به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

گزارش پروژه کانوکس

پروژه سری پنجم

نام و نام خانوادگی	علیرضا فداکار
شماره دانشجویی	۸۱۰۱۹۹۳۵۳
تاریخ ارسال گزارش	

فهرست گزارش سوالات

قسمت a

در این قسمت می‌خواهیم تابع هدف یعنی:

$$f(x,y)=\sum_{i=1}^{m}\left|b_i-x^T u_i v_i^T y\right|$$

را به فرم زیر بنویسیم:

$$f(x,y)=\sum_{i=1}^{m}h\big(c_i(x,y)\big)$$

فرض کنید:

$$h(x)=|x|$$

و:

$$c_i(x,y)=x^T u_i v_i^T y\,,\qquad m\geq i\geq 1$$

با این تعریف داریم:

$$h: R\to R_+$$

و:

$$c_i: R^{n_1+n_2}\to R$$

چون تابع $|x|$ (با توجه به نمودار آن) کانوکس است بنابرین $h(x)$ کانوکس است. در ادامه گرادیان $\nabla c_i(x,y)$ را بدست می‌آوریم. از جبر خطی میدانیم گرادیان $a^T x$ یا $x^T a$ برابر a است. بنابرین:

$$\nabla c_i(x,y)=\begin{bmatrix}\dfrac{\partial(x^T u_i v_i^T y)}{\partial x}\\ \dfrac{\partial(x^T u_i v_i^T y)}{\partial y}\end{bmatrix}=\begin{bmatrix}u_i v_i^T y\\ v_i u_i^T x\end{bmatrix}=\begin{bmatrix}u_i v_i^T y\\ v_i u_i^T x\end{bmatrix}=\begin{bmatrix}(v_i^T y)u_i\\ (u_i^T x)v_i\end{bmatrix}$$

در بالا از این نکته نیز استفاده کردیم که $v_i^T y$ و $u_i^T x$ اسکالر هستند.

در ادامه نگاشت خطی $A: R^{n_1\times n_2}\to R$ به صورت زیر تعریف می‌شود:

$$A(X)=[\operatorname{tr}A_1^T X\quad \operatorname{tr}A_1^T X\quad \dots\quad \operatorname{tr}A_m^T X]^T$$

همچنین ماتریس‌های U و V به صورت زیر تعریف می‌شود:

$$U=[u_1\quad u_2\quad \cdots\quad u_m]^T\in R^{m\times n_1}$$

$$V=[v_1\quad v_2\quad \cdots\quad v_m]^T\in R^{m\times n_2}$$

قسمت b

قبل از اینکه ثابت کنیم بروزرسانی بیان شده ، از نوع کاهشی است ، ابتدا تابع هدف را با استفاده از تعریف $\ell_1\ norm$ ساده‌تر می‌کنیم:

$$f(x,y) = \sum_{i=1}^{m} \left| b_i - tr\ A_i^T x y^T \right| = \| b - A(xy^T) \|_1$$

طبق تعریف صورت سوال داریم:

$$f_{(x,y)}(x+\Delta_x, y+\Delta_y) := \left\| b - \mathcal{A}(xy^T) - \mathbf{diag}(Vy)U\Delta_x - \mathbf{diag}(Ux)V\Delta_y \right\|_1,$$

اگر در رابطه بالا به جای Δ_x و Δ_y صفر قرار دهیم داریم:

$$f_{(x,y)}(x,y) = \| b - A(xy^T) \|_1 = f(x,y)\ (\star)$$

با توجه به اینکه:

$$(\Delta x^k, \Delta y^k) := \operatorname*{argmin}_{\Delta_x, \Delta_y} \left\{ f_{(x^k,y^k)}(x^k+\Delta_x, y^k+\Delta_y) + \frac{1}{2}\Delta_x^T U^T U \Delta_x + \frac{1}{2}\Delta_y^T V^T V \Delta_y \right\}$$

اگر فرض کنیم:

$$I = f_{(x^k,y^k)}\left(x^k+\Delta_x^k, y^k+\Delta_y^k\right) + \frac{1}{2}{\Delta_x^k}^T U^T U \Delta_x^k + \frac{1}{2}{\Delta_y^k}^T V^T V \Delta_y^k$$

آنگاه به ازای هر δ_x و δ_y دلخواه داریم:

$$I \le f_{(x^k,y^k)}\left(x^k+\delta_x^k, y^k+\delta_y^k\right) + \frac{1}{2}\delta_x^T U^T U \delta_x + \frac{1}{2}\delta_y^T V^T V \delta_y$$

اگر در نامساوی بدست آمده اگر قرار دهیم $\delta_x = \delta_y = 0$ و همچنین با توجه به $(\star)$ آنگاه داریم:

$$I \le f_{(x^k,y^k)}(x^k,y^k) = f(x^k,y^k)\ \ (3)$$

از طرفی با توجه به کران 2 صورت سوال یعنی

$$f(x+\Delta_x, y+\Delta_y) \le f_{(x,y)}(x+\Delta_x, y+\Delta_y) + \frac{1}{2}\Delta_x^T U^T U \Delta_x + \frac{1}{2}\Delta_y^T V^T V \Delta_y \qquad (2)$$

داریم:

$$I \ge f\left(x^k+\Delta_x^k, y^k+\Delta_y^k\right) = f(x^{k+1}, y^{k+1})\ \ (4)$$

پس با توجه به روابط (3) و (4) داریم:

$$f(x^k, y^k) \geq f(x^{k+1}, y^{k+1})$$

بنابرین بروزرسانی مذکور ، کاهشی (descent) است.

قسمت c

در این قسمت با استفاده از دیتای فایل matrix_sco_data الگوریتم بیان شده در قسمت b را در متلب پیاده سازی می‌کنیم. ابتدا به کمک کد زیر دیتا را import می‌کنیم:

```
clc
clear
close all
% Import data
matrix_sco_data;
```

سپس تعداد اجراهای برنامه (که طبق صورت پروژه برابر 10 قرار می‌دهیم) و مقدار ترشولد ($\epsilon = 10^{-4}$) را مشخص می‌کنیم:

```
%% part c
number_of_runs = 10;
max_iter = 1e2;
epsilon = 1e-4;
threshold = epsilon^2;
```

در ادامه الگوریتم را به صورت زیر پیاده سازی می‌کنیم:

```
figure
for num = 1:number_of_runs
% initiate x, y
x = randn(n_1, 1);
y = randn(n_2, 1);
termination = inf;
obj_vals = zeros(1, max_iter+1);
obj_vals(1) = norm(b - A_map(U, V, x*y.'), 1);
iter = 1;

while((termination > threshold) && (iter <= max_iter))
    cvx_begin quiet
        variables delta_x(n_1) delta_y(n_2)
```

```
        minimize(norm(b - A_map(U, V, x*y.') -
diag(V*y)*U*delta_x - diag(U*x)*V*delta_y , 1) +
0.5*delta_x.'*U.'*U*delta_x +
0.5*delta_y.'*V.'*V*delta_y);
    cvx_end

    % update x , y
    x = x + delta_x;
    y = y + delta_y;

    % calculate objective
    obj_vals(iter+1) = norm(b - A_map(U, V, x*y.'), 1);
    termination = norm(delta_x, 2)^2 + norm(delta_y,
2)^2;
    iter = iter + 1;
end
```

توضیحات کد:

توجه داشته باشید در ابتدای الگوریتم با استفاده از دستور randn بردارهای x و y را با مقادیر تصادفی از توزیع نرمال استاندارد $N(0,1)$ مقداردهی اولیه می‌کنیم و سپس برای شرط توقف الگوریتم یک متغیر به نام termination و همچنین یک متغیر به نام max_iter که کران بالا برای تعداد تکرارها را نشان می‌دهد تعریف کرده ایم. متغیر termination را با inf مقداردهی اولیه می‌کنیم و در انتهای هر تکرار آن را با مقدار زیر آپدیت می‌کنیم:

$$\|\Delta x^k\|_2^2 + \|\Delta y^k\|_2^2$$

همانطور که در کد مشاهده می‌شود الگوریتم اصلی را داخل یک حلقه while قرار می‌دهیم و شرط خاتمه را نیز به صورت زیر قرار می‌دهیم:

$$\|\Delta x^k\|_2^2 + \|\Delta y^k\|_2^2 \leq \epsilon^2 \quad \text{where} \quad \epsilon = 10^{-4}.$$

در ابتدای این حلقه ، با استفاده از cvx مسئله زیر را حل کرده و طول گامهای Δ_x و Δ_y را می‌یابیم. در قسمت b ثابت کردیم که با این طول گام تابع هدف حتما کاهش می‌یابد.

$$(\Delta x^k, \Delta y^k) := \underset{\Delta_x, \Delta_y}{\operatorname{argmin}} \left\{ f_{(x^k, y^k)}(x^k + \Delta_x, y^k + \Delta_y) + \frac{1}{2}\Delta_x^T U^T U \Delta_x + \frac{1}{2}\Delta_y^T V^T V \Delta_y \right\}$$
$$(x^{k+1}, y^{k+1}) := (x^k + \Delta x^k, y^k + \Delta y^k)$$

که در مسئله بهینه سازی بالا:

$$f_{(x,y)}(x + \Delta_x, y + \Delta_y) := \left\| b - \mathcal{A}(xy^T) - \mathbf{diag}(Vy)U\Delta_x - \mathbf{diag}(Ux)V\Delta_y \right\|_1 ,$$

لازم به ذکر است برای محاسبه مپینگ $A(X)$ که تعریف آن به صورت

$$A(X) = [\operatorname{tr} A_1^T X \quad \operatorname{tr} A_1^T X \quad \dots \quad \operatorname{tr} A_m^T X]^T$$

است یک تابع به نام A_map تعریف کرده ایم که کد آن در ادامه قابل مشاهده است:

```
function out = A_map(U, V, X)
    [m, ~] = size(U);
    out = zeros(m, 1);

    for i = 1:m
        out(i) = U(i,:)*X*V(i,:)';
    end
```

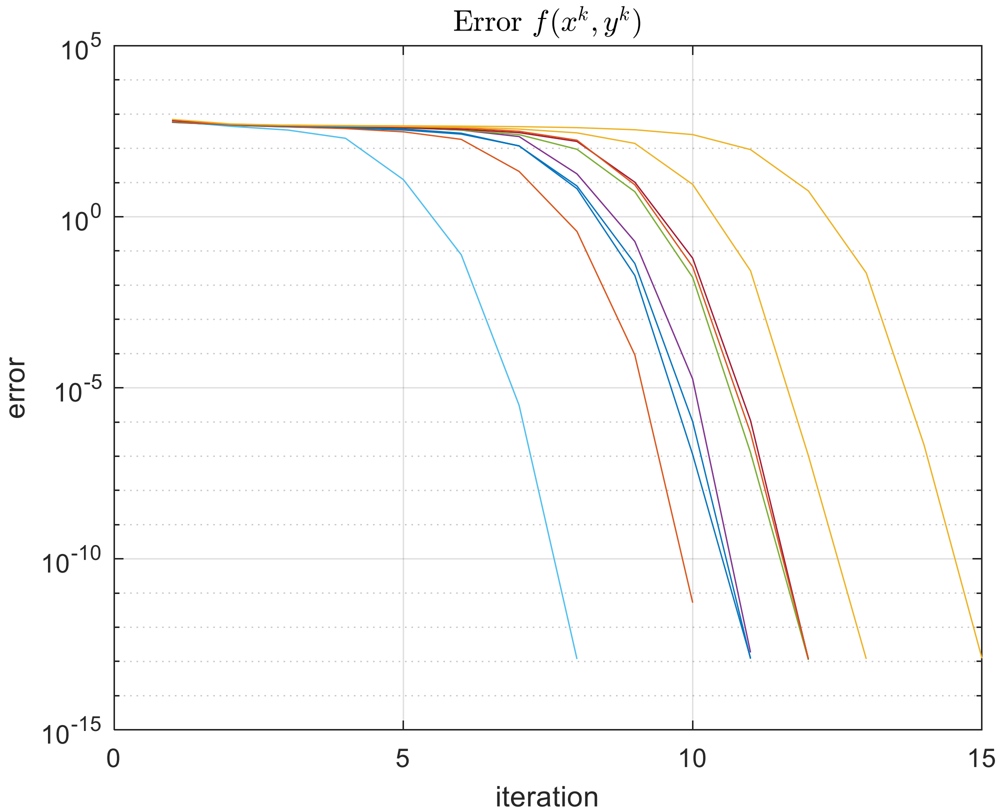
سپس بردارهای x و y را مطابق زیر بروزرسانی می‌کنیم:

$$(x^{k+1}, y^{k+1}) := (x^k + \Delta x^k, y^k + \Delta y^k)$$

سپس در آخر تکرار ، تابع هدف را بدست آورده و آن را برای اینکه در انتها نمودار $f(x,y)$ را رسم کنیم ، ذخیره می‌کنیم.

این الگوریتم تا مرحله‌ای تکرار می‌شود که شرط توقف برقرار گردد.

پس از اجرای الگوریتم به تعداد 10 بار با نقاط اولیه مختلف ، نمودارهای خطای $f(x^k, y^k)$ را با استفاده از دستور semiology در یک نمودار رسم می‌کنیم که در شکل زیر قابل مشاهده است:



همانطور که مشاهده می‌شود همه 10 نمودار ، در حداکثر 15 تکرار همگرا شدند. ولی با توجه به شکل به ازای مقادیر اولیه متفاوت نمودارها منطبق نشده‌اند که قابل انتظار نیز بود. چون مسئله اصلی non convex است و روش SCP یک روش heuristic است و بنابرین به مقادیر اولیه حساس است و ممکن است در تعداد تکرارهای مختلف همگرا شود.