دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

تمرین شماره: ۰۲

بهینه‌سازی پیشرفته

نام و نام‌خانوادگی: سیدرضا مسلمی

شماره دانشجویی: ۸۱۰۱۰۳۳۲۶

پاییز ۱۴۰۳

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

حل مسئله اول

۱.۱ مقدمه

به ما داده شده است که:

$$\bar{x}^k = \arg\min_{x \in X} \left\{ \nabla f(x^k)^T (x - x^k) + \frac{1}{2s^k} (x - x^k)^T H^k (x - x^k) \right\}.$$

هدف این است که نشان دهیم این معادل است با:

$$\min_{x \in X} \frac{1}{2} \| x - \left(x^k - s^k (H^k)^{-1} \nabla f(x^k) \right) \|_{H^k}^2,$$

که در آن نماد $\| \cdot \|_{H^k}$ به صورت زیر تعریف می‌شود:

$$\|z\|_{H^k}^2 = z^T H^k z.$$

۲.۱ گام اول: گسترش عبارت درجه دوم

هدف بهینه‌سازی برای $\bar{x}^k$ به شکل زیر است:

$$\min_{x \in X} \left\{ \nabla f(x^k)^T (x - x^k) + \frac{1}{2s^k} (x - x^k)^T H^k (x - x^k) \right\}.$$

۱. گسترش عبارت دوم:

$$(x - x^k)^T H^k (x - x^k) = x^T H^k x - 2(x^k)^T H^k x + (x^k)^T H^k x^k.$$

۲. جایگذاری در هدف بهینه‌سازی:

$$\nabla f(x^k)^T (x - x^k) + \frac{1}{2s^k} \left(x^T H^k x - 2(x^k)^T H^k x + (x^k)^T H^k x^k \right).$$

۳. تنظیم مجدد جملات:

$$= \frac{1}{2s^k} x^T H^k x - \left(\frac{1}{s^k} H^k x^k - \nabla f(x^k) \right)^T x + \text{ثابت}.$$

۳.۱ گام دوم: مقایسه با مسئله معادل

مسئله معادل به صورت زیر است:

$$\min_{x \in X} \frac{1}{2} \| x - \left(x^k - s^k (H^k)^{-1} \nabla f(x^k) \right) \|_{H^k}^2.$$

گسترش نُرم:

$$\| x - \left(x^k - s^k (H^k)^{-1} \nabla f(x^k) \right) \|_{H^k}^2 = (x - v)^T H^k (x - v),$$

که در آن:

$$v = x^k - s^k (H^k)^{-1} \nabla f(x^k).$$

گسترش عبارت:

$$= x^T H^k x - 2v^T H^k x + v^T H^k v.$$

جایگذاری $v = x^k - s^k (H^k)^{-1} \nabla f(x^k)$:

$$v^T H^k = (x^k)^T H^k - s^k \nabla f(x^k)^T.$$

بنابراین هدف به صورت زیر می‌شود:

$$\frac{1}{2} x^T H^k x - \left(H^k x^k - s^k \nabla f(x^k) \right)^T x + \text{ثابت}.$$

۴.۱ گام سوم: تطابق اهداف

مقایسه بین هر دو فرمول به این صورت است که بهینه‌سازی:

$$\nabla f(x^k)^T (x - x^k) + \frac{1}{2s^k} (x - x^k)^T H^k (x - x^k)$$

معادل است با بهینه‌سازی:

$$\frac{1}{2} \| x - \left(x^k - s^k (H^k)^{-1} \nabla f(x^k) \right) \|_{H^k}^2.$$

۵.۱ نتیجه‌گیری

هر دو فرمول معادل هستند. بنابراین، $\bar{x}^k$ مسئله دوم را نیز حل می‌کند.

فصل ۲

شرایط کاروش-کاهن-تاکر (KKT) برای مسئله بهینه‌سازی

۱.۲ مقدمه

برای حل این مسئله، باید KKT برای مسئله بهینه‌سازی داده شده را بررسی کنیم. شرایط KKT به صورت زیر است:

$$\nabla_x L(x^*, \lambda^*) = 0$$

$$\lambda_i^* \geq 0, \quad \forall i \in \mathcal{I}$$

$$\lambda_i^* = 0, \quad \forall i \notin \mathcal{A}(x^*)$$

۲.۲ تعریف مسئله

تابع هدف:

$$f(x) = \left(x_1 - \frac{3}{2}\right)^2 + (x_2 - t)^4.$$

محدودیت‌ها:

$$g_1(x) = 1 - x_1 - x_2 \geq 0, \quad g_2(x) = 1 - x_1 + x_2 \geq 0,$$

$$g_3(x) = 1 + x_1 - x_2 \geq 0, \quad g_4(x) = 1 + x_1 + x_2 \geq 0.$$

نقطه‌ای که باید بررسی کنیم: $x^* = (1,0)^T$.

۳.۲ گام اول: گرادیان $f(x)$

گرادیان تابع هدف $f(x)$ به صورت زیر است:

$$\nabla f(x) = 2\left(x_1 - \frac{3}{2}\right) 4(x_2 - t)^3.$$

در نقطه $x^* = (1,0)^T$، این به صورت زیر می‌شود:

$$\nabla f(x^*) = 2\left(1 - \frac{3}{2}\right) 4(0 - t)^3 = -1 - 4t^3.$$

۴.۲ گام دوم: محدودیت‌های فعال در x^*

برای بررسی اینکه کدام محدودیت‌ها در $x^* = (1,0)^T$ فعال هستند، مقادیر $x_1 = 1$ و $x_2 = 0$ را در محدودیت‌ها جایگذاری می‌کنیم:

۱. $g_1(x^*) = 1 - 1 - 0 = 0$ (فعال)،

۲. $g_2(x^*) = 1 - 1 + 0 = 0$ (فعال)،

۳. $g_3(x^*) = 1 + 1 - 0 = 2$ (غیرفعال)،

۴. $g_4(x^*) = 1 + 1 + 0 = 2$ (غیرفعال).

بنابراین، محدودیت‌های فعال $g_1(x)$ و $g_2(x)$ هستند.

۵.۲ گام سوم: گرادیان‌های محدودیت‌های فعال

گرادیان‌های محدودیت‌های فعال به صورت زیر هستند:

$$\nabla g_1(x) = -1 - 1, \quad \nabla g_2(x) = -11.$$

۶.۲ گام چهارم: شرایط KKT

شرایط KKT به صورت زیر بیان می‌شود:

$$\nabla f(x^*) + \lambda_1 \nabla g_1(x^*) + \lambda_2 \nabla g_2(x^*) = 0,$$

که در آن $\lambda_1 \geq 0$ و $\lambda_2 \geq 0$ ضرب‌کننده‌های لاگرانژ مربوط به محدودیت‌های فعال هستند.

گرادیان‌ها را در شرایط KKT جایگذاری می‌کنیم:

$$-1 - 4t^3 + \lambda_1 - 1 - 1 + \lambda_2 - 11 = 0.$$

این دو معادله را به دست می‌آوریم:

۱. $-1 - \lambda_1 - \lambda_2 = 0,$

۲. $-4t^3 - \lambda_1 + \lambda_2 = 0.$

از معادله اول:

$$\lambda_1 + \lambda_2 = -1. \quad (\lambda_1, \lambda_2 \geq 0 \text{ زیرا } \text{، این غیرممکن است}).$$

بنابراین، برای نقطه $x^* = (1,0)^T$، شرایط KKT برای هر مقداری از t نمی‌توانند برقرار باشند.

۷.۲ نتیجه‌گیری

نقطه $x^* = (1,0)^T$ شرایط KKT را برای هیچ مقداری از t برآورده نمی‌کند.

فصل ۳

پیاده‌سازی و نتایج روش گرادیان شرطی

۱.۳ تعریف مسئله

هدف این است که تابع درجه دو زیر را مینیمم کنیم:

$$f(x) = \frac{1}{2}(x_1^2 + x_2^2 + 0.1x_3^2) + 0.55x_3$$

با محدودیت‌های زیر:

$$x_1 + x_2 + x_3 = 1, \quad x_1, x_2, x_3 \leq 0.$$

این مسئله با استفاده از روش گرادیان شرطی حل می‌شود. پیاده‌سازی شامل مراحل مینیمم‌سازی خطی برای محاسبه اندازه گام بهینه در هر تکرار است.

۲.۳ روش‌شناسی

الگوریتم به صورت زیر پیش می‌رود:

۱. مقداردهی اولیه: با یک نقطه اولیه مجاز x_0 در داخل سیمپلکس شروع می‌کنیم.

۲. محاسبه گرادیان: گرادیان تابع هدف را در تکرار فعلی x_k محاسبه می‌کنیم.

۳. مسئله خطی‌شده: مسئله خطی‌شده را بر روی رئوس سیمپلکس حل می‌کنیم.

۴. محاسبه اندازه گام: اندازه گام را با استفاده از فرمول زیر محاسبه می‌کنیم:

$$\alpha_k = \arg\min_{\alpha \in [0,1]} f(x_k + \alpha(x_{\text{bar}} - x_k)).$$

۵. بروزرسانی: نقطه تکرار $x_{k+1} = x_k + \alpha_k(x_{\text{bar}} - x_k)$ را به روز می‌کنیم.

۶. بررسی همگرایی: هنگامی که تغییر در مقدار تابع هدف کمتر از آستانه تعیین‌شده شد، الگوریتم را متوقف می‌کنیم.

۳.۳ پیاده‌سازی

کد زیر روش گرادیان شرطی را با استفاده از روش‌شناسی فوق پیاده‌سازی می‌کند:

```
def linearized_subproblem(grad, vertices):
    """Solve the linearized subproblem min grad^T x over the
    simplex vertices."""
    values = [np.dot(grad, v) for v in vertices]
    return vertices[np.argmin(values)]

def conditional_gradient_method(x0, max_iter=200, tol=1e-10):
    """Conditional Gradient Method with line minimization
    stepsize."""
    # Define simplex vertices
    vertices = [
        np.array([1, 0, 0]),
        np.array([0, 1, 0]),
        np.array([0, 0, 1])
    ]

    x = x0
    f_values = [objective_function(x)]

    for k in range(max_iter):
        grad = gradient(x)
        # Solve the linearized subproblem
        x_bar = linearized_subproblem(grad, vertices)

        # Compute the stepsize
        alpha = -np.dot(grad, x_bar - x) / (np.linalg.norm(x_bar \
            - x)**2
             + 1e-10)
        alpha = max(0, min(1, alpha))  # Ensure alpha is in [0, 1]

        # Update x
        x_new = x + alpha * (x_bar - x)

        # Store function value for convergence plot
        f_values.append(objective_function(x_new))

        # Check for convergence
        if np.abs(objective_function(x_new) - \
            objective_function(x)) < tol:
            break

        x = x_new
        print(f"Iteration {k+1}: x = {x_new}, f(x) = \
            {objective_function(x_new)}")

    return x, f_values
```

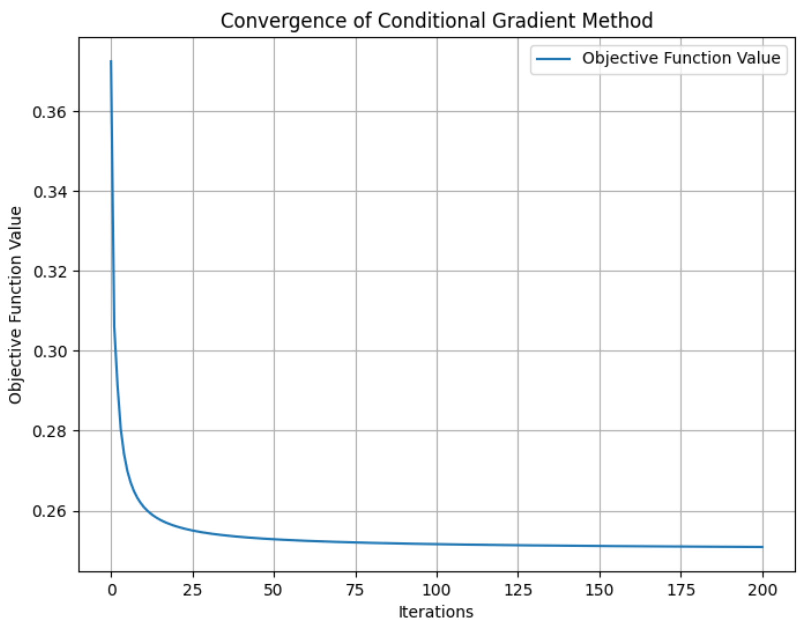
۴.۳ نتایج

- راه‌حل بهینه: $x^* = [0.491, 0.493, 0.016]$

- مقدار بهینه تابع هدف: $f(x^*) \approx 0.251$

۵.۳ رسم همگرایی

شکل ۱.۳ همگرایی مقدار تابع هدف را نشان می‌دهد که در حین پیشرفت الگوریتم انجام می‌شود:



شکل ۱.۳: رسم همگرایی تابع هدف

۶.۳ مشاهدات

این روش به سرعت به یک راه‌حل با دقت مورد نظر همگرا می‌شود.

فصل ۴

برنامه‌ریزی درجه دوم با استفاده از روش مجموعه فعال (الگوریتم ۳.۱۶)

۱.۴ مقدمه

این گزارش به ارائه پیاده‌سازی و نتایج استفاده از الگوریتم ۳.۱۶ (روش مجموعه فعال) برای حل یک مسئله برنامه‌ریزی درجه دوم (QP) محدب می‌پردازد. روش مجموعه فعال یک الگوریتم گام‌به‌گام برای حل مسائل QP با محدودیت‌های خطی است.

۲.۴ تعریف مسئله

مسئله بهینه‌سازی درجه دوم به صورت زیر تعریف شده است:

$$\min \quad f(x) = x_1^2 + 2x_2^2 - 2x_1 - 6x_2 - 2x_1x_2,$$

با محدودیت‌های:

$$\frac{1}{2}x_1 + \frac{1}{2}x_2 \leq 1, \quad -x_1 + 2x_2 \leq 2, \quad x_1 \geq 0, \quad x_2 \geq 0.$$

هدف این است که مقدار $f(x)$ را با توجه به محدودیت‌های داده شده کمینه کنیم. منطقه مجاز توسط محدودیت‌های بالا تعریف می‌شود.

۳.۴ شرح الگوریتم

الگوریتم ۳.۱۶ (روش مجموعه فعال) برای حل این مسئله به صورت زیر عمل می‌کند:

- یک نقطه اولیه مجاز x_0 را محاسبه کنید.
- مجموعه W_0 را به عنوان زیرمجموعه‌ای از محدودیت‌های فعال در x_0 تنظیم کنید.
- برای $k = 0, 1, 2, \ldots$:
  - زیرمسئله QP کاهش‌یافته زیر را حل کنید:

$$\min_p \quad \frac{1}{2}p^T G p + g_k^T p,$$

$$\text{با محدودیت:} \quad a_i^T p = 0, \quad i \in W_k.$$

  - اگر $p_k = 0$:
    * ضریب‌های لاگرانژ λ_i را که شرط زیر را ارضا می‌کنند محاسبه کنید:

$$\lambda_i \geq 0, \quad \forall i \in W_k \cap I.$$

    * اگر این شرط برقرار باشد، الگوریتم متوقف شده و $x^* = x_k$ به عنوان جواب بهینه خروجی داده می‌شود.
    * در غیر این صورت، محدودیت j با کوچک‌ترین $\lambda_j < 0$ از مجموعه W_k حذف می‌شود.
  - اگر $p_k \neq 0$:
    * گام بهینه α_k را با استفاده از فرمول زیر محاسبه کنید:

$$\alpha_k = \min \left(1, \min_{i \notin W_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right).$$

* نقطه جدید را به صورت زیر به‌روزرسانی کنید:

$$x_{k+1} = x_k + \alpha_k p_k.$$

* اگر محدودیت‌های مسدودکننده‌ای وجود داشته باشند، یکی از آن‌ها به مجموعه W_k اضافه می‌شود.

- تکرار تا زمانی که همگرایی حاصل شود.

۴.۴ پیاده‌سازی

کد مربوط به پیاده‌سازی این الگوریتم در ادامه آمده است:

```python
def solve_qp_active_set(G, c, A, b, x0):
    print(f"Initial point: {x0}, Active set: {W}")

    # Iterative process
    for _ in range(1000):
        try:
            # Solve for search direction considering active
            # constraints
            W_mat = A[W] if W else np.zeros((0, len(x)))
            if W_mat.size > 0:
                KKT_mat = np.block([
                    [G, -W_mat.T],
                    [W_mat, np.zeros((len(W), len(W)))]
                ])
                KKT_rhs = np.block([-G @ x - c, np.zeros(len(W))])
                solution = np.linalg.solve(KKT_mat, KKT_rhs)
                pk = solution[:len(x)]
            else:
                pk = -np.linalg.pinv(G) @ (G @ x + c)
        except np.linalg.LinAlgError:
            print("Singular matrix encountered.")
            break

        # Check optimality condition
        if np.linalg.norm(pk) < tol:
            lambdas = np.linalg.lstsq(-A[W].T, G @ x + c, \
                rcond=None)[0] if W else np.array([])

            if all(l >= 0 for l in lambdas):
                print(f"Optimal solution found: {x}, with {_} \
                    iterations.")
                return x_history
            else:
                min_lambda_index = np.argmin(lambdas)
                W.pop(min_lambda_index)
                continue

        # Compute step length ensuring feasibility
        alpha_k = 1
        for i, (a, b_i) in enumerate(zip(A, b)):
            if a @ pk > tol and i not in W:
                alpha_k = min(alpha_k, (b_i - a @ x) / (a @ pk))

        # Take the step
        x = x + alpha_k * pk
        x_history.append(x.copy())

        # Update active set
        for i, (a, b_i) in enumerate(zip(A, b)):
            if np.abs(a @ x - b_i) < tol and i not in W:
                W.append(i)

    print(f"Max iterations reached, solution: {x}")
    return x_history
```

۵.۴ نقاط اولیه

الگوریتم با سه نقطه اولیه مختلف اجرا شد:

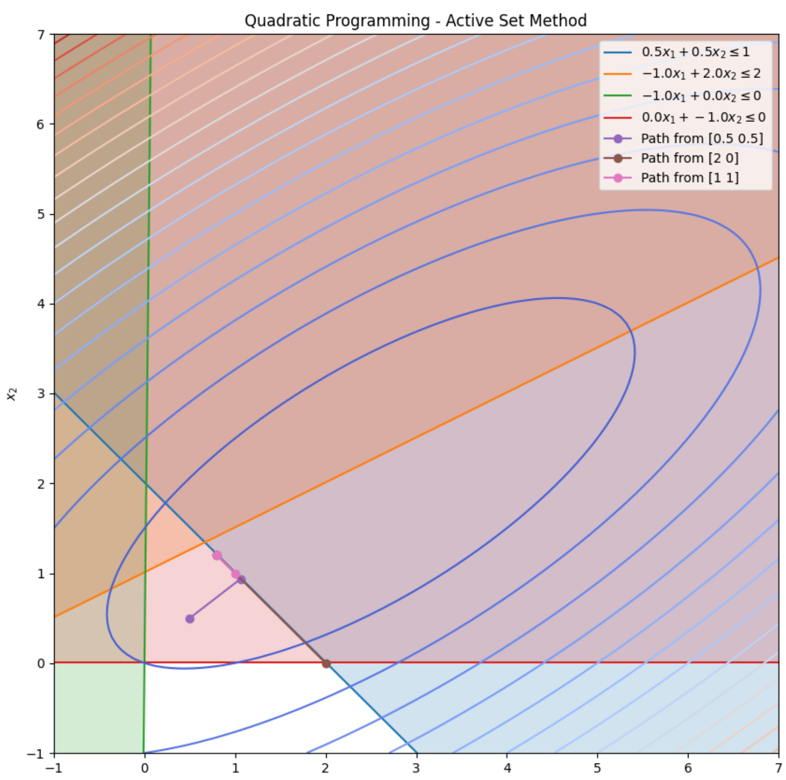
۱. نقطه‌ای در داخل منطقه مجاز: $(0.5, 0.5)$،

۲. نقطه‌ای در یک رأس: $(2, 0)$،

۳. نقطه‌ای غیر رأسی روی مرز: $(1, 1)$.

۶.۴ نتایج و تحلیل

الگوریتم برای تمام نقاط اولیه به راه‌حل بهینه همگرا شد. مسیرهای طی‌شده توسط الگوریتم برای هر نقطه اولیه در شکل ۱.۴ نشان داده شده است.



شکل ۱.۴: مسیرهای طی‌شده توسط روش مجموعه فعال برای نقاط اولیه مختلف.

۱.۶.۴ راه‌حل بهینه

الگوریتم راه‌حل بهینه زیر را شناسایی کرد:

$$x^* = (0.8, 1.2), \quad f(x^*) = 7.2$$

۲.۶.۴ جدول تکرارها

جدول ۱.۴: خلاصه تکرارها برای هر نقطه اولیه

نقطه اولیه	تعداد تکرارها	راه‌حل بهینه	مقدار تابع هدف
$(0.5, 0.5)$	2	$(0.8, 1.2)$	-7.2
$(2, 0)$	2	$(0.8, 1.2)$	-7.2
$(1, 1)$	1	$(0.8, 1.2)$	-7.2

۷.۴ نتیجه‌گیری

روش مجموعه فعال به طور مؤثری مسئله QP داده شده را حل کرد. مسیرهای همگرایی توانایی روش در مدیریت نقاط اولیه مختلف را نشان می‌دهند، و در تمام موارد به همان راه‌حل بهینه همگرا شدند. نتایج درستی و پایداری پیاده‌سازی را تأیید می‌کنند.

کتاب‌نامه