탐색과 최적화 – 3

경사하강법과 제약조건 최적화

이건명

충북대학교 산업인공지능학과

인공지능 : 튜링 테스트에서 딥러닝까지

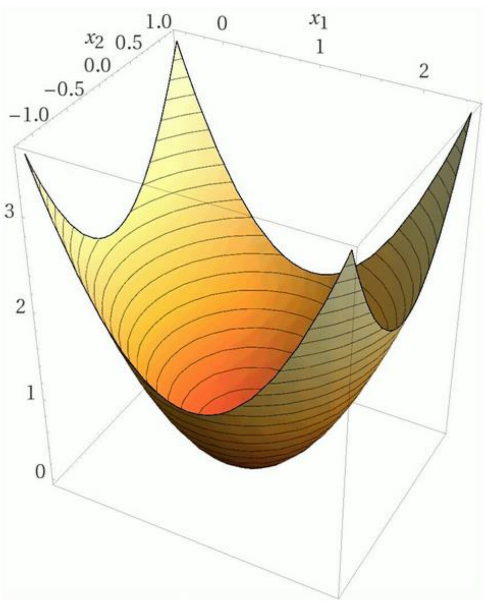
학습 내용

- **함수 최적화 문제**와 **경사하강법**에 대해서 알아본다.
- **제약조건 최적화 문제**와 **라그랑지 함수 기반 해법**에 대해서 알아본다.
- **이차계획법 문제**의 해를 cvxopt 패키지를 사용하여 구하는 방법을 알아본다.

1. 함수 최적화

❖ 함수 최적화(function optimization)

- 어떤 **목적 함수**(objective function)가 있을 때, 이 함수를 **최대**로 하거나 **최소**로 하는 **변수 값**를 찾는 최적화 문제



Find x_1, x_2

which minimizes $f(x_1, x_2) = (x_1 - 1)^2 + x_2^2$

목적함수 (objective function)

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 - 2 = 0 \qquad x_1 = 1$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = 2x_2 = 0 \qquad x_2 = 0$$

$$(x_1{}^*, x_2{}^*) = (1,0)$$

경사 하강법

❖ **경사 하강법**(gradient descent method)

- 복잡한 함수 $f(\boldsymbol{x})$ 의 최적해 탐색을 위해, **임의**의 **위치**에서 **시작**하여 함수 $f(\boldsymbol{x})$의 **그레디언트**(gradient) **반대 방향**으로 조금씩 움직여 가며 최적의 변수 값를 찾으려는 방법
  - **그레디언트**(gradient)
    - 함수 $f(\boldsymbol{x})$를 각 변수에 대해 편미분한 벡터

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

- 탐색 중의 각 위치에서 그레디언트를 계산하여, 그레디언트 반대방향으로 이동하도록 **변수의 값**을 **반복적**으로 **조금씩 조정**

$$x_1 \leftarrow x_1 - \eta \frac{\partial f}{\partial x_1}$$

$$x_2 \leftarrow x_2 - \eta \frac{\partial f}{\partial x_2}$$

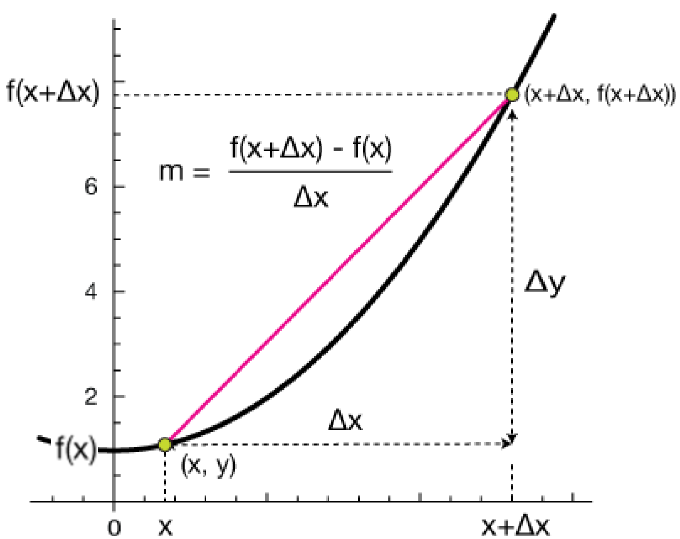
η : 학습율(learning rate)

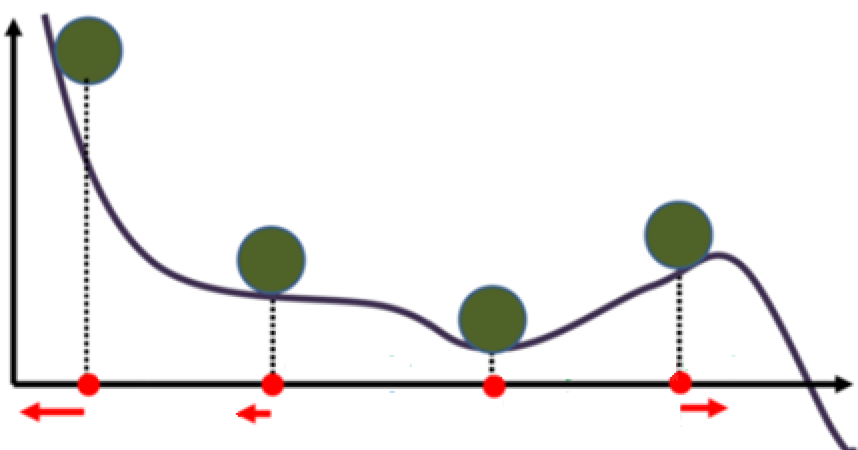
경사 하강법

❖ 경사 하강법 – cont.

- **미분**(differentiation, 微分)
  - 함수 $f(x)$의 변수 x에 대한 **순간변화율**
  - x의 아주 미세한 변화에 대한 $f(x)$의 변화량

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$





[참고] 미분

❖ **미분 (differentiation, 微分)**

- 함수 $f(x)$의 변수 x에 대한 순간변화율
- x의 아주 미세한 변화에 대한 $f(x)$의 변화량

$$\boxed{f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}}$$

- 예) $f(x) = 2x^2 + 1$

$$f'(x) = \lim_{\Delta x \to 0} \frac{2(x+\Delta x)^2 + 1 - 2x^2 - 1}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{2x^2 + 4x\Delta x + (\Delta x)^2 + 1 - 2x^2 - 1}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{4x\Delta x + (\Delta x)^2}{\Delta x}$$

$$= \lim_{\Delta x \to 0} (4x + \Delta x)$$

$$= 4x$$

[참고] 미분

- **상수 미분** $\frac{d}{dx}c = 0$
- **거듭제곱 미분** $\frac{d}{dx}x^n = n\,x^{n-1}$
- **지수함수의 미분** $\frac{de^{ax}}{dx} = ae^{ax}$
- **로그함수의 미분** $\frac{d\log x}{dx} = \frac{1}{x}$

- **상수배의 미분** $\frac{d}{dx}[cf(x)] = c\frac{d}{dx}f(x)$
- **합의 미분** $\frac{d}{dx}[f(x)+g(x)] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$
- **곱의 미분** $\frac{d}{dx}[f(x)\,g(x)] = g(x)\frac{d}{dx}f(x) + f(x)\frac{d}{dx}g(x)$
- **분수식의 미분** $\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{g(x)\,f'(x) - f(x)\,g'(x)}{(g(x))^2}$

[참고] 미분

- **연쇄 법칙 (Chain Rule)**

$$y = f\big(g(x)\big)$$

$$y = f(u), \qquad u = g(x)$$

$$\boxed{\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}}$$

$$x \xrightarrow{g} u \xrightarrow{f} y$$

$$\frac{\Delta y}{\Delta x} = \frac{\Delta y}{\Delta u}\frac{\Delta u}{\Delta x}$$

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \left(\frac{\Delta y}{\Delta u}\frac{\Delta u}{\Delta x}\right) = \left[\lim_{\Delta x \to 0} \frac{\Delta y}{\Delta u}\right]\left[\lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x}\right]$$

$\Delta x \to 0$**이면,** $\Delta u \to 0$ **이므로**

$$= \left[\lim_{\Delta u \to 0} \frac{\Delta y}{\Delta u}\right]\left[\lim_{\Delta x \to 0} \frac{\Delta u}{\Delta x}\right]$$

$$= \frac{dy}{du}\frac{du}{dx}$$

[참고] 미분

❖ **편미분** (partial differentiation)

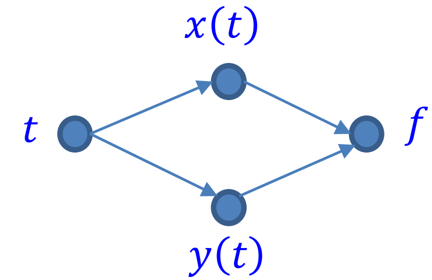
- **다변수 함수**에 대하여, 그 중 **하나의 변수**에 주목하고 나머지 변수의 값을 고정시켜 놓고 그 변수에 대해 하는 **미분**

- **예.** $f(x, y) = x^2 + xy + y^2$

$$\frac{\partial f(x, y)}{\partial x} = 2x + y$$

$$\frac{\partial f(x, y)}{\partial y} = x + 2y$$

[참고] 미분



❖ 다변수 함수의 연쇄 법칙

- $f\big(x(t), y(t)\big)$

$$\frac{df(x(t),y(t))}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

- 예. $f\big(x(t), y(t)\big) = x(t) + 2y(t)$
  $x(t) = 2t + 4, \quad y(t) = t^2$

$$\frac{\partial f}{\partial x} = 1\,,\ \frac{\partial f}{\partial y} = 2$$
$$\frac{dx}{dt} = 2\,, \frac{dy}{dt} = 2t$$
$$\frac{df(x(t),y(t))}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$
$$= 2 + 4t$$

- $g(x(t), y(t), z(t))$

$$\frac{dg(x(t),y(t),z(t))}{dt} = \frac{\partial g}{\partial x}\frac{dx}{dt} + \frac{\partial g}{\partial y}\frac{dy}{dt} + \frac{\partial g}{\partial z}\frac{dz}{dt}$$

- $h\big(x_1(t_1, t_2, \dots, t_m), x_2(t_1, t_2, \dots, t_m), \dots, x_n(t_1, t_2, \dots, t_m)\big)$

$$\frac{\partial h}{\partial t_i} = \frac{\partial h}{\partial x_1}\frac{\partial x_1}{\partial t_i} + \frac{\partial h}{\partial x_2}\frac{\partial x_2}{\partial t_i} + \cdots + \frac{\partial h}{\partial x_n}\frac{\partial x_n}{\partial t_i}$$

미분

❖ 그레디언트 (gradient)

- 함수 $f(x, y, z)$의 각 변수 x, y, z에 대한 편미분을 성분으로 갖는 벡터

$$\nabla f(x, y, z) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

- 함수 $f(x, y, z)$의 값이 가장 커지는 방향과 크기를 나타내는 벡터

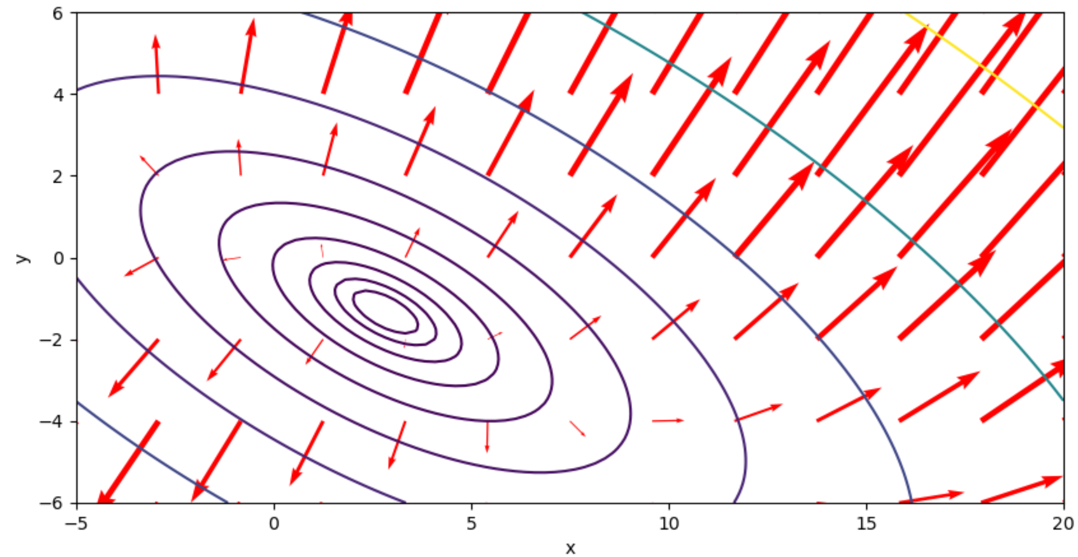
[실습] 그레디언트 그리기

```
import numpy as np
import matplotlib.pyplot as plt

def f(x,y):
    return 2*x**2 +4*x*y + 5*y**2 - 6*x + 2*y + 10

def dx(x,y):
    return 4*x + 4*y - 6

def dy(x,y):
    return 4*x + 10*y + 2

xi = np.linspace(-5, 20, 100)
yi = np.linspace(-6, 6, 100)
X, Y = np.meshgrid(xi, yi)
Z = f(X,Y)

xj = np.linspace(-5, 20, 13)
yj = np.linspace(-6, 6, 7)
X1, Y1 = np.meshgrid(xj, yj)
Dx = dx(X1, Y1)
Dy = dy(X1, Y1)

plt.figure(figsize=(10,5))
plt.contour(X, Y, Z, levels=np.logspace(0,3,10))
plt.quiver(X1, Y1, Dx, Dy, color='red', scale=500, minshaft=4)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

$$f(x,y) = 2x^2 + 4xy + 5y^2 - 6x + 2y + 10$$

$$\frac{\partial f(x,y)}{\partial x} = 4x + 4y - 6$$

$$\frac{\partial f(x,y)}{\partial x} = 4x + 10y + 2$$

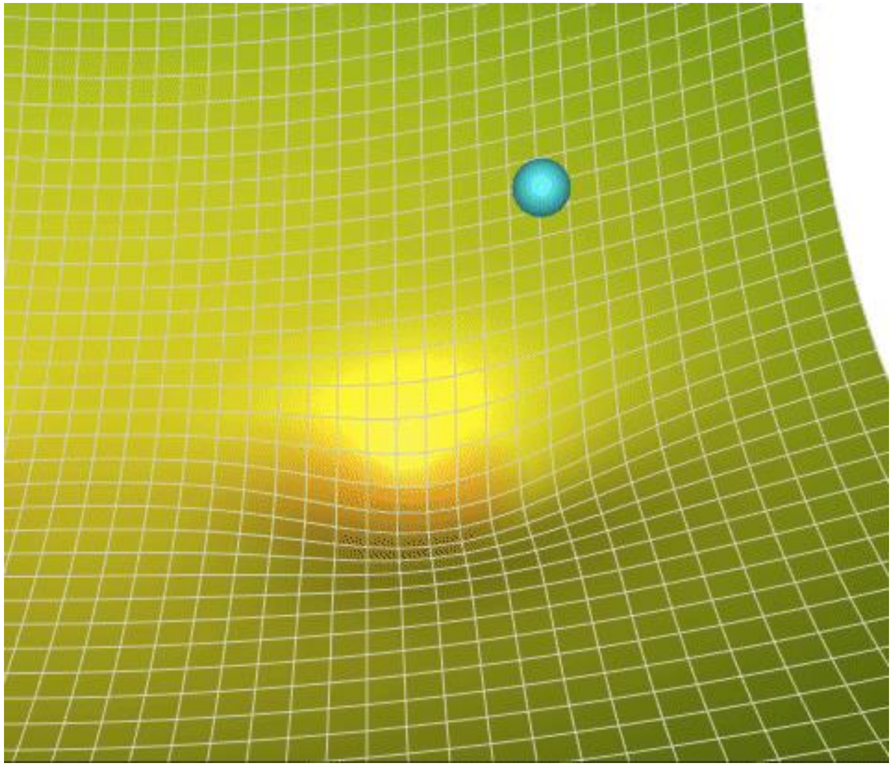


경사 하강법

❖ **경사 하강법 – cont.**

- **함수 값**이 **최소**가 되도록 **그레디언트** $-\nabla f$ 의 반대방향으로 **변수 값**을 점진적으로 **변경**하는 일

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

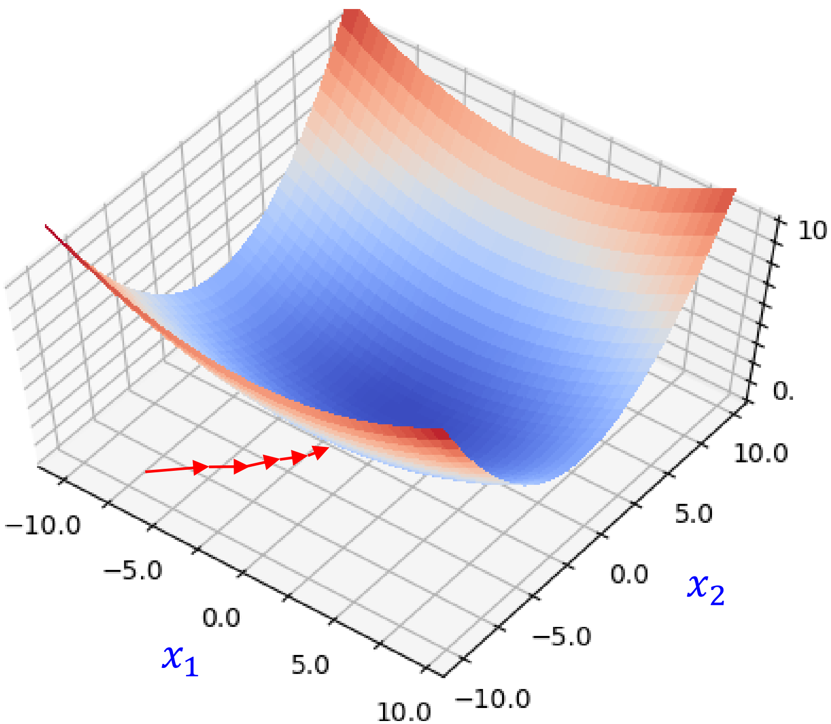


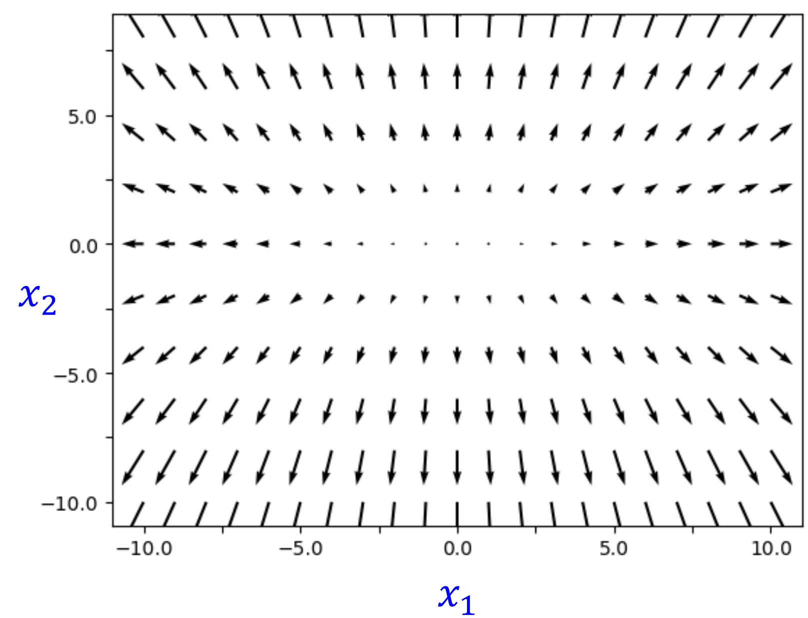
https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

경사 하강법

❖ 경사 하강법 – cont.

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

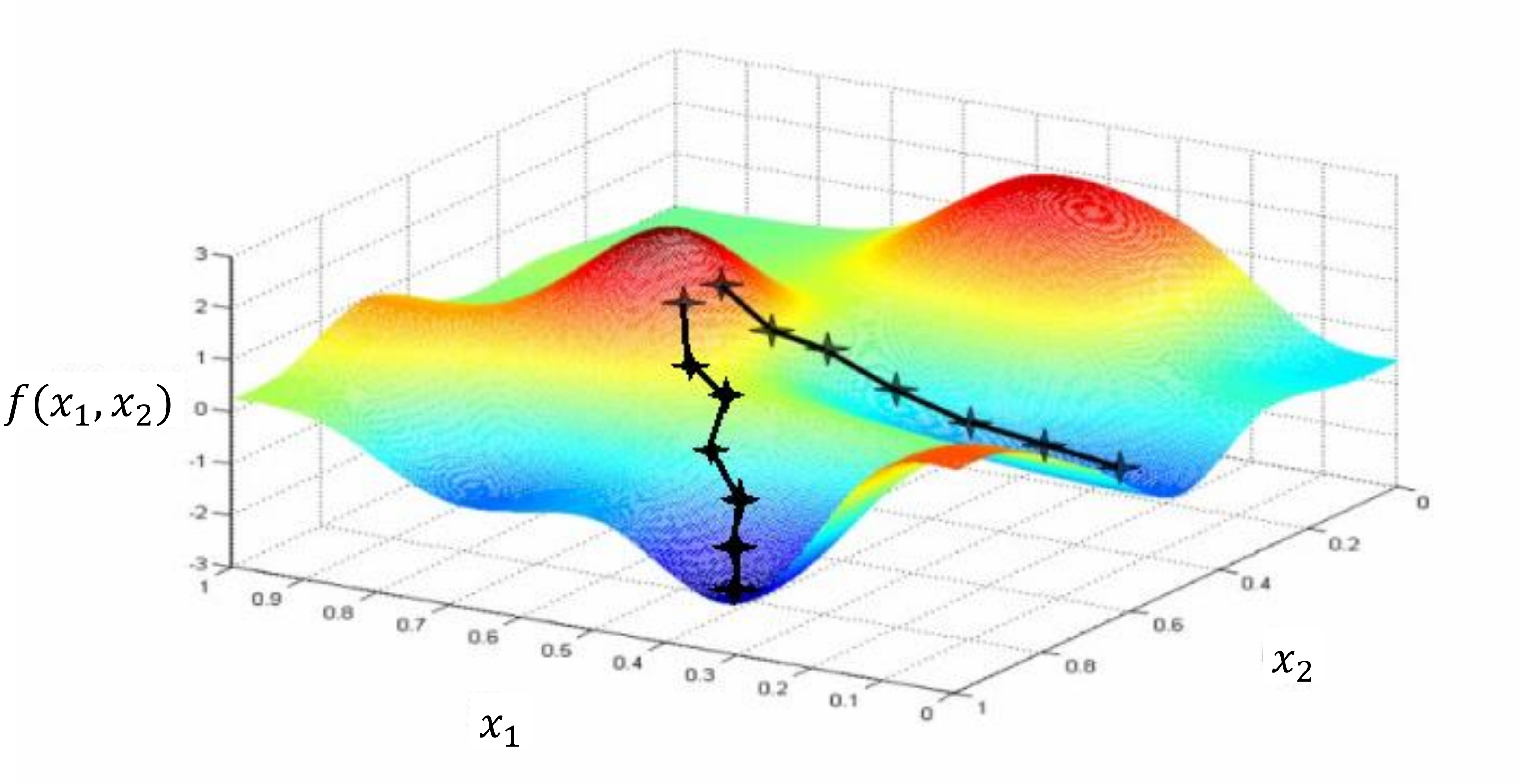




$$x^{(t+1)} \leftarrow x^{(t)} - \eta \nabla_x f$$

경사 하강법

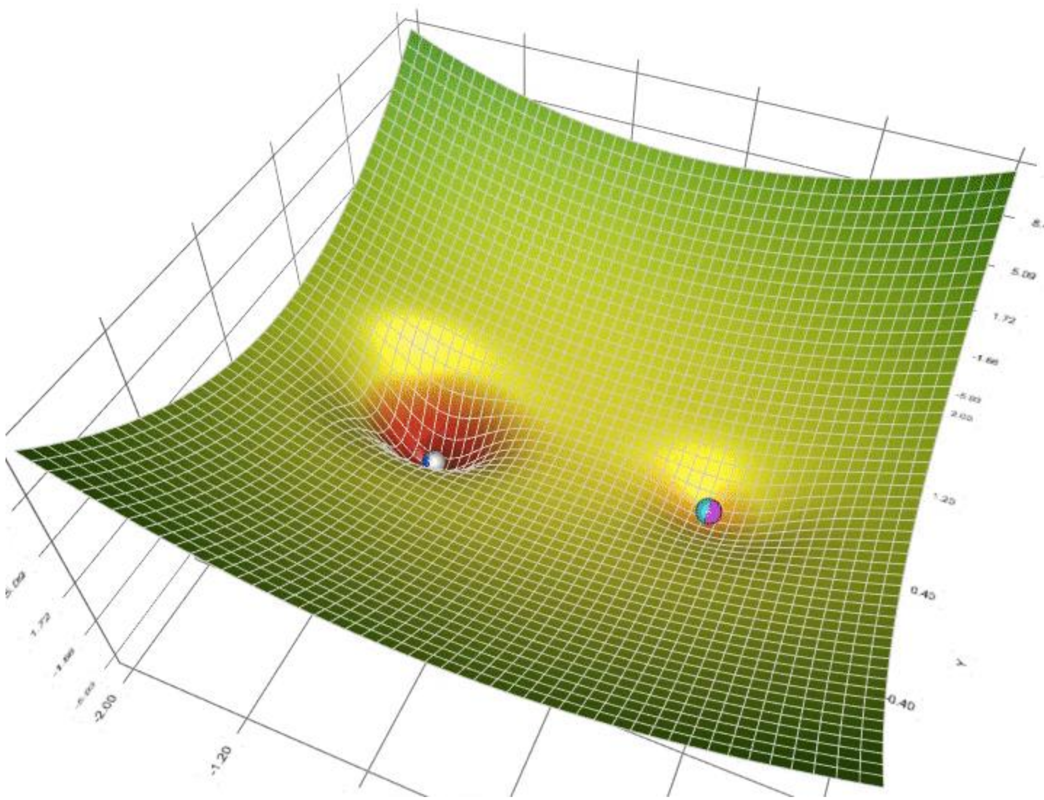
- **경사 하강법 기반의 함수 최적화**



http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/

경사 하강법

- **경사 하강법**
  - 경사하강법의 변형에 따른 탐색 과정



https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

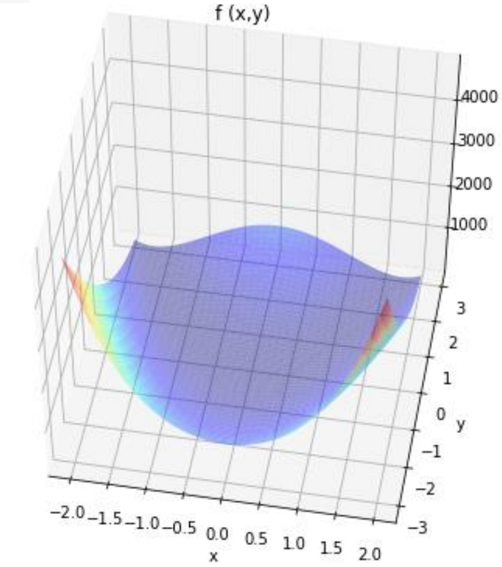
프로그래밍 실습: 경사 하강법

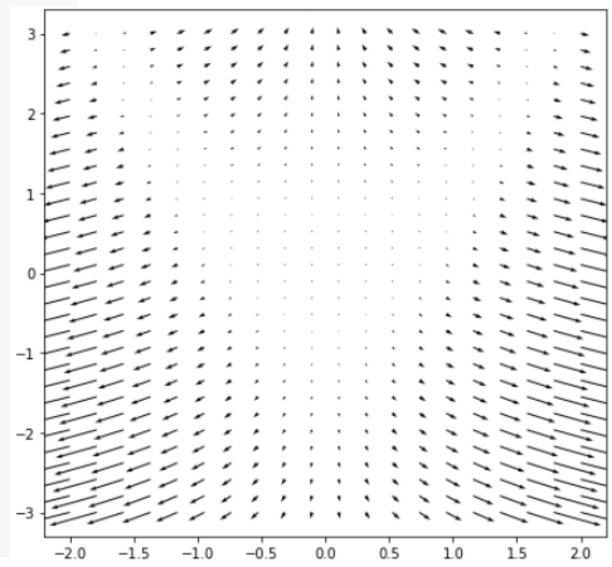
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

def f(x,y):
    return (1 + x)**2 + 100*(y - x**2)**2

def gradient(x,y):
    dx = -400*x*y + 400*x**3 + 2*x -2
    dy = 200*y -200*x**2
    return np.array([dx,dy])

def GradientDescent(Grad, x, y, eta=0.0012, epsilon=0.001, nMax=1000):
    i = 0
    pos_x, pos_y, pos_count = np.empty(0), np.empty(0), np.empty(0)
    error = 1
    sol = np.array([x,y])

    while np.linalg.norm(error) > epsilon and i < nMax:
        i +=1
        pos_x = np.append(pos_x, x)
        pos_y = np.append(pos_y, y)
        pos_count = np.append(pos_count, i)
        sol_prev = sol
        sol = sol - eta * Grad(x,y)
        error = sol - sol_prev
        x, y = sol[0], sol[1]

    print('최저점의 위치: ', sol)
    return sol, pos_x, pos_y, pos_count
```

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

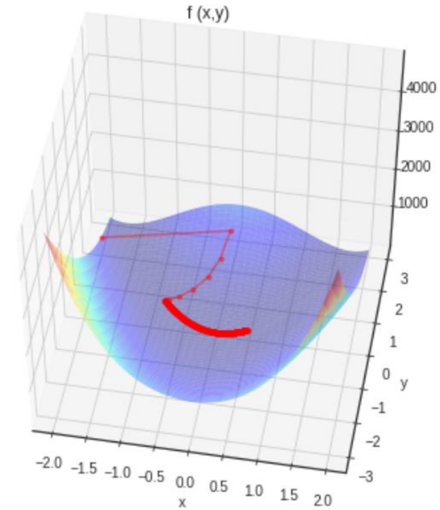
$$x \leftarrow x - \eta \frac{\partial f}{\partial x}$$

$$y \leftarrow y - \eta \frac{\partial f}{\partial y}$$

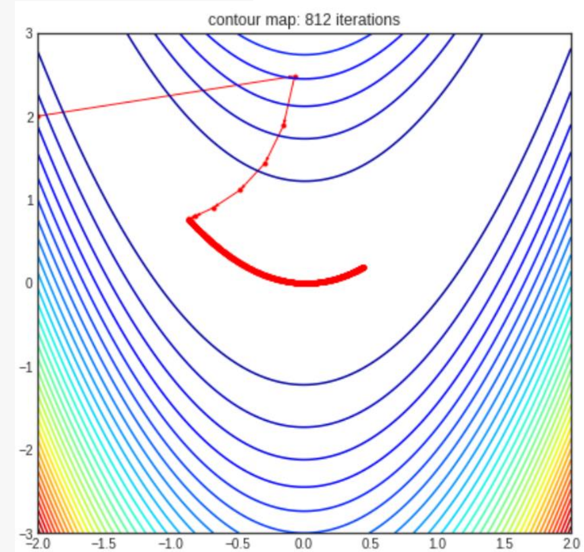




최저점의 위치: [0.44455055 0.19485836]

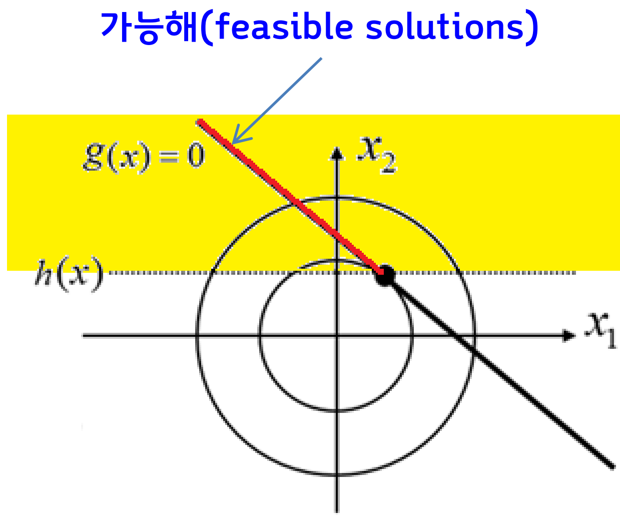



```

solution, pos_x, pos_y, pos_count = GradientDescent(gradient, -2, 2)

x = np.linspace(-2, 2, 200)
y = np.linspace(-3, 3, 300)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
anglesx = pos_x[1:] - pos_x[:-1]    # quiver plot에 사용할 벡터의 x 방향
anglesy = pos_y[1:] - pos_y[:-1]    # quiver plot에 사용할 벡터의 y 방향

%matplotlib inline
fig = plt.figure(figsize = (16,7))
ax = fig.add_subplot(1, 2, 1, projection='3d')
ax.plot_surface(X, Y, Z, rstride=5, cstride=5, cmap='jet', alpha=.4)
ax.plot(pos_x,pos_y, f(pos_x,pos_y), color='r', marker='.', alpha=.4)
ax.view_init(50, 280)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('f (x,y)')

ax = fig.add_subplot(1, 2, 2)
ax.contour(X, Y, Z, 40, cmap = 'jet')  # 등고선 지도
ax.scatter(pos_x, pos_y, color = 'r', marker = '.')
ax.quiver(pos_x[:-1], pos_y[:-1], anglesx, anglesy,scale_units='xy',angles='xy',scale=1,color='r')
ax.set_title('contour map: {} iterations'.format(len(pos_count)))

plt.show()
```



2. 제약조건있는 최적화

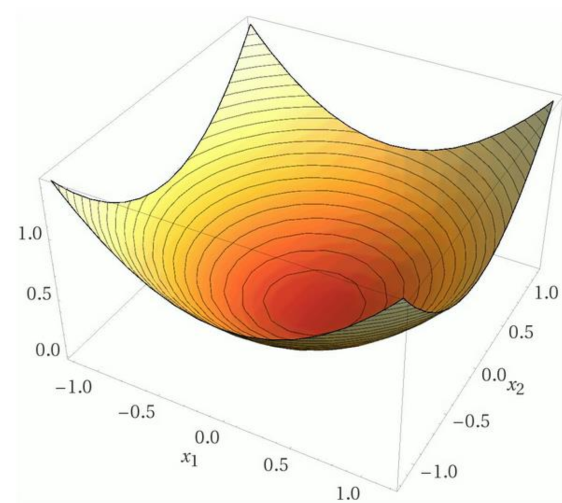
- **제약조건 최적화**(constrained optimization)
  - **제약조건**(constraints)을 만족하면서 **목적함수**를 **최적화**하는 변수들의 값을 찾는 문제



$$\text{Find } x_1, x_2$$

$$\text{which minimizes } f(x_1,x_2) = \frac{1}{2}(x_1^2 + x_2^2) \quad \text{목적함수}$$

$$\left.\begin{aligned} \text{subject to } g(x_1,x_2) &= 1 - x_1 - x_2 = 0 \\ h(x_1,x_2) &= \frac{3}{4} - x_2 \le 0 \end{aligned}\right\} \text{제약조건}$$



  - SVM의 학습에서 사용

제약조건있는 최적화

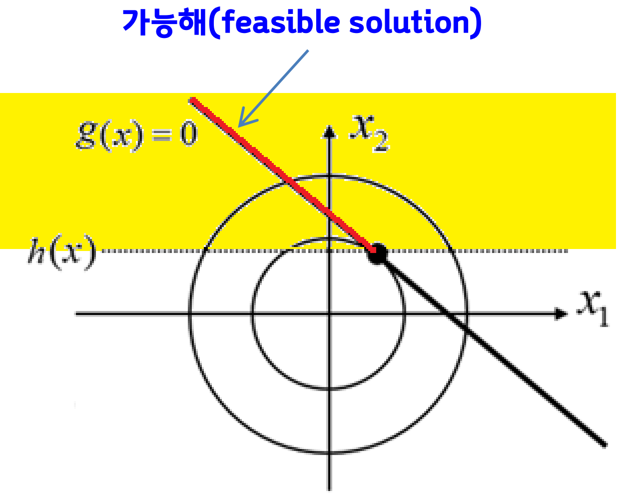
❖ 제약조건 최적화(constrained optimization)

Find x_1, x_2

which minimizes $f(x_1, x_2) = \frac{1}{2}(x_1^2 + x_2^2)$

subject to $g(x_1, x_2) = 1 - x_1 - x_2 = 0$

$$h(x_1, x_2) = \frac{3}{4} - x_2 \le 0$$



- **라그랑주(Lagrange) 함수** : 제약조건들과 목적함수를 결합한 함수

$$L(x_1, x_2, \lambda, \alpha) = f(x_1, x_2) + \lambda g(x_1, x_2) + \alpha h(x_1, x_2) \qquad (\alpha \ge 0)$$

$$= \frac{1}{2}(x_1^2 + x_2^2) + \lambda(1 - x_1 - x_2) + \alpha\left(\frac{3}{4} - x_2\right)$$

λ, α : 라그랑주 승수(Lagrange multiplier)

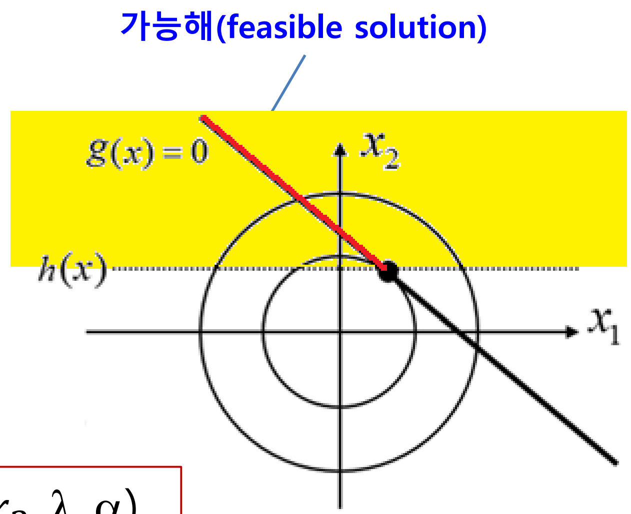
제약조건있는 최적화

❖ **제약조건 최적화(constrained optimization)**

$$L(x_1, x_2, \lambda, \alpha) = f(x_1, x_2) + \lambda g(x_1, x_2) + \alpha h(x_1, x_2) \quad (\alpha \geq 0)$$

$$= \frac{1}{2}(x_1^2 + x_2^2) + \underline{\lambda(1 - x_1 - x_2)} + \alpha\left(\frac{3}{4} - x_2\right) \quad (\alpha \geq 0)$$

λ, α : 라그랑주 승수(Lagrange multiplier)



- **최적화 방법**

$$\boxed{\min_{x_1, x_2 \in FS} f(x_1, x_2) = \min_{x_1, x_2} \max_{\alpha \geq 0, \lambda} L(x_1, x_2, \lambda, \alpha)}$$

FS : 가능해(feasible solution)의 집합

λ, α를 마음대로 바꾸며 $L(x_1, x_2, \lambda, \alpha)$를 아무리 키워도,
$\min_{x_1, x_2} \max_{\lambda, \alpha} L(x_1, x_2, \lambda, \alpha)$의 값은 x_1, x_2가 가능해일 때 나옴

등식 제약조건에 대한 $\lambda g(x_1, x_2)$에서
$g(x_1, x_2) > 0$ 이면, 양수인 λ를 선택해서 $\lambda g(x_1, x_2)$를 매우 큰 값으로 만들 수 있고,
$g(x_1, x_2) < 0$ 이면, 음수인 λ를 선택해서 $\lambda g(x_1, x_2)$를 매우 큰 값으로 만들 수 있음.

➔ $\min_{x_1, x_2} \max_{\lambda} \lambda g(x_1, x_2)$의 값은 $g(x_1, x_2) = 0$인,
즉 등식 제약조건이 만족하는 것 중에서 선택하게 됨.

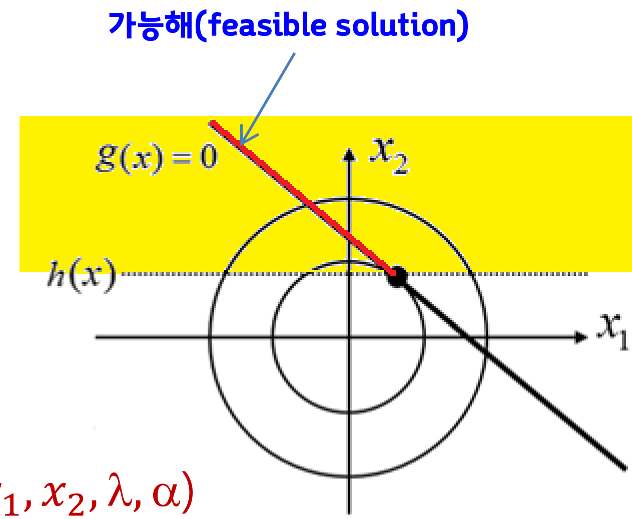
성질(I)

제약조건있는 최적화

❖ 제약조건 최적화(constrained optimization)

$$L(x_1, x_2, \lambda, \alpha) = f(x_1, x_2) + \lambda g(x_1, x_2) + \alpha h(x_1, x_2)$$
$$= \frac{1}{2}(x_1^2 + x_2^2) + \lambda(1 - x_1 - x_2) + \alpha\left(\frac{3}{4} - x_2\right) \quad (\alpha \geq 0)$$

λ, α : 라그랑주 승수(Lagrange multiplier)



- **최적화 방법**

$$\min_{x_1, x_2 \in FS} f(x_1, x_2) = \min_{x_1, x_2} \max_{\alpha \geq 0, \lambda} L(x_1, x_2, \lambda, \alpha)$$

FS : 가능해(feasible solution)의 집합

λ, α를 마음대로 바꾸며 $L(x_1, x_2, \lambda, \alpha)$를 아무리 키워도,
$\min_{x_1, x_2} \max_{\lambda, \alpha} L(x_1, x_2, \lambda, \alpha)$의 값은 x_1, x_2가 가능해일 때 나옴

부등식 제약조건에 대한 $\alpha h(x_1, x_2)$에서 $\alpha \geq 0$이기 때문에,

➔ $\min_{x_1, x_2} \max_{\alpha \geq 0} \alpha h(x_1, x_2)$의 값은 $h(x_1, x_2) \leq 0$인,
즉 등식 제약조건이 만족하는 것 중에서 선택하게 됨.

성질 (II)

제약조건있는 최적화

❖ 제약조건 최적화(constrained optimization)

$$L(x_1, x_2, \lambda, \alpha) = f(x_1, x_2) + \lambda h(x_1, x_2) + \alpha g(x_1, x_2) \quad (\alpha \geq 0)$$

- **성질 (I)과 (II)에 의해서**

$$\boxed{\min_{x_1, x_2 \in FS} f(x_1, x_2) = \min_{x_1, x_2} \max_{\alpha \geq 0, \lambda} L(x_1, x_2, \lambda, \alpha)}$$

FS : **가능해(feasible solution)의 집합**

- **최적화 문제의 해법**

$$\min_{x_1, x_2} \max_{\alpha \geq 0, \lambda} L(x_1, x_2, \lambda, \alpha) \geq \max_{\alpha \geq 0, \lambda} \boxed{\min_{x_1, x_2} L(x_1, x_2, \lambda, \alpha)}$$ 4, 3, 1

$$\geq \max_{\alpha \geq 0, \lambda} L_d(\lambda, \alpha)$$ 3, 2, 5

$$\boxed{L_d(\lambda, \alpha) = \min_{x_1, x_2} L(x_1, x_2, \lambda, \alpha)}$$

쌍대함수(dual function)

- **쌍대함수를 최대화하면서 상보적 여유성(complementary slackness)을 만족하는 x_1, x_2를 구함**

$$\alpha g(x_1, x_2) = 0$$

제약조건있는 최적화

❖ 제약조건 최적화(constrained optimization) – cont.

$$L(x_1,x_2,\lambda,\alpha) = \frac{1}{2}(x_1^2+x_2^2)+\lambda(1-x_1-x_2)+\alpha\left(\frac{3}{4}-x_2\right)$$

$$L_d(\lambda,\alpha) = \min_{x_1,x_2} L(x_1,x_2,\lambda,\alpha)$$

$$\frac{\partial L(x_1,x_2,\lambda,\alpha)}{\partial x_1} = x_1-\lambda=0 \qquad x_1=\lambda$$

$$\frac{\partial L(x_1,x_2,\lambda,\alpha)}{\partial x_2} = x_2-\lambda-\alpha=0 \qquad x_2=\lambda+\alpha$$

$$L_d(\lambda,\alpha) = -\lambda^2-\frac{1}{2}\alpha^2-\lambda\alpha+\lambda+\frac{3}{4}\alpha$$

$$\max_{\lambda,\alpha} L_d(\lambda,\alpha)$$

$$\frac{\partial L_d(\lambda,\alpha)}{\partial \lambda} = -2\lambda-\alpha+1=0 \qquad \frac{\partial L_d(\lambda,\alpha)}{\partial \alpha} = -\alpha-\lambda+\frac{3}{4}=0$$

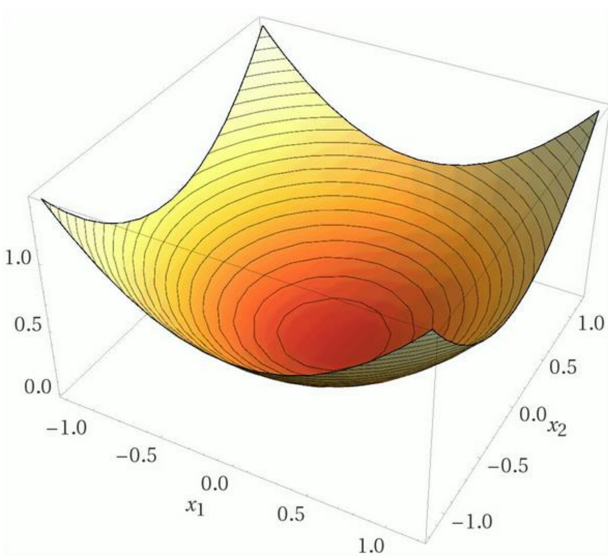
$$\lambda=\frac{1}{4} \qquad \alpha=\frac{1}{2} \qquad\qquad \alpha\left(\frac{3}{4}-x_2\right)=0 \qquad x_2=\frac{3}{4} \qquad 1-x_1-x_2=0 \qquad x_1=\frac{1}{4}$$

$$\therefore\ x_1=\frac{1}{4} \qquad x_2=\frac{3}{4}$$

이차 계획법

❖ 이차 계획법(quadratic programming) 문제

- 목적함수가 **볼록 이차식**(convex quadratic)이고, 제약조건이 모두 일차식인 최적화 문제



Find x_1, x_2 which

$$\text{minimize} \quad \frac{1}{2}(x_1^2 + x_2^2)$$

$$\text{subject to} \quad 1 - x_1 - x_2 = 0$$

$$\frac{3}{4} - x_2 \leq 0$$

- SVM(Support Vector Machine)의 학습에서 사용

이차 계획법

❖ **이차 계획법(quadratic programming) 문제 패키지**

- **Python cvxopt 패키지의 quadratic programming solver**
  - import cvxopt
  - 볼록인(convex)인 함수의 최적화에만 적용 가능

$$\text{Find } \boldsymbol{\alpha} \text{ which minimizes } L_D(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top H \boldsymbol{\alpha} + \boldsymbol{f}^\top \boldsymbol{\alpha}$$

$$\text{subject to } A\boldsymbol{\alpha} \le \boldsymbol{a} \text{ and } B\boldsymbol{\alpha} = \boldsymbol{b}$$

  - **α = cvxopt.solvers.qp(H, f, A, a, B, b)**

$$\text{Find } x_1, x_2$$

$$\text{which minimizes } f(x_1, x_2) = \frac{1}{2}(x_1^2 + x_2^2)$$

$$\text{subject to } g(x_1, x_2) = 1 - x_1 - x_2 = 0$$

$$h(x_1, x_2) = \frac{3}{4} - x_2 \le 0$$

$$\mathrm{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathrm{f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathrm{A} = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \qquad \mathrm{a} = \begin{bmatrix} 0 \\ -3/4 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \mathrm{b} = 1$$

이차 계획법

❖ **cvxopt 패키지의 quadratic programming solver**

```
from cvxopt import matrix, solvers

H = 2*matrix([ [1/2., 0.], [0., 1/2.] ])
f = matrix([0.0, 0.0])
A = matrix([[0.0,0.0],[0.0,-1.0]])
a = matrix([0.0,-3/4.0])
B = matrix([1.0, 1.0], (1,2))
b = matrix(1.0)

sol = solvers.qp(H, f, A, a, B, b)
print('해\n', 'x_1 = ', sol['x'][0])
print(' x_2 = ', sol['x'][1])
```

Find x_1, x_2

which minimizes $f(x_1,x_2) = \frac{1}{2}(x_1^2 + x_2^2)$

subject to $g(x_1,x_2) = 1 - x_1 - x_2 = 0$

$$h(x_1,x_2) = \frac{3}{4} - x_2 \leq 0$$

$$\mathrm{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathrm{f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathrm{A} = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \qquad \mathrm{a} = \begin{bmatrix} 0 \\ -3/4 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \qquad \mathrm{b} = 1$$

```
     pcost       dcost       gap    pres   dres
 0:  5.0500e-01  7.4500e-01  2e+00  2e+00  1e+00
 1:  6.5416e-01  6.2320e-01  6e-02  2e-02  1e-02
 2:  6.2605e-01  6.2500e-01  1e-03  2e-04  1e-04
 3:  6.2501e-01  6.2500e-01  1e-05  2e-06  1e-06
 4:  6.2500e-01  6.2500e-01  1e-07  2e-08  1e-08
Optimal solution found.
해
 x_1 =  0.24999989494427826
 x_2 =  0.7500001050572175
```

Quiz

❖ 다음 이차계획법 문제의 해를 cvxopt 패키지의 solvers.qp(H, f, A, a, B, b) 를 사용하여 구할 때, 파라미터 H, f, A, a, B, b의 값을 구하시오.

$$
\begin{aligned}
\text{minimize} \quad & 2x_1^2 + x_2^2 + x_1x_2 + x_1 + x_2 \\
\text{subject to} \quad & x_1 \geq 0 \\
& x_2 \geq 0 \\
& x_1 + x_2 = 1
\end{aligned}
$$

❖ 함수 최적화에 대한 설명으로 가장 적합하지 않은 것을 선택하시오.

① 함수 최적화 문제에서는 주어진 함수에 대해 일반적으로 최소 또는 최대값을 주는 파라미터의 값을 찾는다.

② 제약조건 최적화 문제는 제약조건을 만족하면서 목적함수를 최적화하는 변수의 값을 찾는 것이다.

③ 제약조건과 목적 함수는 라그랑지 승수를 도입하여 하나의 함수 식으로 표현할 수 있다.

④ 제약조건 최적화 문제에는 반드시 해가 존재한다.

❖ 경사 하강법에 대한 설명으로 가장 적합하지 않은 것을 선택하시오.

① 경사 하강법은 그레디언트 정보를 이용하여 그레디언트 반대 방향으로 파라미터를 조금씩 변경한다.

② 그레디언트를 대상 함수을 각 파라미터별로 편미분 한 것이다.

③ 경사 하강법은 반드시 전역 최적화 해를 찾을 수 있다.

④ 함수 에 대한 최대값의 위치을 찾는 문제는 함수에 경사 하강법을 적용하여 해결할 수 있다.

Quiz

❖ **경사하강법의 주요 목표는 무엇인가?**

① 함수의 극대값을 찾는 것
② 함수의 극소값을 찾는 것
③ 함수의 모든 가능한 값을 찾는 것
④ 함수의 평균값을 찾는 것

❖ **경사하강법에서 그래디언트는 무엇을 나타내는가?**

① 함수의 최대값의 방향
② 함수의 평균값의 방향
③ 함수의 값이 가장 증가하는 방향과 크기
④ 함수의 누적값

❖ **함수의 모양이 '사발' 형태일 때, 경사하강법은 어떻게 동작하는가?**

① 극대값을 찾아 수렴한다.
② 극소값을 찾아 수렴한다.
③ 수렴하지 않고 진동한다.
④ 무작위 방향으로 이동한다.

Quiz

❖ **라그랑주 승수는 어떤 목적으로 사용되는가?**

① 제약조건을 목적 함수에 통합시킨다.
② 탐색 공간을 확장한다.
③ 제약조건을 무시한다.
④ 비용 함수의 기울기를 조정한다.

❖ **제약조건 하에서의 최적화 문제에서 가능해란?**

① 제약조건을 만족하는 해
② 목적 함수를 최대화하는 해
③ 다른 모든 해보다 낮은 비용을 갖는 해
④ 동일한 비용을 갖는 여러 해 중 하나

❖ **제약조건을 가진 최적화 문제의 해는 항상 존재하는가?**

① 항상 존재한다.
② 항상 존재하지 않는다.
③ 문제의 특성에 따라 다르다.
④ 제약조건의 수에 따라 다르다.

Quiz

❖ **2차 계획법**(Quadratic Programming)**은 어떠한 목적함수를 최적화하는가?**

① 선형 함수
② 지수 함수
③ 2차 함수
④ 로그 함수

❖ **2차 계획법에서 최적화되는 목적함수의 계수들은 어떤 행렬로 표현되는가?**

① 정방 행렬
② 대각 행렬
③ 단위 행렬
④ 직사각형 행렬

❖ **2차 계획법 문제의 해는 어디에 위치할 가능성이 가장 높은가?**

① 제약조건의 경계
② 제약조건의 내부
③ 제약조건의 바깥
④ 제약조건과 무관하게 어디든 위치