دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

تمرین شماره: ۰۴

بهینه‌سازی پیشرفته

نام و نام‌خانوادگی: سیدرضا مسلمی

شماره دانشجویی: ۸۱۰۱۰۳۳۲۶

پاییز ۱۴۰۳

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مسئله‌ی بهینه‌سازی روی یک کره‌ی واحد

۱.۱ مقدمه

در این مسئله، هدف بهینه‌سازی تابع زیر را داریم:

$$\min_{x\in R^n} \sum_{j=1}^{m} \|x-a_j\|^2 \quad \text{به طوری که} \|x\|^2 = 1.$$

مرکز ثقل بردارهای $a_1, \ldots, a_m$ به صورت زیر تعریف می‌شود:

$$\hat{a} = \frac{1}{m}\sum_{j=1}^{m} a_j.$$

هدف این است که نشان دهیم اگر $\hat{a} \neq 0$، این مسئله یک مینیمم و یک ماکزیمم یکتا دارد. همچنین، بررسی می‌کنیم چه اتفاقی رخ می‌دهد اگر $\hat{a} = 0$.

۲.۱ بسط تابع هدف

ابتدا تابع $f(x)$ را بسط می‌دهیم:

$$f(x) = \sum_{j=1}^{m} \|x - a_j\|^2 = \sum_{j=1}^{m} \left(x \cdot x - 2x \cdot a_j + a_j \cdot a_j\right)$$

از آنجایی که $\|x\|^2 = x \cdot x = 1$، داریم:

$$f(x) = m - 2x \cdot \sum_{j=1}^{m} a_j + \sum_{j=1}^{m} \|a_j\|^2$$

$$\sum_{j=1}^{m} \langle x, a_j \rangle = mx \cdot \hat{a}.$$

با تعریف $s = \sum_{j=1}^{m} a_j = m \cdot \hat{a}$، این معادله به صورت زیر بازنویسی می‌شود:

$$f(x) = m - 2x \cdot s + C$$

که در آن $C = \sum_{j=1}^{m} \|a_j\|^2$ یک ثابت است.

۳.۱ بازنویسی مسئله

از آنجایی که m و C ثابت هستند، کمینه‌سازی $f(x)$ معادل با بیشینه‌سازی $x \cdot s$ و بیشینه‌سازی آن معادل کمینه‌سازی $x \cdot s$ است، تحت شرط $\|x\|^2 = 1$.

۴.۱ تحلیل بهینه‌سازی

عبارت $x \cdot s$ هنگامی که x در جهت s باشد به بیشینه می‌رسد و هنگامی که x در جهت مخالف s باشد به کمینه می‌رسد. بنابراین بر اساس نابرابری کوشی شوارتز خواهیم داشت:

- بیشینه: $x \cdot s = -\|s\|$ زمانی که $x = -\frac{s}{\|s\|}$ باشد.
- کمینه: $x \cdot s = \|s\|$ زمانی که $x = \frac{s}{\|s\|}$ باشد.

۵.۱ تحلیل حالت‌های خاص

۱. اگر $s \neq 0$ (یا $\bar{a} \neq 0$): در این حالت، یک بیشینه‌ی یکتا در $x = -\frac{s}{\|s\|}$ و یک کمینه‌ی یکتا در $x = \frac{s}{\|s\|}$ وجود دارد.

۲. اگر $s = 0$ (یا $\bar{a} = 0$): از آنجایی که $x \cdot s = 0$ برای تمام x برقرار است، تابع $f(x) = m + C$ روی کره واحد ثابت است و بنابراین بیشینه یا کمینه‌ی مشخصی ندارد.

فصل ۲

مقایسه روش لاگرانژ افزوده و روش ضرایب

۱.۲ تعریف مسئله

در این مسئله بهینه‌سازی، تابع هدف به صورت زیر داده شده است:

$$\min_x f(x) = x_1^2 + 2x_2^2 + 3x_3^2$$

با محدودیت:

$$x_1 + x_2 + x_3 = 1$$

و نقطه شروع:

$$x_0 = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}$$

روش‌های به کار رفته:

- روش لاگرانژ افزوده ($AugmentedLagrangeMethod$)
- روش ضرایب ($MultiplierMethod$)

۲.۲ روش لاگرانژ افزوده

تابع لاگرانژ افزوده به صورت زیر تعریف می‌شود:

$$L(x, \lambda, \mu) = f(x) + \lambda(g(x)) + \frac{\mu}{2}(g(x))^2$$

که در آن:

$$f(x) = x_1^2 + 2x_2^2 + 3x_3^2$$

$$g(x) = x_1 + x_2 + x_3 - 1 = 0$$

$$\lambda = Lagrangemultiplier$$

$$\mu = Penaltyparameter$$

ضرایب لاگرانژ λ و پارامتر جریمه μ در هر تکرار به‌روزرسانی می‌شوند.

```python
def augmented_lagrangian_newton(x0, mu=1.0, lam=0.0, rho=10.0, \
    tol=1e-6, max_iter=100):

    x = x0
    for iteration in range(max_iter):
        # Augmented Lagrangian gradient and Hessian
        lagrangian_grad = grad_objective(x) + lam * \
            grad_constraint_eq() + mu * constraint_eq(x) * \
            grad_constraint_eq()

        hessian_lagrangian = hessian_objective() + mu * \
            np.outer(grad_constraint_eq(), grad_constraint_eq())


        # Newton step
        delta_x = np.linalg.solve(hessian_lagrangian, \
            -lagrangian_grad)

        x = x + delta_x

        # Update Lagrange multiplier and penalty parameter
        lam += mu * constraint_eq(x)
        mu *= rho

        # Check convergence
        if np.linalg.norm(delta_x) < tol and np.abs(
            constraint_eq(x)) < tol:
            break

    return x, objective(x), iteration
```

۳.۲ روش ضرایب $(Multiplier Method)$

در این روش، ترکیبی از به‌روزرسانی ضرایب و پارامتر جریمه استفاده می‌شود. تابع جریمه به صورت زیر تعریف می‌شود:

$$PenaltyFunction = f(x) + \mu(g(x))^2$$

```python
def multiplier_method_newton(x0, mu=1.0, tol=1e-6, max_iter=100):
    x = x0
    for iteration in range(max_iter):
        # Penalty gradient and Hessian
        penalty_grad = grad_objective(x) + mu*constraint_eq(x) *\
        grad_constraint_eq()

        penalty_hessian = hessian_objective() + mu * np.outer(
            grad_constraint_eq(), grad_constraint_eq())


        # Newton step
        delta_x = np.linalg.solve(penalty_hessian, -penalty_grad)
        x = x + delta_x

        # Check convergence
        if np.linalg.norm(delta_x) < tol and np.abs(
            constraint_eq(x)) < tol:

            break

        # Increase penalty (more conservatively for stability)
        mu *= 2

    return x, objective(x), iteration
```

۴.۲ نتایج محاسبات

- روش لاگرانژ افزوده:
  - حل بهینه:

$$x^* = \begin{bmatrix} 0.5455 \\ 0.2727 \\ 0.1818 \end{bmatrix}$$

  - مقدار تابع هدف: $f(x^*) = 0.5455$
  - تعداد تکرارها: 4

- روش ضرایب:
  - حل بهینه:

$$x^* = \begin{bmatrix} 0.5454 \\ 0.2727 \\ 0.1818 \end{bmatrix}$$

  - مقدار تابع هدف: $f(x^*) = 0.5454$
  - تعداد تکرارها: 21

۵.۲ تحلیل و مقایسه نرخ همگرایی

- روش لاگرانژ افزوده تنها در ۴ تکرار به نتیجه رسید. دلیل این امر جریمه‌های قوی است که در این روش استفاده می‌شود و به سرعت محدودیت را اعمال می‌کند. با این حال، این روش در مسائل بزرگ‌تر ممکن است ناپایدار باشد.
- روش ضرایب، با وجود نیاز به ۲۱ تکرار، از پایداری بیشتری برخوردار است. این روش به دلیل به‌روزرسانی تدریجی ضرایب و جریمه‌ها، در مسائل پیچیده یا بدشرط کارایی بهتری خواهد داشت.

فصل ۳

پیاده‌سازی و نتایج روش نقطه داخلی برای دسته‌بندی SVM با حاشیه نرم

۱.۳ تعریف مسئله

هدف این است که مسئله بهینه‌سازی زیر را برای پیدا کردن ابرصفحه جداکننده بهینه حل کنیم:

$$\phi(w,b,\xi) = \min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

با محدودیت‌های زیر:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n$$

که در آن: - w و b ابرصفحه جداکننده را تعریف می‌کنند.

- ξ_i متغیرهای نرمش هستند که اجازه می‌دهند حاشیه نرم به دست آید.

– C پارامتر تنظیم‌کننده است که به مقدار $C = 1.0$ تنظیم شده است.

۲.۳ روش‌شناسی

برای حل این مسئله، از روش نقطه داخلی ($InteriorPointMethod$) استفاده می‌کنیم. این روش از یک ($BarrierFunction$) استفاده می‌کند تا محدودیت‌های نامساوی را به صورت جریمه‌ای در تابع هدف بگنجاند. مراحل اصلی الگوریتم عبارت‌اند از:

۱. پیش‌پردازش داده‌ها:

– داده‌های MNIST بارگذاری شده و فقط نمونه‌های مربوط به ارقام ۰ و ۱ انتخاب می‌شوند.

– ویژگی‌ها با استفاده از StandardScaler استانداردسازی می‌شوند.

– داده‌ها به داده‌های آموزشی و آزمایشی تقسیم می‌شوند.

۲. ($BarrierFunction$):

تابع هدف شامل تابع اصلی SVM به همراه یک جمله جریمه‌ای برای مدیریت محدودیت‌هاست:

$$f(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \epsilon\Big(\sum_{i=1}^{m}\log(\xi_i) + \sum_{i=1}^{m}\log(1 - y_i(w^T x_i))\Big)$$

۳. فرایند بهینه‌سازی:

– مقادیر اولیه w و ξ به ترتیب با صفر و یک مقداردهی اولیه می‌شوند.

– پارامتر Barrier ϵ با مقدار اولیه ۲۰۰ شروع می‌شود و هر ۵ تکرار با ضریب $\mu = 0.1$ کاهش می‌یابد.

– به‌روزرسانی پارامترها با استفاده از روش نیوتن با جستجوی خطی برای اطمینان از مثبت بودن ξ انجام می‌شود.

۴. شرایط توقف:

الگوریتم هنگامی که ϵ از مقدار آستانه 10^{-5} کمتر شود یا به‌روزرسانی‌ها ناچیز باشند، متوقف می‌شود.

۵. دسته‌بندی و ارزیابی:

پس از بهینه‌سازی، دسته‌بندی بر اساس علامت پیش‌بینی به شکل زیر انجام می‌شود:

$$\hat{y}_i = \text{sign}(w^T x_i + b)$$

دقت با استفاده از رابطه زیر محاسبه می‌شود:

$$\text{دقت} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_i = y_i)$$

۳.۳ الگوریتم پیاده‌سازی روش نقطه داخلی برای SVM با حاشیه نرم

الگوریتم پیاده‌سازی روش نقطه داخلی به صورت زیر است:

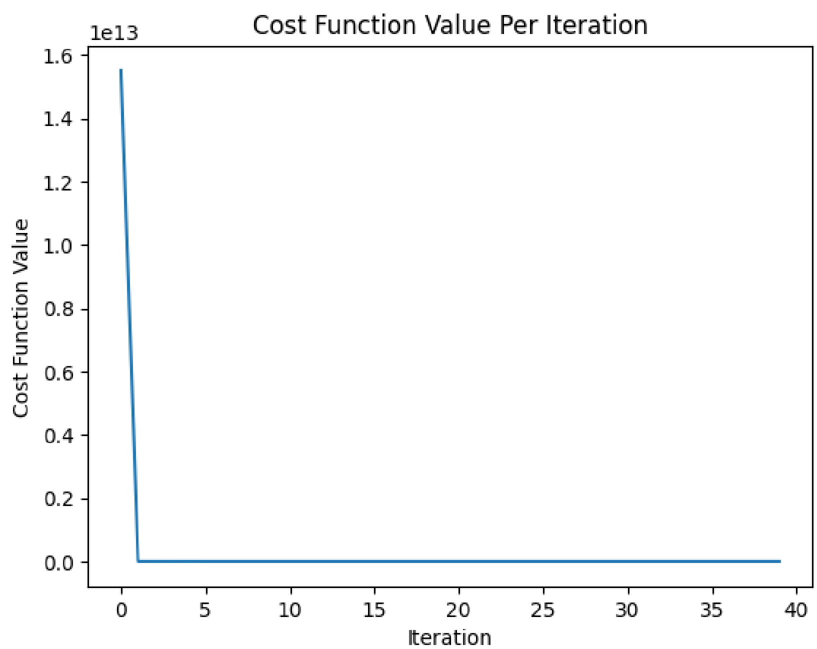
الگوریتم ۱ الگوریتم پیاده‌سازی روش نقطه داخلی برای SVM

Input: داده‌های X و برچسب‌های y
Output: پارامترهای بهینه w و ξ و دقت
بارگذاری داده‌ها و انتخاب فقط ارقام ۰ و ۱
استانداردسازی داده‌ها و تقسیم به مجموعه‌های آموزش و آزمایش
افزودن بایاس به داده‌ها و مقداردهی اولیه به $w = 0$، $\xi = 1$
تنظیم مقادیر اولیه $\epsilon = 200$، $\mu = 0.1$، $C = 1.0$
while $\epsilon > tol$ **do**
  محاسبه گرادیان‌ها
  به‌روزرسانی w و ξ با استفاده از روش نیوتن
  جستجوی خطی برای اطمینان از مثبت بودن ξ
  به‌روزرسانی $w \leftarrow w + \alpha\Delta w$، $\xi \leftarrow \xi + \alpha\Delta\xi$
  ذخیره مقدار تابع هزینه
  کاهش مقدار Barrier ϵ هر ۵ تکرار: $\epsilon \leftarrow \epsilon \times \mu$
end
انجام پیش‌بینی بر روی داده‌های آزمایشی و محاسبه دقت

۴.۳ نتایج

– دقت داده‌های آزمایشی: دقت محاسبه‌شده بر روی داده‌های آزمایشی به مقدار ۹۹،۳۶٪ رسیده است که نشان‌دهنده عملکرد قوی مدل در جدا کردن ارقام ۰ و ۱ است.

– همگرایی تابع هزینه: شکل ۱.۳ روند کاهش مقدار تابع هزینه را در طی تکرارهای الگوریتم نشان می‌دهد.



شکل ۱.۳: نمودار مقدار تابع هزینه بر حسب تکرار

۵.۳ خلاصه نتایج

– همگرایی: الگوریتم نقطه داخلی در ۴۰ تکرار همگرا شده است.

– دقت نهایی: ۹۹،۳۶٪

– رفتار تابع هزینه: تابع هزینه به سرعت کاهش پیدا کرده و پایدار شده است، که نشان‌دهنده همگرایی کارآمد الگوریتم است.

– فرمول دقت: دقت با استفاده از معیار زیر محاسبه شده است:

$$y_i' = \begin{cases} 1 & \text{اگر} \, w^T x_i + b \geq 0 \\ -1 & \text{در غیر این صورت} \end{cases}$$

۶.۳ مشاهدات نهایی

نتایج نشان می‌دهند که استفاده از روش $InteriorPoint$ برای بهینه‌سازی مسئله **SVM** با حاشیه نرم در این داده‌ها، همگرایی سریع و دقت بالا به همراه داشته است.

کتاب‌نامه