CS257 Linear and Convex Optimization Homework 9

Xiangyuan Xue (521030910387)

1. The Lagrangian is

$$\mathcal{L}(x_1, x_2, \mu_1, \mu_2) = x_1^2 + (x_2 - 1)^2 + \mu_1(x_1 - x_2 - 1) + \mu_2\left[(x_1 - 1)^2 + x_2^2 - 1\right]$$

The KKT conditions are

$$\begin{cases} 2x_1 + \mu_1 + 2\mu_2(x_1 - 1) = 0 \\ 2(x_2 - 1) - \mu_1 x_2 + 2\mu_2 x_2 = 0 \\ \mu_1 \geq 0 \\ \mu_2 \geq 0 \\ \mu_1(x_1 - x_2 - 1) = 0 \\ \mu_2\left[(x_1 - 1)^2 + x_2^2 - 1\right] = 0 \\ x_1 - x_2 - 1 \leq 0 \\ (x_1 - 1)^2 + x_2^2 - 1 \leq 0 \end{cases}$$

Case I. Both g_1 and g_2 are active. We have

$$\begin{cases} x_1 - x_2 - 1 = 0 \\ (x_1 - 1)^2 + x_2^2 - 1 = 0 \end{cases}$$

which yields two solutions

$$\begin{cases} x_1 = 1 + \dfrac{\sqrt{2}}{2} \\ x_2 = \dfrac{\sqrt{2}}{2} \end{cases}, \quad \begin{cases} x_1 = 1 - \dfrac{\sqrt{2}}{2} \\ x_2 = -\dfrac{\sqrt{2}}{2} \end{cases}$$

which respectively indicates that

$$\begin{cases} \mu_1 = 8 - 4\sqrt{2} \\ \mu_2 = 5 - 3\sqrt{2} \end{cases}, \quad \begin{cases} \mu_1 = 8 + 4\sqrt{2} \\ \mu_2 = 3 + 5\sqrt{2} \end{cases}$$

Both of them satisfies the KKT conditions, and $\left(1 + \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)^T$ and $\left(1 - \frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)^T$ yields the same value $f = 3$, which is either maximal or minimal.

Case II. Both g_1 and g_2 are inactive, then $\mu_1 = \mu_2 = 0$. We have

$$\begin{cases} x_1 = 0 \\ x_2 = 1 \end{cases}$$

which violates the inequality constraints.

Case III. g_1 is active and g_2 is inactive, then $\mu_2 = 0$. We have

$$\begin{cases} 2x_1 + \mu_1 = 0 \\ 2(x_1 - 1) - \mu_1 x_2 = 0 \\ x_1 - x_2 - 1 = 0 \end{cases}$$

which yields two solutions

$$\begin{cases} x_1 = 1 \\ x_2 = 0 \\ \mu_1 = -2 \end{cases}, \quad \begin{cases} x_1 = -1 \\ x_2 = -2 \\ \mu_1 = 2 \end{cases}$$

The former solution violates $\mu_1 \geq 0$. The latter solution violates the inequality constraints. Neither of them can be the optimal point $\boldsymbol{x}^*$.

Case IV. g_1 is inactive and g_2 is active, then $\mu_1 = 0$. We have

$$\begin{cases} x_1 + \mu_2(x_1 - 1) = 0 \\ (x_2 - 1) + \mu_2 x_2 = 0 \\ (x_1 - 1)^2 + x_2^2 - 1 = 0 \end{cases}$$

which yields two solutions

$$\begin{cases} x_1 = 1 - \dfrac{\sqrt{2}}{2} \\ x_2 = \dfrac{\sqrt{2}}{2} \\ \mu_2 = -1 + \sqrt{2} \end{cases}, \quad \begin{cases} x_1 = 1 + \dfrac{\sqrt{2}}{2} \\ x_2 = -\dfrac{\sqrt{2}}{2} \\ \mu_2 = -1 - \sqrt{2} \end{cases}$$

The latter solution violates the KKT conditions, but the former solution satisfies all the constraints, so $\left(1 - \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)^T$ yields $f = 3 - 2\sqrt{2}$, which is either maximal or minimal.

Since the problem is convex, we can conclude that the optimal point $\boldsymbol{x}^* = \left(1 - \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)^T$ and the corresponding Lagrange multipliers $\boldsymbol{\mu}^* = (0, \sqrt{2} - 1)$, which yields the minimum value $f^* = 3 - 2\sqrt{2}$.

2. Reformulate the problem as

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = x_1^2 + x_2^2 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = (x_1 - 1)^2 + (x_2 - 1)^2 - 1 \leq 0 \\ & g_2(\boldsymbol{x}) = (x_1 - 1)^2 + x_2^2 - 1 \leq 0 \end{aligned}$$

(a) $\boldsymbol{x}^{(1)} = (1, 1)^T$ satisfies the inequality constraints. Consider the KKT conditions where g_1 is inactive and g_2 is active, so we have

$$\begin{cases} x_1 + \mu_2(x_1 - 1) = 0 \\ x_2 + \mu_2 x_2 = 0 \end{cases}$$

which yields $\mu_2 = -1 < 0$. Since $\nabla g_2(\boldsymbol{x}^{(1)}) = (0, 2)^T \neq \mathbf{0}$, we know that $\boldsymbol{x}^{(1)}$ is a regular point and cannot be an optimal solution.

(b) $\boldsymbol{x}^{(2)} = (0, 1)^T$ violates $g_1(\boldsymbol{x}) \leq 0$ and is not a feasible point, not to mention an optimal solution.

(c) $\boldsymbol{x}^{(3)} = (1 - \frac{\sqrt{2}}{2}, 1 - \frac{\sqrt{2}}{2})^T$ satisfies the inequality constraints. Consider the KKT condition where g_1 is active and g_2 is inactive, so we have

$$\begin{cases} x_1 + \mu_1(x_1 - 1) = 0 \\ x_2 + \mu_1(x_2 - 1) = 0 \end{cases}$$

which yields $\mu_1 = \sqrt{2} - 1 > 0$, satisfying the KKT conditions. Since the problem is convex, we can conclude that $\boldsymbol{x}^{(3)}$ is an optimal solution.

3. The implementation of projected gradient descent is as follow

```
def proj_gd(fp, proj, x0, stepsize, tol=1e-5, maxiter=100000):
    x_traces = [np.array(x0)]
    y_traces = []
    x = np.array(x0)
    for _ in range(maxiter):
        y = x - stepsize * fp(x)
        xp = proj(y)
        if np.linalg.norm(xp - x) < stepsize * tol:
            break
        x = xp
        y_traces.append(y)  # y is output of gradient step before
    projection
        x_traces.append(np.array(x))  # x is output of projected
    gradient step
    return x_traces, y_traces
```

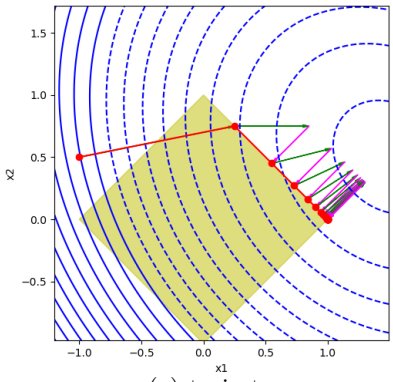
According to the slides, the projection function is implemented as follow

```
def proj(x, t=1):
    if np.linalg.norm(x, ord=1) <= t:
        return x
    y = np.sort(np.abs(x))[::-1]
    mu = (np.cumsum(y) - t) / np.arange(1, len(y) + 1)
    x_base = np.maximum(np.abs(x) - mu[np.where(mu <= y)[0][-1]], 0)
    return np.sign(x) * x_base
```

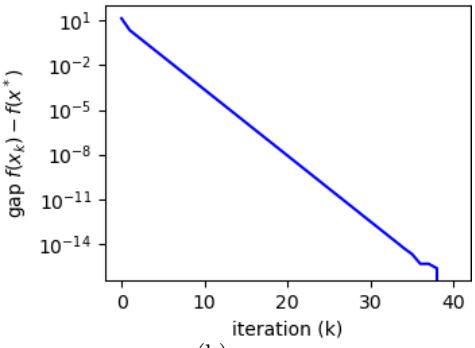
The results are reported as follow

```
iterations: 40
state: converge
solution: [9.99999998e-01 1.67093689e-09]
value: 1.500000
```

which indicates that the solution is $\boldsymbol{w}^* = (1, 0)^T$ and the number of iterations is 40.



(a) trajectory



(b) gap

Figure 1: trajectory and gap for projected gradient descent

Trajectory of $\boldsymbol{w}_k$ and the gap $f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*)$ are shown in the above figures.

4. (a) The Lagrangian is

$$\mathcal{L}(x_1, x_2, x_3, \lambda) = e^{2x_1} + e^{x_2} + e^{x_3} + \lambda(x_1 + x_2 + x_3 - 1)$$

By the Lagrange condition

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial x_1} = 2e^{2x_1} + \lambda = 0 \\ \dfrac{\partial \mathcal{L}}{\partial x_2} = e^{x_2} + \lambda = 0 \\ \dfrac{\partial \mathcal{L}}{\partial x_3} = e^{x_3} + \lambda = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = x_1 + x_2 + x_3 - 1 = 0 \end{cases}$$

which yields

$$\begin{cases} x_1 = \dfrac{1 - 2\ln 2}{5} \\ x_2 = \dfrac{2 + \ln 2}{5} \\ x_3 = \dfrac{2 + \ln 2}{5} \\ \lambda = -\sqrt[5]{2e^2} \end{cases}$$

Since the problem is convex, this solution is optimal. We can conclude that the optimal solution $\boldsymbol{x}^* = (\frac{1-2\ln 2}{5}, \frac{2+\ln 2}{5}, \frac{2+\ln 2}{5})^T$, the Lagrange multiplier $\lambda^* = -\sqrt[5]{2e^2}$ and the optimal value $f^* = \frac{5}{2}\sqrt[5]{2e^2}$.

(b) The Hessian matrix is

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} 4e^{x_1} & 0 & 0 \\ 0 & e^{x_2} & 0 \\ 0 & 0 & e^{x_3} \end{pmatrix}$$

Then the KKT system is

$$\begin{pmatrix} 4e^{x_1} & 0 & 0 & 1 \\ 0 & e^{x_2} & 0 & 1 \\ 0 & 0 & e^{x_3} & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \lambda \end{pmatrix} = \begin{pmatrix} -2e^{x_1} \\ -e^{x_2} \\ -e^{x_3} \\ 0 \end{pmatrix}$$

which yields

$$\begin{cases} d_1 = -\dfrac{2e^{x_1+x_2} + 2e^{x_1+x_3} - 2e^{x_2+x_3}}{4e^{x_1+x_2} + 4e^{x_1+x_3} + e^{x_2+x_3}} \\ d_2 = -\dfrac{4e^{x_1+x_2} - 6e^{x_1+x_3} + e^{x_2+x_3}}{4e^{x_1+x_2} + 4e^{x_1+x_3} + e^{x_2+x_3}} \\ d_3 = -\dfrac{-6e^{x_1+x_2} + 4e^{x_1+x_3} + e^{x_2+x_3}}{4e^{x_1+x_2} + 4e^{x_1+x_3} + e^{x_2+x_3}} \\ \lambda = -\dfrac{10e^{x_1+x_2+x_3}}{4e^{x_1+x_2} + 4e^{x_1+x_3} + e^{x_2+x_3}} \end{cases}$$

where $\boldsymbol{d} = (d_1, d_2, d_3)^T$ is the Newton direction.

(c) The implementation of constrained Newton's method is as follow

```
def newton_eq(f, fp, fpp, x0, A, b, initial_stepsize=1.0, alpha
    =0.5, beta=0.5, tol=1e-8, maxiter=100000):
    x_traces = [np.array(x0)]
    m = len(b)
    x = np.array(x0)
    for _ in range(maxiter):
        gradient = fp(x)
        hessian = fpp(x)
        solution = np.linalg.solve(
            np.block([[hessian, A.T], [A, np.zeros((m, m))]]),
            np.block([[-1 * gradient.reshape(-1, 1)], [np.zeros((m
    , 1))]])
        )
        d = solution[0: len(x)].reshape(-1)
        if np.linalg.norm(d) < tol:
            break
        stepsize = initial_stepsize
        k = 1
        while(f(x + stepsize * d) > f(x) + alpha * stepsize * (
    gradient @ d)):
            stepsize = beta * stepsize
            k += 1
            if k > 5:
                break
        x = x + stepsize * d
```

```
        x_traces.append(np.array(x))
    return x_traces
```

The results are reported as follow

```
state: converge
solution: [-0.07725887  0.53862944  0.53862944]
value: 4.284141440311191
iteration 0:    [0. 1. 0.]
iteration 1:    [-0.11369186  0.56845929  0.54523257]
iteration 2:    [-0.09499115  0.5534111   0.54158005]
iteration 3:    [-0.08601059  0.54598966  0.54002094]
iteration 4:    [-0.0816069   0.5423022   0.53930469]
iteration 5:    [-0.07942602  0.54046402  0.538962  ]
iteration 6:    [-0.07834074  0.53954628  0.53879446]
iteration 7:    [-0.07779938  0.53908775  0.53871163]
iteration 8:    [-0.07752902  0.53885857  0.53867046]
iteration 9:    [-0.07739392  0.53874399  0.53864993]
iteration 10:   [-0.07732639  0.53868671  0.53863968]
iteration 11:   [-0.07729263  0.53865807  0.53863455]
iteration 12:   [-0.07727575  0.53864376  0.538632  ]
iteration 13:   [-0.07726731  0.5386366   0.53863072]
iteration 14:   [-0.07726309  0.53863302  0.53863008]
iteration 15:   [-0.07725887  0.53862944  0.53862944]
```

which are consistent with the results we have figured out in 4a.