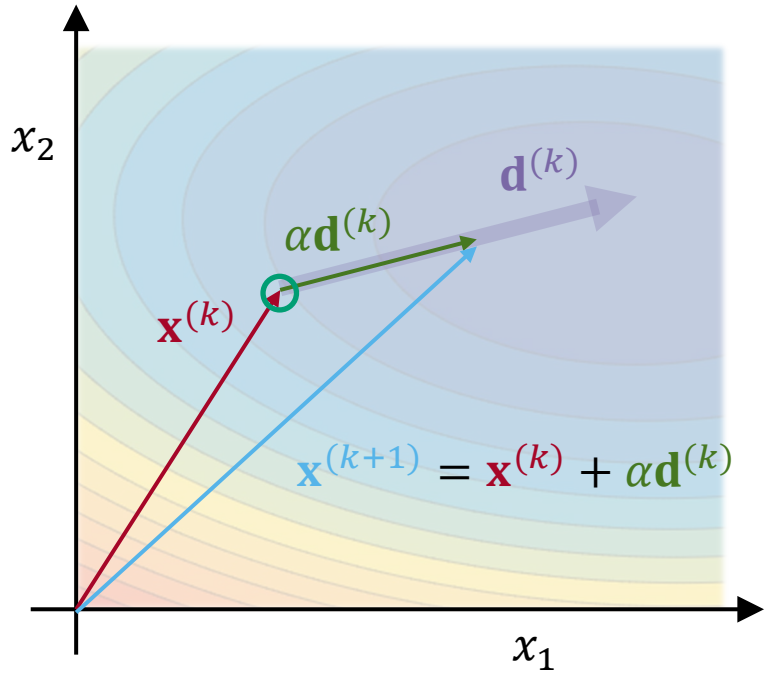
이동 거리Step Size

- 경사도 벡터 : 이동 방향을 결정
- 그 방향으로 얼마나 가야 하나?
  - 계속 함수의 값이 줄어 들지 않는다.
- 1차원 함수에 대한 경사하강
  - 고비용
- 보다 비용이 적게 드는
  - 황금분할탐색, 다항식 보간
- 더 비용이 적게 드는
  - 고정 이동 거리 : 학습률



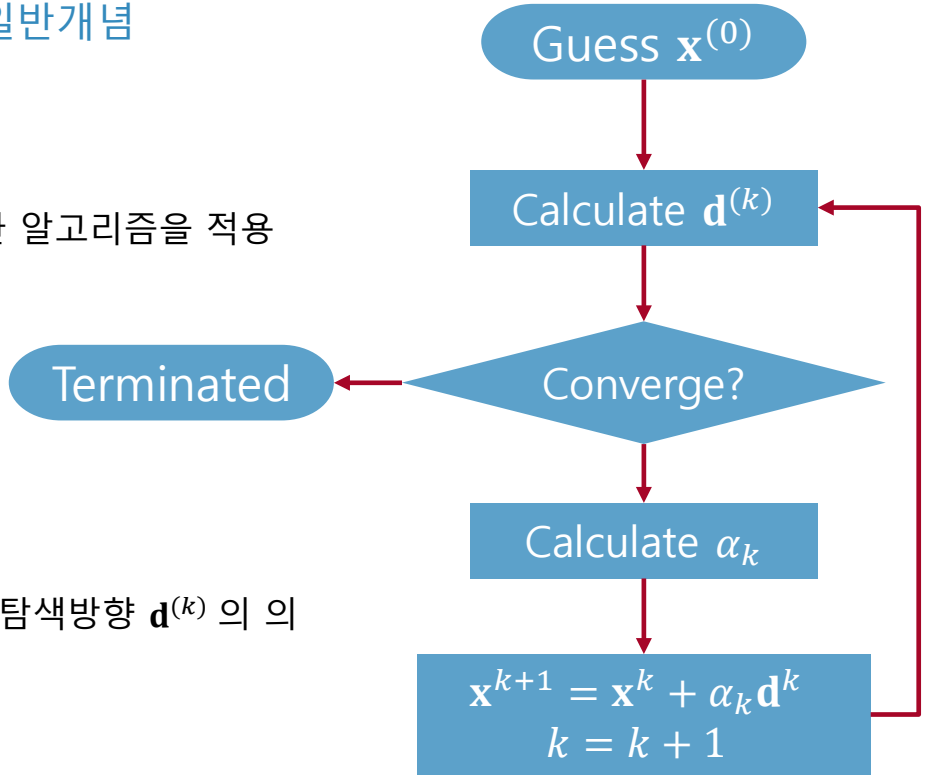
변수 업데이트

- Step 1. 시작점 $\mathbf{x}^{(k)}$에서 강하방향 $\mathbf{d}^{(k)}$를 계산
- Step 2. $\mathbf{d}^{(k)}$방향으로 스탭사이즈 α결정
- Step 3. $\mathbf{c}^{(k)}$ 두 벡터 $\mathbf{x}^{(k)}$와$\alpha\mathbf{d}^{(k)}$를 벡터 합
- Step 4. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + a_k\mathbf{d}^{(k)}$로 현재 위치를 수정



경사하강법 일반순서

- 경사도 벡터를 사용하는 수치 알고리즘 일반개념
  - 단계 1. 타당성 있는 출발점 $\mathbf{x}^{(0)}$ 추정,
  - 단계 2. 탐색방향 $\mathbf{d}^{(k)}$ 를 계산
  - 단계 2. 수렴 검토
  - 단계 4. 양의 이동거리 α_k계산(여러 다양한 알고리즘을 적용 가능)
  - 단계 5. 새로운 설계 변수 계산
    $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k,\ k = k + 1$
    단계 2로 가서 반복



- 알아봐야 하는 내용
  - 단계 2에서 경사도 벡터로 부터 결정되는 탐색방향 $\mathbf{d}^{(k)}$ 의 의미
  - 단계 4에서 양의 이동거리 α_k의 의미

최속강하법Steepest Descent

- Step 1. 시작점 $\mathbf{x}^{(0)}$을 정하고 반복 카운터를 $k = 0$으로 설정. 수렴조건 ϵ도 적당히 설정.
- Step 2. $\mathbf{x}^{(k)}$에서 $f(\mathbf{x})$의 경사도벡터를 다음처럼 계산. $\mathbf{c}^{(k)} = \nabla f(\mathbf{x}^{(k)})$
- Step 3. $\mathbf{c}^{(k)}$의 크기를 $\|\mathbf{c}^{(k)}\|$로 구하고 만약 $\|\mathbf{c}^{(k)}\| < \epsilon$이면 중단, 그렇지 않으면 계속 진행.
- Step 4. $\mathbf{x}^{(k)}$에서 탐색 방향을 $\mathbf{d}^{(\mathrm{k})} = -\mathbf{c}^{(k)}$로 설정.
- Step 5. $f(\alpha) = f\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{k}\right)$를 최소화하는 α_k를 계산.(다양한 알고리즘이 사용될 수 있다.)
- Step 6. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + a_k \mathbf{d}^{(k)}$로 현재 위치를 수정, $k = k + 1$로 두고 Step 2로 가서 반복.

공액경사법Conjugate Gradient

- Step 1. 시작점 $\mathbf{x}^{(0)}$을 정하고 반복 카운터를 $k=0$으로 설정. 수렴조건 ϵ도 적당히 설정. $\mathbf{x}^{(0)}$에서 강하 방향을 계산 $\mathbf{d}^{(0)}=-\mathbf{c}^{(0)}=-\nabla f(\mathbf{x}^{(0)})$. $\mathbf{c}^{(k)}$의 크기를 $\|\mathbf{c}^{(k)}\|$로 구하고 만약 $\|\mathbf{c}^{(k)}\|<\epsilon$이면 중단, 그렇지 않으면 Step 5로 이동(공액경사법에서 첫 반복은 최속강하법과 동일)

- Step 2. $\mathbf{x}^{(k)}$에서 $f(\mathbf{x})$의 경사도벡터를 다음처럼 계산. $\mathbf{c}^{(k)}=\nabla f(\mathbf{x}^{(k)})$

- Step 3. $\mathbf{c}^{(k)}$의 크기를 $\|\mathbf{c}^{(k)}\|$로 구하고 만약 $\|\mathbf{c}^{(k)}\|<\epsilon$이면 중단, 그렇지 않으면 계속 진행.

- Step 4. $\mathbf{x}^{(k)}$에서 공액 탐색 방향을 $\mathbf{d}^{(\mathrm{k})}=-\mathbf{c}^{(k)}+\beta_k\mathbf{d}^{(k-1)}$로 설정.

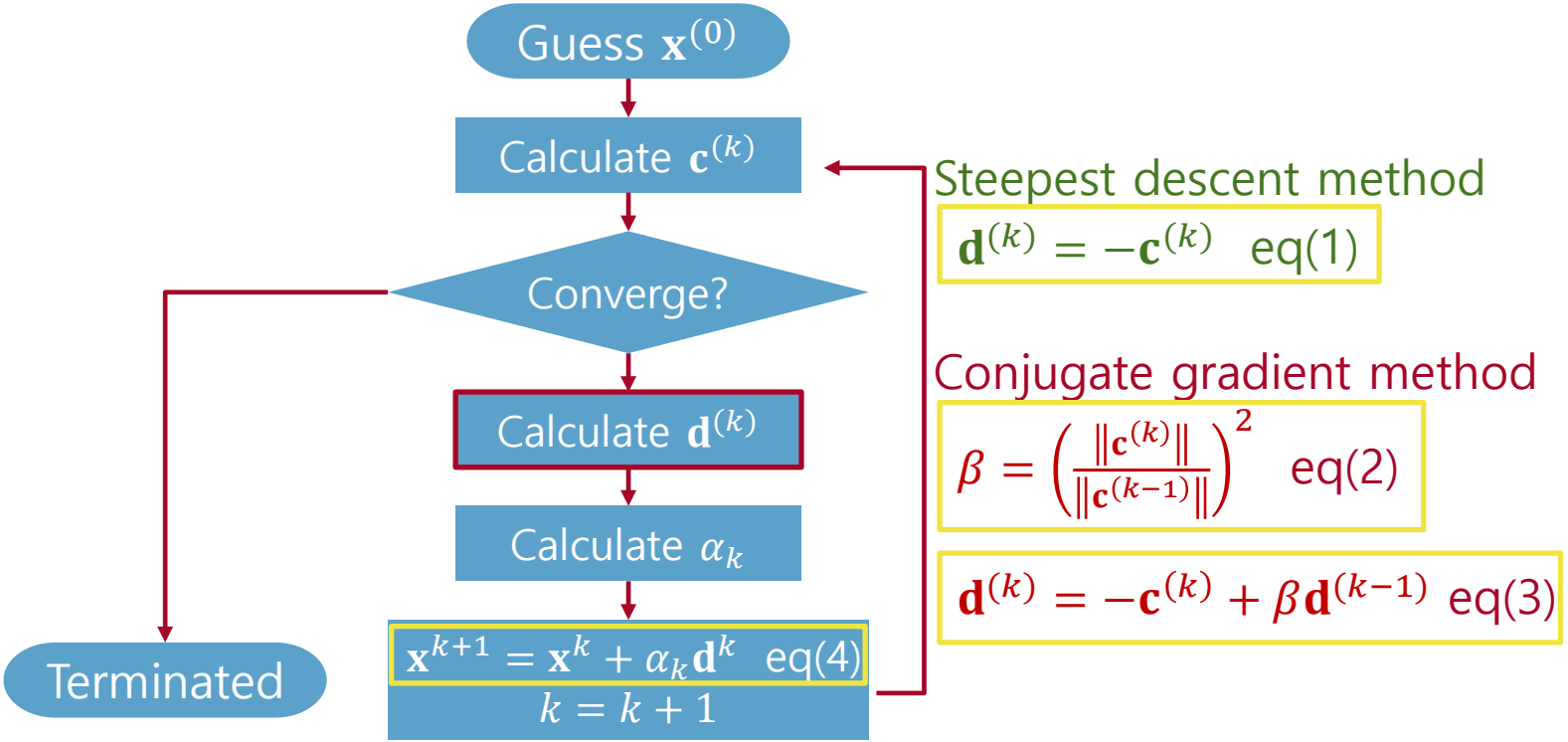
$$\beta_k=\left(\frac{\|\mathbf{c}^{(k)}\|}{\|\mathbf{c}^{(k-1)}\|}\right)^2=\frac{\left(\mathbf{c}^{(k)}\cdot\mathbf{c}^{(k)}\right)}{\left(\mathbf{c}^{(k-1)}\cdot\mathbf{c}^{(k-1)}\right)}$$

- Step 5. $f(\alpha)=f\left(\mathbf{x}^{(k)}+\alpha\mathbf{d}^{k}\right)$를 최소화하는 α_k를 계산.(다양한 알고리즘이 사용될 수 있다.)

- Step 6. $\mathbf{x}^{(k+1)}=\mathbf{x}^{(k)}+a_k\mathbf{d}^{(k)}$로 현재 위치를 수정, $k=k+1$로 두고 Step 2로 가서 반복.

강하방향 차이SDM vs CGM

- 최속강하법과 공액경사법의 강하방향 차이

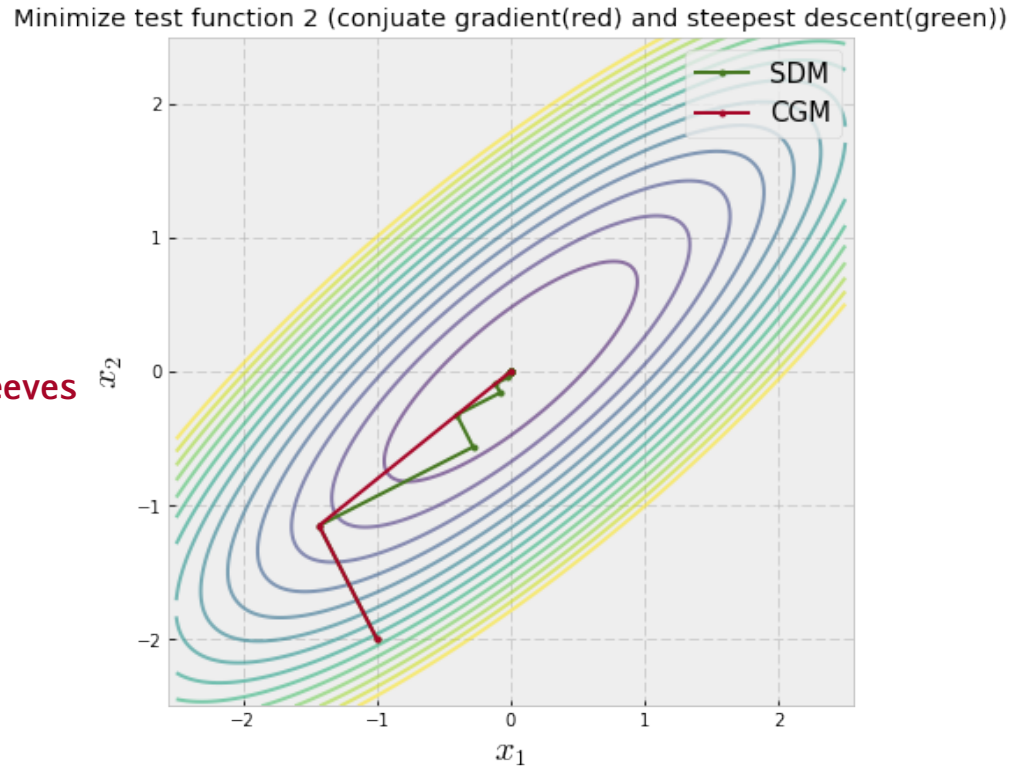


SDM vs CGM

```
INIT POINT : [-1 -2], dtype : int64
METHOD : Steepest Descent
Stop criterion break
iter: 26, x:[-3.5836e-07 -2.8791e-07],
cost:0.0000000
```

```
INIT POINT : [-1 -2], dtype : int64
METHOD : Conjugate gradient Fletcher-Reeves
Stop criterion break
iter: 3, x:[ 2.1295e-08 -4.1714e-08],
cost:0.0000000
```

jupyter optimum.ipynb



scipy.optimize.minimize

```
fun    : 3.0666884056615452e-12
jac    : array([8.8665e-06, 6.3681e-07])
message: 'Optim. terminated success.'
nfev   : 344
nit    : 36
njev   : 86
status : 0
success: True
x      : array([2., 4.])
```