دانشگاه صنعتی شریف
دانشکده ریاضی و علوم کامپیوتر

گزارش پروژه درس بهینه سازی غیرخطی

روش تصویرسازی و گرادیان کاهش یافته

استاد درس
دکتر نظام الدین مهدوی امیری

نگارنده
مرضیه عبدالحمدی

دیماه ۱۴۰۰

فهرست مطالب

چکیده

در این پروژه، به مطالعه و بررسی چند روش اولیه برای حل مسائل بهینه سازی غیرخطی از جمله روش گرادیان تقلیل یافته و روش تصویرگرادیان میپردازیم و ایده های اساسی در فرایند ساخت الگوریتم های مذکور این روشها را بیان میکنیم.

فرایند فوق هم برای قیود خطی و هم غیرخطی بررسی میشود و مثال هایی از هرکدام آورده میشود. سپس به بیان مشکلات پیاده سازی الگوریتم ها در حالت قیود غیرخطی میپردازیم و روش هایی برای حل این مشکلات بصورت نظری داده میشود.

همچنین کدهای Python و Matlab این الگوریتم ها قرار داده شده اند که مثال هایی با آن ها بررسی شده است و روش های گرادیان تقلیل یافته و کاهش گرادیان در آن ها مقایسه شده اند.

در انتها نیز یک نتیجه گیری کلی از آهنگ همگرایی و سرعت این روش ها خواهیم داشت.

صورت کلی مسأله

مسأله برنامه ریزی غیرخطی زیر را درنظر میگیریم:

$$\begin{aligned} minimize \quad & f(x) \\ s.t. \quad & g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

که در آن n متغیر داریم و p قید نامساوی و m قید تساوی داریم. در سرتاسر این متن، فرض بر آن است که همه توابع دارای مشتقات جزئی پیوسته از مرتبه ۳ هستند.

روش های اولیه

ابتدا به این سوال پاسخ میدهیم که منظور از روش های اولیه چیست؟

- یک روش جستجو است
- بطور مستقیم با جستجو در ناحیه شدنی برای جواب بهینه, بر روی مسأله اولیه انجام میشود
- هر نقطه در این فرایند شدنی است
- مقدار تابع هدف دائما کاهش میابد

در یک مسأله با n متغیر و فقط m قید تساوی، روش های اولیه در فضای شدنی، که دارای بعد $n - m$ است عمل میکنند.

حال به بیان مزیت های روش های اولیه میپردازیم:

۱. اگر فرایند قبل از رسیدن به جواب متوقف شود، نقطه فعلی شدنی است چون در واقع هر نقطه تولید شده در این فرایند جستجو شدنی است.
۲. اغلب میتوان تضمین نمود که اگر دنباله ای همگرا تولید کنند، نقطه ی حدی آن دنباله باید دست کم یک مینیمم مقید موضعی باشد. در واقع خصوصیات همگرایی سراسری این روش ها رضایتبخش است.
۳. بیشتر روش های اولیه متکی بر ساختار خاصی، همچون تحدب، نیستند. به همین خاطر این روش ها در مسائل غیرخطی کاربرد دارند.

در عین حال این روش ها عاری از مشکلات هم نیستند. برای نمونه دو مورد از معایب این روشها عبارتند از:

۱. نیاز به نقطه شروع شدنی دارند(که با حل مسأله فاز ۱ برنامه ریزی خطی بدست می آید).
۲. از نظر محاسباتی، قرار داشتن در ناخیه ی شدنی در روند اجرا، بسیار سخت است. این مشکل بویژه در مسائل با قیود غیرخطی بوجود می آید.

روش های جهت های شدنی

ایده روش جهت های شدنی این است که گام ها را در ناحیه شدنی بصورت زیر برداریم:

$$x_{k+1} = x_k + \alpha_k d_k$$

که d_k یک بردار جهت و α_k اسکالر نامنفی است و به گونه ای انتخاب میشود که تابع هدف با شرط شدنی بودن نقطه x_{k+1} و خط واصل x_k و x_{k+1} می نیمم شود. این روش تعمیمی از روش های کاهشی نامقید است و هرگام مرکب است از انتخاب یک جهت شدنی و یک جستجوی خطی مقید.

حال بردار d_k را یک جهت شدنی گوییم اگر

$$\exists\, \gamma > 0 : \quad \forall\ 0 \le \alpha \le \gamma \qquad x_k + \alpha d_k \quad feasible$$

همچنین بردار d_k را یک جهت شدنی بهبود دهنده گوییم اگر

$$\exists\, \gamma > 0 : \quad \forall\ 0 \le \alpha \le \gamma \qquad x_k + \alpha d_k \quad feasible\,,\ f(x_k + \alpha d_k) < f(x_k)$$

روش زونتدیک (Zoutendijk Method)

یکی از پیشنهادهای اولیه برای یک روش جهت های شدنی، بکارگیری یک زیرمسأله برنامه ریزی خطی است. فرض کنید قیود ما همه خطی باشند و مسأله برنامه ریزی زیر را در نظر میگیریم:

$$\begin{aligned} & minimize \quad f(x) \\ * \qquad & s.t. \quad Ax \le b \\ & \qquad\ Qx = q \end{aligned}$$

حال الگوریتم زیر را برای این روش داریم:

- مرحله مقداردهی اولیه:

$$x_1 \ feasible : \ Ax_1 \le b\,,\ Qx_1 = q\,, \qquad k \leftarrow 1$$

- مرحله اصلی:

۱. x_k را داریم

۲. $A^T = (A_1^T, A_2^T),\ b^T = (b_1^T, b_2^T)$ را تجزیه های ماتریس A^Tو b^T میگیریم بطوری که در A_1 قیودی که بطور تساوی برقرار میشوند را در نظر میگیریم. پس داریم

$$A_1 x_k = b_1\ , A_2 x_k < b_2$$

۳. حال d_k را میخواهیم تعیین کنیم که پاسخ بهینه برای مسأله زیر است:

$$\begin{aligned} minimize \quad & \nabla f(x)^T d \\ s.t. \quad & A_1 d \le 0 \\ & Qd = 0 \end{aligned}$$
$$for\ j = 1,2,\dots,n \quad -1 \le d_j \le 1 \ \ (P_1)$$
$$(or\ d^T d \le 1\ (P_2) or \nabla f(x)^T d \ge -1\ (P_3))$$

۴. اگر $\nabla f(x)^T d = 0$ توقف کن و x_k نقطه KKTاست؛ در غیر این صورت به گام ۵ می رویم.

۵. α_k را جواب مسأله جستجوی خطی زیر میگیریم:

$$\begin{aligned} minimize \quad & f(x_k + \alpha d_k) \\ s.t. \quad & 0 \le \alpha \le \alpha_{max} \end{aligned}$$

که داریم:

$$\alpha_{max} = \begin{cases} min\left\{\dfrac{\widehat{b_\iota}}{\widehat{d_\iota}} :\ \widehat{d_\iota} > 0\right\}, & if\ \hat{d} \le 0 \\ \infty, & if\ \hat{d} \ge 0 \end{cases}$$

۶. حال قرار میدهیم

$$x_{k+1} = x_k + \alpha_k d_k$$

۷. قیود مؤثر در x_{k+1} را شناسایی میکنیم

۸. $A_1\ , A_2$ را بروزرسانی میکنیم و $k \leftarrow k+1$ قرار میدهیم و به گام ۱ میرویم.

لم ۱:

مسأله (*) را در نظر میگیریم. فرض کنید x یک نقطه شدنی باشد و $A_1 x = b_1,\ A_2 x < b_2$که $A^T = (A_1^T, A_2^T),\ b^T = (b_1^T, b_2^T)$. حال بردار نامنفی d جهت شدنی در xاست اگر و تنها اگر $Qd = 0$ و $A_1 d \le 0$. همچنین اگر $\nabla f(x)^T d < 0$ آنگاه d یک جهت بهبود دهنده است.

لم ۲:

مسأله (*) را در نظر میگیریم. فرض کنید x یک نقطه شدنی باشد و $A_1x = b_1,\ A_2x < b_2$ که $A^T = (A_1^T, A_2^T),\ b^T = (b_1^T, b_2^T)$. حال برای هر $i = 1,2,3$، x نقطه KKT است اگر وتنها اگر مقدار بهینه تابع هدف مسائل P_i برابر 0 باشد.

پس دو لم فوق، اگر در گام ۴ الگوریتم مقدار بهینه تابع هدف، منفی باشد یک جهت شدنی بهبود دهنده داریم و از طرفی اگر برابر ۰ باشد، نقطه فعلی یک نقطه KKT است.

حال به بیان یک مثال میپردازیم:

مثال۱:

$$\begin{aligned} minimize\ \ & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ s.t.\ \ & x_1 + x_2 \le 2 \\ & x_1 + 5x_2 \le 5 \\ & -x_1 \le 0 \\ & -x_2 \le 0 \end{aligned}$$

میخواهیم مسأله را به روش زونتدیک با نقطه شروع $x_1 = (0,0)^T$ حل کنیم.

$$\rightarrow \nabla f(x) = (4x_1 - 2x_2 - 4,\ 4x_2 - 2x_1 - 6)^T$$

در هر تکرار پاسخ مسأله گام ۴ را که یک جهت جستجو را میدهد بدست اورده و سپس جستجوی خطی در این جهت انجام میدهیم. نهایتا پس از ۳ تکرار به پاسخ بهینه میرسیم.

			Search Direction				*Line Search*		
Iter. k	x_k	$f(x_k)$	$\nabla f(x_k)$	I	d_k	$\nabla f(x_k)^T d_k$	α_{max}	α_k	x_{k+1}
1	$(0,0)$	0	$(-4,-6)$	$\{3,4\}$	$(1,1)$	-10	$\frac{5}{6}$	$\frac{5}{6}$	$(\frac{5}{6},\frac{5}{6})$
2	$(\frac{5}{6},\frac{5}{6})$	-6.94	$(-\frac{7}{3},-\frac{13}{3})$	$\{2\}$	$(1,-\frac{1}{5})$	$-\frac{22}{15}$	$\frac{5}{12}$	$\frac{55}{186}$	$(\frac{35}{31},\frac{24}{31})$
3	$(\frac{35}{31},\frac{24}{31})$	-7.16	$(-\frac{32}{31},-\frac{160}{31})$	$\{2\}$	$(1,-\frac{1}{5})$	0			

محاسبات روش زونتدیک برای مثال۱.

حال مسأله زیر را در نظر میگیریم که ناحیه شدنی توسط قیود نامساوی که لزوما خطی نیستند تعریف شده است:

$$** \qquad \begin{aligned} &minimize \quad f(x) \\ &s.t. \quad g_i(x) \le 0 \quad for\ i = 1,2,\ \dots, m \end{aligned}$$

حال میخواهیم به بیان الگوریتم در این حالت بپردازیم اما قبل از آن دو قضیه زیر را بیان میکنیم.

قضیه۱: مسأله (**) را در نظر بگیرید. فرض کنید x یک جواب شدنی باشد و I مجموعه قیود موثر باشد که $I = \{i:\ g_i(x) = 0\}$. همچنین فرض کنید f و g_i برای $i \in I$مشتق پذیرند در x و هر g_i برای $i \notin I$ در xپیوسته است. اگر $\nabla f(x)^T d < 0$و $\nabla g_i{}^T d < 0$ برای $i \in I$ آنگاه dیک جهت شدنی بهبود دهنده است.

قضیه۲: مسأله (**) را در نظر بگیرید. فرض کنید x یک جواب شدنی باشد و I مجموعه قیود موثر باشد که $I = \{i:\ g_i(x) = 0\}$. مسأله یافتن جهت زیر را در نظر بگیرید:

$$\begin{aligned} &minimize \quad z \\ &s.t. \quad \nabla f(x)^T d - z \le 0 \\ &\qquad \nabla g_i(x)^T d - z \le 0 \ \ for\ i \in I \\ &\qquad -1 \le d_j \le 1 \ \ for\ j = 1,2,\dots,m \end{aligned}$$

حال xیک نقطه $\underline{Fritz\ John}$است اگر و تنها اگر مقدار بهینه تابع هدف در مسأله فوق برابر 0 باشد.

حال به بیان الگوریتم زونتدیک برای مسائل با قیود نامساوی غیرخطی میپردازیم:

- مقداردهی اولیه: انتخاب نقطه شدنی اولیه x_1 و سپس قرار میدهیم $k \leftarrow 1$ و به مرحله اصل می رویم.
- مرحله اصلی:

۱. x_k را داریم

۲. قرار میدهیم$I = \{i:\ g_i(x) = 0\}$

۳. (z_k, d_k) پاسخ بهینه برای مسأله زیر است

$$\begin{aligned} &minimize \quad z \\ &s.t. \quad \nabla f(x)^T d - z \le 0 \\ &\qquad \nabla g_i(x)^T d - z \le 0 \ \ for\ i \in I \\ &\qquad -1 \le d_j \le 1 \ \ for\ j = 1,2,\dots,m \end{aligned}$$

۴. اگر $z_k = 0$ توقف کن؛ x_k یک نقطه $Fritz\ John$ است. در غیر اینصورت اگر $z_k < 0$ به گام ۵ میرویم.

۵. α_k را جواب مسأله جستجوی خطی زیر میگیریم:

$$\begin{aligned} &minimize \ \ f(x_k + \alpha d_k) \\ &s.t. \ \ 0 \le \alpha \le \alpha_{max} \end{aligned}$$

که داریم:

$$\alpha_{max} = \sup\{\alpha: g_i(x_k + \alpha d_k) \leq 0 \ \ \forall i\}$$

۶. قرار می دهیم $x_{k+1} = x_k + \alpha_k d_k$

۷. حال قرار میدهیم $k \leftarrow k+1$ و به گام ۱ میرویم.

حال به بیان مثالی با قیود غیر خطی میپردازیم و با روش زنتدیک آن را حل میکنیم:

مثال۲:

$$\begin{aligned} minimize \ \ & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ s.t. \ \ & 2x_1^2 - x_2 \leq 0 \\ & x_1 + 5x_2 \leq 5 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0 \end{aligned}$$

نقطه شروع را $x_1 = (0.00, 0.75)^T$ میگیریم.

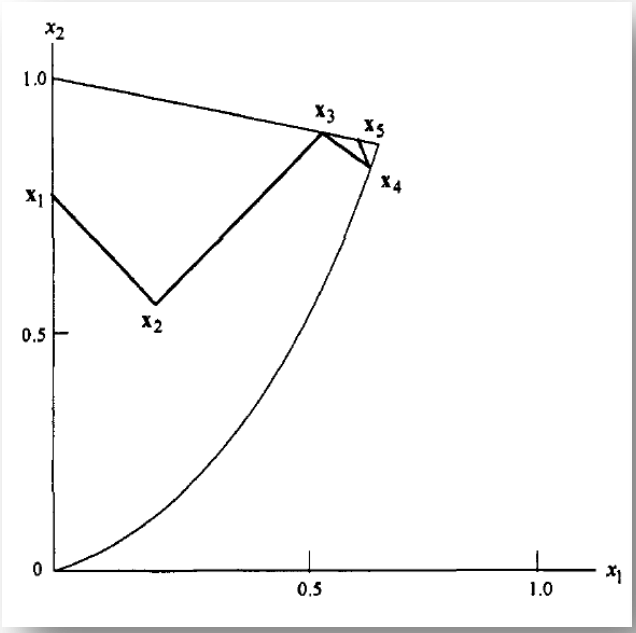
$$\rightarrow \nabla f(x) = (4x_1 - 2x_2 - 4, \ 4x_2 - 2x_1 - 6)^T$$

نهایتا با محاسبات ۴ تکرار از روش را بصورت جدول صفحه بعد داریم. مشکلی که وجود دارد این است که برای رسیدن به جواب دار نوسان شدیدی میشویم در این حالت.

			Line Search					
Iter. k	x_k	$f(x_k)$	$\nabla f(x_k)$	d_k	z_k	α_{max}	α_k	x_{k+1}
1	(0.00, 0.75)	−3.3750	(−5.50, −3.00)	(1.0000, −1.0000)	−1.000	0.4140	0.2083	(0.2083,0.5417)
2	(0.2083, 0.5477)	−3.6354	(−4.25, −4.25)	(1.0000, 1.000)	−8.500	0.3472	0.3472	(0.5555,0.8889)
3	(0.5555,0.8889)	−6.3455	(−3.5558, −3.5554)	(1.0000, −0.5325)	−1.663	0.09245	0.09245	(0.6479,0.8397)
4	(0.6479,0.8397)	−6.4681	(−3.0878, −3.9370)	(−0.5171,1.0000)	−2.340	0.343	0.0343	(0.6302,0.8740)

محاسبات روش زونتدیک برای مثال۲.

در شکل زیر میبینیم که پدیده ی زیگزاگ(zigzagging) رخ داده است بدلیل تقریبات مرتبه اولی که در این روش داریم که باعث میشود الگوریتم کند شود یا حتی به نقطه ای غیر از مینیمم موضعی یا سراسری همگرا شود.



روش زونتدیک برای مثال۲.

در واقع روش جهت های شدنی دو **نارسایی** عمده دارد که باعث میشود در اکثر موارد نیاز به ترمیم و تعدیل داشته باشد:

۱. در مسائل کلی ممکن است هیچ جهت شدنی وود نداشته باشد. برای مثال اگر مسأله ای دارای قیود تساوی غیرخطی باشد، ممکن است با وضعیتی که در شکل زیر میبینیم مواجه شویم که همانطور که میبینیم هی خط مستقیمی از x_k دارای قطعه شدنی نیست.

عدم وجود جهت شدنی

راه حل هایی برای این مشکل وجود دارد؛ مثلا میتوان با مجاز دانستن فاصله ی مختصر نقاط از رویه ی

قیدی، در التزام به شدنی بودن تخفیف دهیم و مثلا راحت تر بگیریم یا میتوان مفهوم حرکت در امتداد منحنی ها را بجای خطوط مستقیم وارد کار کرد(بجای تقریب مرتبه اول)

۲. بیشتر روش های جهت های شدنی در شکل ساده خودشان همگرای سراسری نیستند و در معرض پدیده ی ازدحام یا زیگزاگ هستند که در آن دنباله نقاط تولید شده در فرایند به نقطه ای همگرا میشود که حتی یک نقطه مینیمم موضعی مقید نیست. میتوان این پدیده را به دلیل بسته نبودن نگاشت الگوریتمی توجیه کرد.

این که از همین روش، روش هایی تولید کنیم که بسته باشند امکان پذیر است و در نتیجه در معرض ازدحام نیز نیستند اما روش ها تاحدی پیچیده میشوند و راهکار ساده تر برای برخورد با قیود نامساوی، بکاریگیری روش مجموعه مؤثر میباشد.

روش مجموعه مؤثر

ایده اساسی روش، افراز کردن قیود نامساوی به دو گروه است: آنهایی که مؤثر در نظر گرفته میشوند / آنهایی که نامؤثر در نظر گرفته میشوند

که ما از قیود نامؤثر اساسا صرف نظر میکنیم.

حال مسأله مقید زیر را داریم:

$$minimize\ f(x)$$
$$s.t\ \ g(x) \leq 0$$

که برای سادگی فقط قیود نامساوی را در نظر گرفته ایم. شرایط لازم برای مسأه عبارتند از:

$$\nabla f(x) + \lambda^T \nabla g(x) = 0$$
$$g(x) \leq 0$$
$$\lambda^T g(x) = 0$$
$$\lambda \geq 0$$

میتوان این شرایط را بصورت ساده تری برجسب مجموعه قیود مؤثر بیان کرد. اگر گیریم

$$A = \{i:\ g_i(x^*) = 0\}$$

در این صورت شرایط لازم بصورت زیر در می آیند:

$$\nabla f(x) + \sum_{i \in A} \lambda_i \nabla g_i(x) = 0$$
$$g_i(x) = 0 \;\; i \in A$$
$$g_i(x) < 0 \;\; i \notin A$$
$$\lambda_i \geq 0 \;\; i \in A$$
$$\lambda_i = 0 \;\; i \notin A$$

واضح است که اگر مجموعه مؤثر مشخص باشد، مسأله اصلی را میتوان به مسأله ای متناظر تبدیل کرد که فقط شامل قیود تساوی باشد. اگر این مسأله ی جدید حل شود و بقیه قیود نیز برقرار باشند و ضرایب لاگرانژ نامنفی بدست آیند حواب بدست آمده صحیح است.

ایده روش مجموعه مؤثر مبتنی بر تعریف مجموعه ای قیود به نام مجموعه کاری که در واقع مجموعه قیود در هر تکرار یا هر مرحله از الگوریتم است، میباشد. پس:

مجموعه کاری: زیرمجموعه ای از قیود که در نقطه جاری مؤثرند و بنابراین نقطه جاری برای مجموعه کاری، شدنی است.

رویه کاری: به رویه ای که توسط مجموعه کاری مشخص میشود، رویه کاری گوییم.

حال الگوریتم از اینجا با حرکت بر رویه ای که بوسیله مجموعه کاری قیود، تعریف میشود به سوی نقطه بهتر پیش میرود. در نقطه جدید ممکن است مجموعه کاری تغییر یابد. پس هر روش مجموعه مؤثر شامل گام های زیر است:

۱. تعیین یک مجموعه کاری فعلی که زیرمجموعه ای است متشکل از قیود مؤثر فعلی.
۲. حرکت بر رویه ای که بوسیله مجموعه کاری مشخص میشود به سوی نقطه ای بهتر.

حال روش های مختلفی برای تعیین این حرکت بر رویه کاری وجود دارد که درباره انها صحبت میکنیم. جهت حرکت معمولا به وسیله ی تقریب های مرتبه اول یا دوم توابع در نقطه فعلی تعیین میشود. خواص همگرایی مجانبی برای این روش ها هم به روند حرکت بر رویه کاری وابتسه است چون در نزدیکی جواب، مجموعه کاری عموما همان مجموعه مؤثر صحیح است و فرایندمان بطور متوالی بر رویه تعیین شده بوسیله آن قیدها حرکت میکند و مثلا اگر بر رویه کاری خوب حرکت نگنیم ممکن است به جواب همگرایی خوبی نداشته باشیم.

حال فرض کنید که برای یک مجموعه کاری مفروض به نام W از حل مسأله زیر با قیود تساوی، نقطه ی x_W بدست آید.

$$minimize \; f(x)$$
$$s.t \;\; g_i(x) = 0 \;\; i \in W$$

و x_W در رابطه $g_i(x_W) < 0 \quad i \notin W$ صدق میکند.

این نقطه در شرایط لازم زیر صدق میکند:

$$\nabla f(x_W) + \sum_{i \in A} \lambda_i \nabla g_i(x_W) = 0$$

همچنین اگر $\forall i \in W \ \lambda_i \geq 0$ آنگاه نقطه x_W جواب موضعی برای مسأله اصلی است. در این میان اگر $\exists i \in W; \ \lambda_i < 0$ آنگاه میتوان با رهاکردن قید متناظرش یعنی قید i تابع هدف را کاهش داد. دلیلیش هم این است که اگر حساسیت ضرایب لاگرانژ را تحلیل کنیم میبینیم که یک کاهش کوچک در مقدار یک قید از 0 به $-c$ ، باعث تغییر تابع هدف به اندازه $\lambda_i c$ (که مقداری منفی است) میشود؛ پس پاسخ تابع هدف کاهش میابد و در نتیجه با حذف قید iام از مجموعه کاری میتوان جواب بهتری بدست آورد. در نتیجه میتوان گفت ضرایب لاگرانژ یک مسأله به عنوان یک شاخص برای تعیین قیودی که باید از مجموعه کاری حذف شوند، عمل میکنند.

نکته دیگر این که در روند مینیمم سازی $f(x)$بر رویه کاری لازم است که مقادیر قیود دیگر را نیز بررسی کنیم تا مطمئن شویم نقض نشوند. گاهی اوقات این اتفاق می افتد که جین حرکت بر رویه کاری با مرز قید جدیدی مواجه میشویم که در این صورت مناسب است که این قید را هم به مجموعه کاری اضافه کنیم و در نتیجه بر رویه ای با یک بعد کمتر قبلی پیش برویم.

پس با تلفیق این دو ایده میتوان یک **استراتژی کامل مجموعه مؤثر** برای حذف و اضافه کردن قیود بدست اورد:

- مجموعه مؤثر اولیه داده شده
- مینیمم سازی بر رویه کاری
- اگر با مرز قید جدیدی مواجه شدیم، آن را به مجموعه کاری اضافه میکنیم و در غیر این صورت به گام بعد می رویم
- یافتن نقطه ای که f را نسبت به مجموعه کاری قیود فعلی مینیمم می سازد

- تعیین ضرایب لاگرانژ مربوطه
- اگر همه ضرایب لاگرانژ نامنفی باشند، جواب موجود بهینه است؛ در غیر اینصورت قیود با ضرایب لاگرانژ منفی از مجموعه کاری حذف میشوند
- فرایند با مجموعه کاری جدید دوباره آغاز می شود و f در تکرار بعدی اکیدا کاهش میابد.

حال یک قضیه مطرح میکنیم که نشان میدهد روش هایی که نین استراتژی را پیش میگیرند به جواب بهینه همگرا میشوند.

قضیه مجموعه مؤثر: فرض کنید به ازای هر زیرمجموعه W از اندیس های قیود، مسأله مقید

$$minimize\ \ f(x)$$
$$s.t.\ \ g_i(x) = 0\ \ i \in W$$

خوش تعریف و دارای یک ناتباهیده است. در این صورت دنباله نقاط تولید شده با استراتژی مجموعه مؤثر بنیادی به نقطه KKTهمگرا می شود.

اما مشکلات این روش عبارتند از:

- باید چندین مسأله با مجموعه مؤثرهای نادرست حل شوند.
- جواب های مسأله های میانی باید در حالت کلی، می نیمم سراسری دقیق باشند.
- برای روش هایی که با یک سری تغییرات از این روش حاصل می شوند، همگرایی را نمیتوان تضمین کرد و با پدیده ازدحام یا زیگراگ روبرو می شویم که باعث می شود مجموعه کاری نامتناهی بار تغییر یابد. اما در عمل این پدیده به ندرت رخ میدهد و این استراتژی با تعدیلهای مختلف کاراست و الگوریتم های مختلفی برای جزء اصلی روش مجموعه مؤثر که الگوریتم برای مینیمم سازی بر رویه کاری است داریم.

روش تصویر گرادیان

یکی از روش ها، روش تصویر گرادیان(Rosen 1960) است که با الهام از روش معمولی تندترین کاهش برای مسائل غیرمقید توسعه یافته است. در اینجا برای تعیین جهت حرکت، منفی گرادیان بر برویه کاری تصویر میشود. حال روش را مبتنی بر یک استراتژی مجموعه مؤثر بیان میکنیم.

ایتدا حالتی که قیودخطی باشند را در نظر میگیریم.

$$\begin{aligned} &minimize\ \ f(x) \\ &s.t.\ \ a_i^T x \le b_i\ i \in I_1 \\ &s.t.\ \ a_i^T x = b_i\ i \in I_2 \end{aligned}$$

تعریف: ماتریس $P_{n\times n}$یک ماتریس تصویر (Projection) است اگر $P = P^T$و $PP = P$.

لم: فرض کنید P یک ماتریس $n \times n$، آنگاه داریم:

۱. اگر P یک ماتریس تصویر باشد، Pنیمه معین مثبت است.

۲. Pماتریس تصویر است اگر و تنها اگر $I - P$ ماتریس تصویر باشد.

۳. فرض کنید Pماتریس تصویر است و $Q = I - P$. آنگاه $L = \{Px: x \in R^n\}$ و $L^\perp = \{Qx: x \in R^n\}$ زیرفضاهای خطی عمودبرهم می باشند. بعلاوه هر $x \in R^n$را بصورت منحصربفرد می توان بصورت $p + q$نمایش داد که $p \in L$ و $q \in L^\perp$.

حال به توضیح الگوریتم میپردازیم و سپس مراحل آن را دقیق بیان میکنیم.

در یک نقطه شدنی x که گفتیم از حل فاز ۱ بدست می آید، تعداد مشخص q قید مؤثر داریم. ابتدا مجموعه کاری $W(x)$ را مجموعه قیود مؤثر میگیریم. سپس در نقطه شدنی x به جستحوی یک بردار شدنی d با ضابطه $\nabla f(x)d < 0$ میپردازیم تا حرکت در جهت d باعث کاهش در تابع f شود.

ابتدا جهت های را که در $a_i^T d = 0\ \ i \in W(x)$ صدق میکنند در نظر میگیریم تا همه قیود کاری مؤثر بمانند. این کار باعث میشود بردار جهت d در زیرفضای مماس M که بوسیله مجموعه کاری قیود تعریف میشود واقع شود. بردار جهت خاصی که بکار میگیریم، تصویر منفی گرادیان بر این فضاست.

A_q را متشکل از سطر های گرادیان قیودکاری میگیریم که یک ماتریس $q \times n$ است با رتبه $q < n$.

سپس بردار d_k بصورت $d_k = -P_k g_k$ تعریف میشود که P_k ماتریس تصویر متناظر با زیرفضای M است:

$$P_k = [I - A_q^T(A_q A_q^T)^{-1} A_q]$$

بکارگیری ماتریس P_k روی هر بردار تصویرش بر M را بدست میدهد. میتوان دید اگر $d_k \neq 0$ آنگاه یک جهت کاهشی است. چون $g_k + d_k$ بر d_k عمود است، داریم:

$$g_k^T d_k = (g_k^T + d_k^T - d_k^T)d_k = -|d_k|^2$$

و در نتیجه با محاسبه d_k اگر مخالف 0 باشد یک جهت شدنی کاهشی بر رویه کاری است.

حال میخواهیم اندازه گام را انتخاب کنیم. با افزایش α از 0 ، نقطه $x + \alpha d$ در اغاز شدنی باقی میماند و مقدار متناظر f کاهش میابد. حال طول قطعه شدنی اط خط صادر شده از α را بدست می آوریم و fرا بر این قطعه مینیمم میکنیم. اگر مینیمم در نقطه انتهایی واقع شود، قید جدیدی مؤثر میشود و به مجموعه کاری اضافه میشود. حال اگر این امکان را که تصویر منفی گرادیان صفر باشد در نظر بگیریم داریم:

$$\nabla f(x) + \lambda_k^T A_q = 0$$

در نتیجه نقطه x_kدر شرایط لازم برای مینیمم بر رویه کاری صدق میکند . همچنین اگر λ_k ی متناظر با نامساوی های مؤثر همگی نامنفی باشند پس نقطه x_k نقطه ی KKT است و فرایند به پایان میرسد؛ اما اگر حداقل یکی از λ_k ها منفی باشند میوان قید متناظرش را رها کرد و سپس امکان حرکت در یک جهت جدید به سمت نقطه ای بهتر فراهم میشود.

حال یک تکرار از الگوریتم به صورت زیر است:

۱. M، زیرفضای قیود مؤثر را بدست آور و A_q, $W(x)$ را تشکیل بده

۲. محاسبه کن:

$$d = -P\nabla f(x)^T\ ,\ \ P = I - A_q^T(A_q A_q^T)^{-1} A_q$$

۳. اگر $d \neq 0$، α_1, α_2 را پیدا کن بطوریکه به ترتیب در موارد زیر صدق کند:

$$\alpha_1 = arg\ max\{\alpha:\ x + \alpha d\ \ feasible\}$$
$$\alpha_2 = arg\ min\{f(x + \alpha d): 0 \leq \alpha \leq \alpha_1\}$$

$x + \alpha_2 d$ را در x قراره بده و به (۱) برو.

۴. اگر $d = 0$ محاسبه کن: $\lambda = -(A_q A_q^T)^{-1} A_q \nabla f(x)^T$

i. اگر به ازای هر j مربوط به قیود نامساوی، $\lambda_j \geq 0$ توقف کن؛ xدر شرایط کیون-تاکر صدق میکند.

ii. در غیر اینصورت، سطر متناظر نامساوی با کوچکترین مولفه λرا از A_q حذف کن(و قید متناظر را از $W(x)$حذف کن) و به ۲ برو.

حال نکته این است که در هر نقطه جدید لازم نیست ماتریس تصویر را بطور کامل محاسبه کنیم و چون قیود مؤثر در مجموعه کاری هربار با حداکثر یک قید تغییر میکن، امکان محاسبه ماتریس تصویر از ماتریس قبلی، با یک بهنگام سازی وجود دارد و این باعث میشود محاسبات کاهش یابند.

مثال۳:

$$\begin{aligned} minimize\ \ & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ s.t.\ \ & x_1 + x_2 \le 2 \\ & x_1 + 5x_2 \le 5 \\ & -x_1 \le 0 \\ & -x_2 \le 0 \end{aligned}$$

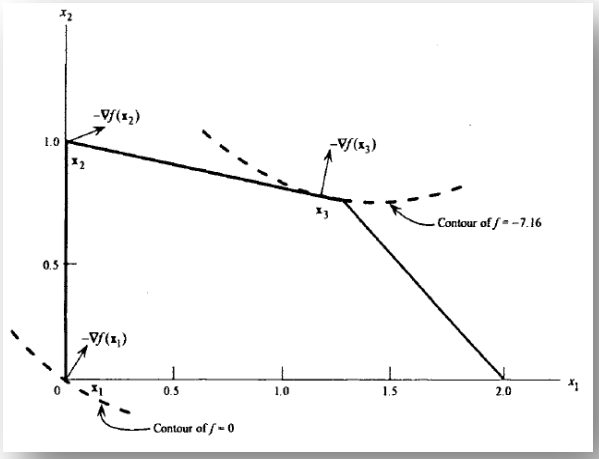
نقطه شروع را $x_1 = (0,0)^T$ میگیریم.

$$\rightarrow \nabla f(x) = (4x_1 - 2x_2 - 4,\ 4x_2 - 2x_1 - 6)^T$$

پس از ۳ تکرار به پاسخ رسیده ایم و چون تابع اکیدا محدب است پس نقطه بدست امده جواب سراسری است.

Iter.			*Search Direction*						*Line Search*		
k	x_k	$f(x_k)$	$\nabla f(x_k)$	I	A_1	P	d_k	u	α_{max}	α_k	x_{k+1}
1	$(0,0)$	0	$(-4,-6)$	$\{3,4\}$ $\{3\}$	$\begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}$ $[-1,0]$	$\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$ $\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}$	$(0,0)$ $(0,6)$	$(-4,-6)$ –	$-$ $\frac{1}{6}$	$-$ $\frac{1}{6}$	$-$ $(0,1)$
2	$(0,1)$	-4.00	$(-6,-2)$	$\{2,3\}$ $\{2\}$	$\begin{bmatrix}1 & 5\\ -1 & 0\end{bmatrix}$ $[1,5]$	$\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$ $\begin{bmatrix}\frac{25}{26} & -\frac{5}{26}\\ -\frac{5}{26} & \frac{1}{26}\end{bmatrix}$	$(0,0)$ $(\frac{70}{13},-\frac{14}{13})$	$(\frac{2}{5},-\frac{28}{5})$ –	$-$ $\frac{1}{4}$	$-$ $\frac{7}{31}$	$-$ $(\frac{35}{31},\frac{24}{31})$
3	$(\frac{35}{31},\frac{24}{31})$	-7.16	$(-\frac{32}{31},-\frac{160}{31})$	$\{2\}$	$[1,5]$	$\begin{bmatrix}\frac{25}{26} & -\frac{5}{26}\\ -\frac{5}{26} & \frac{1}{26}\end{bmatrix}$	$(0,0)$	$(\frac{32}{31})$	–	–	–

محاسبات روش تصویر گرادیان برای مثال۳.



روش تصویر گرادیان برای مثال۳.

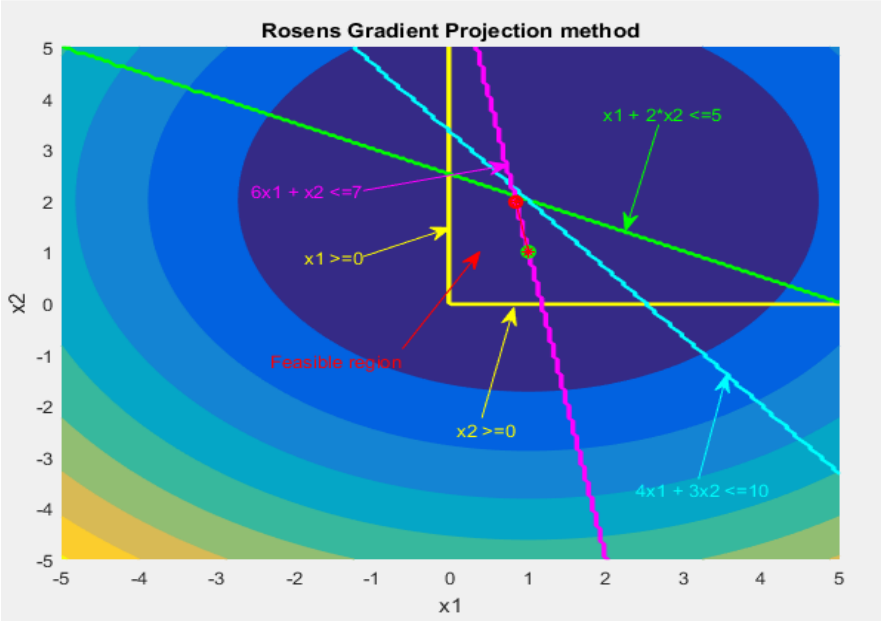
مثال۴:

$$minimize\ f(x) = (x_1 - 1)^2 + (x_2 - 2)^2 - 4$$
$$x_1 + 2x_2 \leq 5$$
$$4x_1 + 3x_2 \leq 6$$
$$6x_1 + x_2 \leq 7$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

حال نتایج حاصل از روش تصویر گرادیان بصورت زیر است:

Initial Objective Function Value: -3
Minimum succesfully obtained...
Number of Iterations for Convergence: 2
Point of Minima: [8.378378e-01,1.972973e+00]
Objective Function Minimum Value: -3.972973e+00

Iterations	X_coordinate	Y_coordinate	Objective_value
1	1	1	-3
2	0.83784	1.973	-3.973

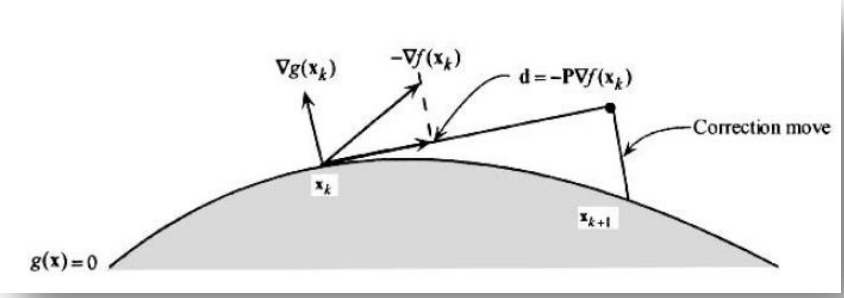


روش تصویر گرادیان برای مثال۴.

حال حالتی که در آن قیود غیرخطی باشند را در نظر میگیریم و به تحلیل روش کاهش گرادیان میپردازیم:

$$minimize\ f(x)$$
$$s.t\ h(x) = 0$$
$$g(x) \leq 0$$

ایده اساسی در تعمیم روش تصویر گرادیان در این مسائل این است که در یک نقطه شدنی x_k، قیود مؤثر تعیین شوند و منفی گرادیان به روی زیرفضای مماس بر رویه حاصل از این قیود تصویر شود. این بردار به شرط این که مخالف صفر باشد جهت را برای تکرار بعدی مشخص میکند اما خودش در حالت کلی یک جهت شدنی نمیدهد چون اینجا قیود غیرخطی اند و ممکن است رویه مربوطه دارای انحنا باشد، پس حرکت در جهت منفی گرادیان تصویر شده برای بدست اوردن نقطه بعدی همیشه امکان پذیر نیست مانند شکل زیر:



تصویر گرادیان در حضور قیدهای غیرخطی

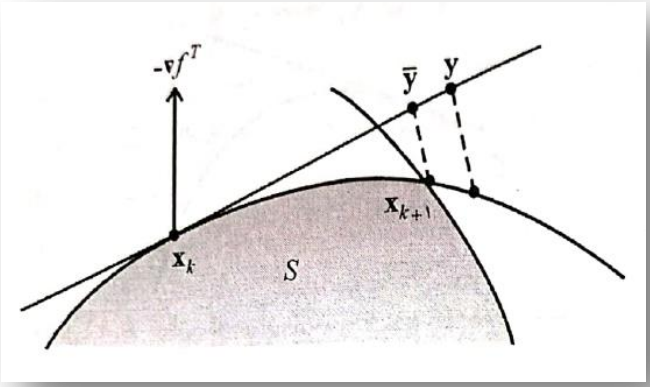
برای حل این مشکل ایده این است:

ابتدا در امتداد منفی گرادیان تصویر شده حرکتی انجام میشود و آن نقطه را y می نامیم. سپس حرکتی در امتداد عمود بر صفحه ی مماس ئر نقطه اولیه به سوی نقطه شدنی مجاور بر رویه کاری صورت میگیرد. سپس مقدار تابع هدف تعیین میشود. این حرکت برای y های گوناگون انجام میشود تا یک نقطه شدنی که یکی از معیارهای متعارف کاهش را برای بهتر شدن نسبت به نقطه اولیه برآورده سازد، بدست آید.

این روند یعنی خروج از ناحیه شدنی و بازگشت مجدد به آن باعث بروز مشکلاتی میشود که رفع آنها نیازمند یک رشته درون یابی و حل معادلات غیرخطی میشود. پس روش کلی مطلوب پیچیده میشود ولی اینجا هدف ما آشنیی با طبیعت کلی آن هاست.

مشکلات روش مطرح شده:

- هنگام بازگشت به نقطه ای که در قیود مؤثر قدیمی صدق میکند، ممکن است برخی نامساوی ها که ابتدا برقرار بودند، دیگر برقرار نباشند. برای حل این مشکل باید با استفاده از درون یابی یک نقطه جدید $\bar{y}$ در امتداد منفی گرادیان بدست آورد تا در هنگام بازگشت به قیود مؤثر، هیچ قید نامؤثر اولیه نقض نشود. یافتن $\bar{y}$ مناسب تاحدی سعی و خطاست. نهایتا نیز بازگشت به قیود مؤثر خود یک مسأله غیرخطی است که باید با یک روش تکراری حل شود. اما در یک تکرار نمیتوان دقیقا به رویه دست یافت و برقراری قیود در سراسر فرایند با مقدارخطای δ پذیرفته میشود.



درون یابی برای دستیابی به نقطه شدنی

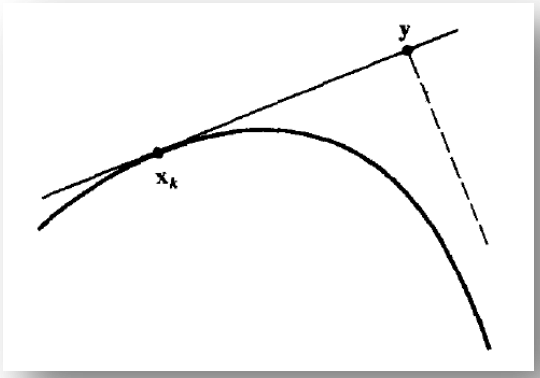
- محاسبه تصویرها در حالت غیرخطی مشکل تراست. برای سادگی نمادها، با ادغام نامساوی های مؤثر و تساوی ها در $h(x_k)$ ماتریس تصویر در x_k چنین میشود:

$$P_k = I - \nabla h(x_k)^T [\nabla h(x_k) \nabla h(x_k)^T]^{-1} \nabla h(x_k)$$

در نقطه x_k میتوان این ماتریس را بهنگام نمود تا اضافه شدن یا کاستن یک قید را مثل حالت خطی منظور کند ولی در هنگام حرکت از x_k به x_{k+1} ماتریس ∇h تغییر میکند و ماتریس تصویر جدید را نمیشود از قدیمی بدست آورد و در هر تکرار باید محاسبه شود.

حال گفتیم خصوصیت جدید این روش، مسأله بازگشت به ناحیه شدنی از نقاط خارج از این ناحیه است. ما در اینجا نوع **روش تکراری** را که نوعی متداول در برنامه ریزی غیرخطی است را میگوییم.

از نقطه ای در نزدیگی x_k در جهتی عمود بر صفحه مماس در x_k به رویه ی قیدی برمیگردیم. پس از نقطه ای مثل y در جستجوی نقطه ای به شکل $y + \nabla h(x_k)^T \alpha = y^*$ برمی آییم که $h(x^*) = 0$. حال ممکن است همانطور که در شکل زیر میبینیم جوابی وجود نداشته باشد اما برای y ای که به اندازه کافی نزدیک به x_k است، وجود دارد.



حالتی که بازگشت به رویه غیرممکن است

حال برای پیدا کردن تقریب اولیه مناسبی برای α و نتیجتا برای y^* معادله برا در x_k خطی میکنیم:

$$h(y + \nabla h(x_k)^T \alpha) \simeq h(y) + \nabla h(x_k) \nabla h(x_k)^T \alpha$$

تقریب فوق برای مقادیر کوچک $|\alpha|$ و $|y - x|$ مناسب است. سپس به نخستین تقریب رهنمون میشویم:

$$\rightarrow \alpha_1 = -[\nabla h(x_k) \nabla h(x_k)^T]^{-1} h(y)$$
$$\rightarrow y_1 = y - \nabla h(x_k)^T [\nabla h(x_k) \nabla h(x_k)^T]^{-1} h(y)$$

حال y_1 را بجای y قرار میدهیم و فرایند را تکرار میکنیم و دنباله بازگشتی زیر بدست می آید:

$$y_{j+1} = y_j - \nabla h(x_k)^T [\nabla h(x_k) \nabla h(x_k)^T]^{-1} h(y_j)$$

اگر نقطه شروع به اندازه کافی نزدیک به x_k و رویه قیدی باشد، به y^* همگرا میشود.

نکته: پیاده سازی موفقیت آمیز این روش همراه با رفع مشکلا زیادی که از لزوم به قرار گرفتن در ناحیه شدنی میشود نیازمند مهارت است ولی در کل کارایی این روش در حل مسائل برنامه ریزی غیرخطی معلوم شده است. ارزش واقعی این روش را با تحلیل آهنگ همگرایی آن میتوان درک کرد.

مثال۵:

$$minimize\ f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 - 2x_1 - 3x_4$$
$$s.t.\quad 2x_1 + x_2 + x_3 + 4x_4 - 7 \geq 0$$
$$x_1 + x_2 + x_3^2 + x_4 - 5.1 \geq 0$$
$$x_i \geq 0 \quad i = 1,2,3,4$$

$iter.$ k	x_k	$f(x_k)$
1	(2,2,1,0)	5.0
2	(2.036,1.822,1.126,0)	4.64

محاسبات روش تصویر گرادیان برای مثال۵.

آهنگ همگرایی روش تصویر گرادیان

کافی است الگوریتم ساده شده ی دیگری را که بطور مجانبی مثل تصویرگرادیان عمل میکند در نزدیکی جواب، تحلیل کنیم. نشان می دهیم آهنگ همگرایی اش به ساختار ویژه مقداری ماتریس هسی لاگرانژی محدود به زیرفضای مماس قیدی وابسته است.

به بیان الگوریتم کاهش ژئودزیک میپردازیم. برای سادگی، ابتدا مسأله را با قیود تساوی در نظر میگیریم:

$$minimize\ f(x)$$
$$s.t.\ \ h(x) = 0$$

تغییر دیدگاه: دیدن مسأله از دید حشره روی رویه قیدی ← از دید او مسأله نامقید و $n - m$ بعدی است← تندترین کاهش را پیشنهاد میدهد.

فرض کنیم مفهوم فاصله برای او مانند ما است آن گاه مسیر $x(t)$ بین دو نقطه ی x_1 , x_2 واقع بر رویه ی او که $\int_{x_1}^{x_2} |\dot{x}(t)| dt$ را مینیمم میکند یک خط مستقیم است در نظر او.

چنین منحنی که دارای طول کمان مینیمم بین دو نقطه ی مفروض است، ژئودزیک نامیده میشود.

حال به دیدگاه خودمان باز میگردیم و با الهام از دیدگاه حشره روش خود را بیان میکنیم:

- تصویر P $-\nabla f(x)^T$را بر روی صفحه مماس در x_k بدست آور
- ژئودزیک $x(t),\ t \geq 0$ از رویه قیدی را با ضوابط $x(0) = x_k$ و $x(0) = P$ بدست آور

- مقدار t_kمی نیمم کننده $f(x(t))$ را نسبت به $t \geq 0$ بدست آور و قرار ده $x_{k+1} = x(t_k)$

این روش صرفا نظری است و از نظر محاسباتی عملی نیست اما فلسفه اصلی روش تصویرگرادیان را دارد و آهنگ همگرایی مساوی دارند. حشره محدود شده به رویه، آهنگ همگرایی را به سادگی برحسب کوچکترین و بزرگترین مقدارویژه های هسی f بیان میکند:

$$(\frac{A-a}{A+a})^2$$

که a و A به ترتیب کوچکترین و بزرگترین مقدارویژه های L، هسی لاگرانژی محدود شده به زیرفضای مماس Mهستند. این آهنگ را آهنگ متعارف نیز می نامند.

در ادامه از جزئیات اثبات ها میگذریم(درکتاب fletcher,1987 جزئیات دقیق آمده است) و به بیان قضیه اصلی میپردازیم:

قضیه اصلی آهنگ همگرایی: فرض کنید x^* یک جواب موضعی برای مسأله با قیود تساوی است و $A\ ,\ a > 0$ به ترتیب بزرگترین و کوچکترین مقدارویژه های $L(x^*)$محدود شده به زیرفضای مماس $M(x^*)$ است. اگر $\{x_k\}$ دنباله ای تولید شده از روش کاهشی ژئودزیک باشد که همگرا به x^* است، آنگاه دنباله ی مقادیر هدف $\{f(x_k)\}$ بطور خطی با نسبتی نابزرگتر از

$$\left(\frac{A-a}{A+a}\right)^2$$

به $f(x^*)$ همگرا می شود.

گرادیان تقلیل یافته

روش گرادیان تقلیل یافته (Frank–Wolfe 1963)، از دیدگاه محاسباتی ارتباط نزدیکی با روش سیمپلکس برنامه ریزی خطی دارد که در آن متغیرهای مسأله به گروه های پایه ای و غیرپایه ای افراز میشوند. از دیدگاه نظری نیز این روش خیلی شبیه به روش تصویر گرادیان عمل میکند.

ابتدا حالت قیودخطی را در نظر میگیریم:

$$\begin{aligned} &minimize\ \ f(x) \\ &s.t.\ \ Ax = b \\ &x \geq 0 \end{aligned}$$

که در آن x متغیری n بعدی و b برداری m بعدی میباشد و ماتریس A نیز $m \times n$ است. در اینجا قیود به شکل استاندارد در برنامه ریزی خطی بیان شده اند و برای سهول در نمادگذاری هر متغیر را نامنفی میگیریم.

یک **فرض تاتبهگونی** را در نظرمیگیریم: هر مجموعه m ستونی از Aمستقل خطی است و هر جواب پایه ای برای قیود، m متغیر اکیدا مثبت دارد.

با این فرض هر جواب شدنی حداکثر $n - m$ متغیر با مقدار صفر دارد.

حال برای هر بردار x که در قیود صدق میکند متغیرها را به دو گروه افراز میکنیم: $x = (y, z)$ که y دارای بعد mو z دارای بعد $n - m$ میباشد. همه ی متغیرها در y اکیدا مثبت هستند و برای سهول m تای اول هستند. پس مسأله را میتوان به صورت زیر بازنویسی کرد:

$$\begin{aligned} &minimize\ \ f(y,z) \\ &s.t.\ \ By + Cz = b \\ &y \geq 0,\ z \geq 0 \end{aligned}$$

و در آن $A = [B, C]$. در واقع z را متغیرهای وابسته و y را متغیرهای مستقل میگیریم چون اگر zمشخص شود، y را بطور یکتا از معادله قیود میتوان بدست آورد. همچنین اگر در z تغییر کوچکی دهیم مثل Δz نسبت به مقدار اولیه که $z + \Delta z$ را نامنفی نگه میدارد، برای جواب معادله قیود منجر به جواب شدنی دیگری میشود چون y در آغاز اکیدا مثبت گرفته شد و $y + \Delta y$ برای مقدار کوچک Δy نیز مثبت خواهد بود.

پس با انتخاب یک Δz و حرکت دادن z روی خط $z + \alpha\Delta z$ که$\alpha \geq 0$ میتوانیم به نقطه شدنی دیگری حرکت کنیم. به همین ترتیب y در امتداد یک خط متناظر $y + \alpha\Delta y$ حرکت میکند.

اگر در این حرکت متغیری 0 شود یک قید نامساوی جدید موثر میشود. اگر متغیر مستقلی 0شود، یک جهت جدید Δz باید انتخاب شود. اگر متغیر وابسته ای 0 شود، افراز تغییر میکند و آن متغیر را مستقل و یکی از مستقل های اکیدا مثبت وابسته میشود.

این روش مبتنی بر ایده ای است که مسأله در هر مرحله تنها برحسب متغیرهای مستقل در نظر گرفته می شود. تابع هدف را میشود منحصرا تابعی از z گرفت. برای در نظر گرفتن قیود میتوان از روش تندترین کاهش الهام گرفت. گرادیان نسبت به متغیرهای مستقل(گرادیان تقلیل یافته) با محاسبه گرادیان $f(B^{-1}b - B^{-1}Cz,\ z)$ بدست میاید:

$$r^T = \nabla_x f(y,z) - \nabla_y f(y,z)B^{-1}C$$

نقطه ای چون z در شرایط لازم مرتبه اول برای بهینگی صدق میکند اگر وتنها اگر

$$r_i = 0 \quad \forall z_i > 0$$
$$r_i \geq 0 \quad \forall z_i = 0$$

در شکل مجموعه موثر روش گرادیان تقلیل یافته، بردار z در جهت گرادیان تقلیل یافته بر رویه کاری حرکت داده میشود. پس در هر تکرار جهتی بصورت زیر تعیین و کاهشی ایجاد میشود:

$$\Delta z_i = \begin{cases} -r_i & i \notin W(z) \\ 0 & i \in W(z) \end{cases}$$

هرگاه یک متغیر جدید 0 شود، بر مجموعه کاری افزاوده میشود. اگر پایه ای باشد، افراز جدیدی نیز شکل میگیرد. اگر نقطه ای پیدا شود که در آن $r_j = 0 \quad \forall j \notin W(z)$ اما به ازای برخی $r_j < 0 \quad j \in W(z)$ آنگاه j مطابق استراتژی مجموعه مؤثر از $W(z)$ حذف میشود. میتوان با حرکت کردن به دور از قید مؤثر مورد نظر، اگر باعث وضعیت بهتری میشود، از استراتژی خالص مجموعه مؤثر عدول کرد و منتظر نماند تا یک مینیمم دقیق بر رویه کاری پیدا شود. یک نوع از این روش این است که بردار z در جهت منفی گرادیان تقلیل یافته کلی حرکت داده شود، بجز اینکه مؤلفه های صفر مقدار z که در نتیجه اینکار منفی میشوند در صفر نگه داشته شوند.

یک تکرار از این روش بصورت زیر است:

- قرار ده: $\Delta z_i = \begin{cases} -r_i & r_i < 0 \ or \ z_i > 0 \\ 0 & o.w \end{cases}$
- اگر Δz برابر با ۰ است توقف کن؛ نقطه جاری یک جواب است. در غیر اینصورت محاسبه کن:

$$\Delta y = -B^{-1}C\Delta z$$

مقادیر $\alpha_1, \alpha_2, \alpha_3$ را به ترتیب زیر پیدا کن:

$$\alpha_1 = arg\ max\{\alpha : y + \alpha\Delta y \geq 0\}$$
$$\alpha_2 = arg\ max\{\alpha : z + \alpha\Delta z \geq 0\}$$

$$\alpha_3 = arg\ min\{f(x + \alpha\Delta x): 0 \le \alpha \le \alpha_1, 0 \le \alpha \le \alpha_2\}$$

قرار ده: $\overline{x} = x + \alpha_3 \Delta x$

- اگر $\alpha_3 < \alpha_1$ به ۱ برو. در غیر این صورت متغیر صفر شونده در مجموعه وابسته را مستقل کن و یک متغیر اکیدا مثبت در مجموعه مستقل را وابسته کن. $B,\ C$ را بهنگام کن.

در واقع الگوریتم مطرح شده برای پیدا کردن جهت در روش تقلیل یافته، بسته نیست زیرا حرکتی جزئی به دور از یک مرز یک قید نامساوی ممکن است منجر به تغییر ناگهانی جهت جستجو شود. پس میتوان گفت این روش در معرض ازدحام میباشد؛ ولی با یک تعدیل ساده یک نگاشت بسته و نتیجتا همگرایی سراسری حاصل میشود. با در نظر گرفتن دو تعدیل زیر میتوان هنگرایی سراسری الگوریتم گرادیان تقلیل یافته تعدیل شده را اثبات نمود:

- متغیرهای پایه ای در شروع هر تکرار همواره m تا از بزرگترین متغیرها (به ترتیب) احتیار می شوند.
- فرمول Δz بصورت زیر جایگزین می شود:

$$\Delta z_i = \begin{cases} -r_i & r_i \le 0 \\ -x_i r_i & r_i > 0 \end{cases}$$

مثال۶:

$$\begin{aligned} minimize\ & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ s.t.\quad & x_1 + x_2 + x_3 = 2 \\ & x_1 + 5x_2 + x_4 = 5 \\ & x_i \ge 0 \quad i = 1,2,3,4 \end{aligned}$$

نقطه شروع را $x_1 = (0,0,2,5)^T$ میگیریم.

$$\to \nabla f(x) = (4x_1 - 2x_2 - 4,\ 4x_2 - 2x_1 - 6,0,0)^T$$

نهایتا نتایج حاصل بصورت جدول زیر میباشد:

$iter.$ k	x_k	$f(x_k)$
1	(0,0,2,5)	0.0
2	$(\frac{10}{17}, \frac{15}{17}, \frac{9}{17}, 0)$	-6.436
3	$(\frac{35}{31}, \frac{24}{31}, \frac{3}{31}, 0)$	-7.16

محاسبات روش گرادیان تقلیل یافته برای مثال۶

حال به سراغ بیان الگوریتم برای قیود غیرخطی میرویم. مسأله برنامه ریزی غیرخطی زیر را در نظر میگیریم:

$$minimize\ f(x)$$
$$s.t.\ hx = 0$$
$$a \le x \le b$$

که در آن m قید داریم. در واقع مسأله برنامه ریزی غیرخطی کلی را میتوان همواره درصورت نیاز با معرفی متغیرهای لنگی $(slack)$ و درصورت لزوم با مجاز دانستن اینکه برخی مؤلفه های a و b مقادیر $-\infty$ یا $+\infty$ را اختیار میکنند به این شکل بیان کرد.

همانند حالت خطی، یک **فرض تاتبهگونی** بیان میکنیم: در هر نقطه x افرازی بصورت $x = (y, z)$ با خواص زیر وجود دارد:

۱. yدارای بعد mو z دارای بعد $n - m$ است
۲. اگر $a = (a_y, a_z)$و $b = (b_y, b_z)$ افرازهای متناظر $a,\ b$ باشند آنگاه $a_y < y < b_y$
۳. ماتریس $\nabla_y h(y, z)$، $m \times m$ در $x = (y, z)$ ناتکین است.

همچنین مجددا y و z را به ترتیب متغیرهای وابسته و مستقل میگیریم.

گرادیان تقلیل یافته نسبت به z:

$$r^T = \nabla_x f(y, z) + \lambda^T \nabla_z h(y, z)$$

λنیز در رابطه زیر صدق میکند:

$$\nabla_y f(y, z) + \lambda^T \nabla_y h(y, z) = 0$$

روش تقریبا مانند شیوه خطی است با این تفاوت که با وجود اینکه z مانند قبل در امتداد یک خط مستقیم حرکت میکند، بردار متغیرهای وابسته y باید بطور غیرخطی حرکت کنند تا در قیود تساوری بطور پیوسته صدق کنند.

محاسبات به صورت زیر انجام میشود:

- ابتدا با یک حرکت خطی در امتداد مماس بر رویه تعریف شده توسط $z \to z + \Delta z$, $y \to y + \Delta y$ با

$$\Delta y = -[\nabla_y h]^{-1} \nabla_z h \Delta z$$

انجام میگیرد.

- سپس یک رویه اصلاحی مانند روش تصویر گرادیان بکار میبریم
- مانند تصویر گرادیان حدودی برای پذیرش شدنی بودن باید معرفی شود
- یک الگوی تکرار ی برای بازگشت به رویه بکار گرفته میشود:

$$y_{j+1} = y_j - \left[\nabla_y h(x_k)\right]^{-1} h(y_j, W)$$

این روش همان مشکلات اساسی روش تصویر گرادیان را دارد ولی میتوان آن ها کمابیش برطرف کرد.

در عوض محاسبات در گرادیان تقلیل یافته نسبتا پیچیدگی کمتری دارد چون به جای محاسبه $[\nabla h(x_k)\nabla h(x_k)^T]^{-1}$ در هرگام, ماتریس $[\nabla_y h(y,z)]^{-1}$ بکار می رود.

آهنگ همگرایی روش گرادیان تقلیل یافته

همانطور که در روش قبل گفتیم، برای تحلیل اهنگ همگرایی کافی ست مسأله را با قیود تساوی در نظر بگیریم:

$$minimize\ f(x)$$
$$s.t.\ \ h(x) = 0$$

مجددا دیدگاه حشره را باز بکار میگیریم و یک روش ایده ال میدهیم(جزئیات روش در کتاب Fletcher آمده است). نهایتا قضیه زیر بدست می آید:

قضیه: فرض کنید x^* یک جواب موضعی برای مسأله باشد. فرض کنید روش گرادیان تقلیل یافته ایده آل دنباله $\{x_k\}$ را همگرا به x^* تولید کند و در سرتاسر دنباله افراز $x = (y,z)$ داریم. L را میاتریس هسی لاگرانژی در x^* بگیرید . ماتریس C رابصورت زیر تعریف میکنیم:

$$C = \begin{bmatrix} Y(Z^*) \\ - \\ I \end{bmatrix}$$
$$Y(z) = -[\nabla_y h(y,z)]^{-1}\nabla_z h(y,z)$$

در اینصورت دنباله مقادیر هدف $\{f(x_k)\}$ بطور خطی با نسبتی نابزرگتر از

$$\left(\frac{B-b}{B+b}\right)^2$$

که b و B به ترتیب کوچکترین و بزرگترین مقدارویژه های ماتریس $Q = C^T LC$ هستند به $f(x^*)$ همگرا میشود.

نکته: در قضیه فوق، Q در واقع تحدید $L(x^*)$ به زیرفضای مماس M میباشد ولی یک ماتریس متقارت معین مثبت از چپ و راست در آن ضرب شده است. پس آهنگ همگرایی در اینجا میتواند سریعتر یا کندتر از روش تصویر گرادیان باشد ولی بطور کلی اگر C خوش حالت باشد میتوان انتظار داشت که نسبت مقدار ویژه های روش تقلیل یافته از همان مرتبه اندازه روش تصویر گرادیان باشد.

مثال۷:

$$minimize \sum_{i=1}^{20}(20 - i + 0.5)y_i$$

$$s.t. \sum_{i=1}^{20} y_i = 0$$

$$\sum_{i=1}^{20}\sqrt{1 - y_i^2} = 16$$

جواب شدنی اولیه را بصورت زیر میگیریم:

$$y_i = \begin{cases} -0.6 & 1 \le i \le 10 \\ 0.6 & 11 \le i \le 20 \end{cases}$$

نهایتا نتایج بصورت زیر بدست میآیند:

Iteration	Value	Solution (½ of chain)
0	−60.00000	$y_1 = -.8148260$
10	−66.47610	$y_2 = -.7826505$
20	−66.52180	$y_3 = -.7429208$
30	−66.53595	$y_4 = -.6930959$
40	−66.54154	$y_5 = -.6310976$
50	−66.54537	$y_6 = -.5541078$
60	−66.54628	$y_7 = -.4597160$
69	−66.54659	$y_8 = -.3468334$
70	−66.54659	$y_9 = -.2169879$
		$y_{10} = -.07492541$
Lagrange multipliers −9.993817, −6.763148		

محاسبات روش گرادیان کاهش یافته برای مثال۷.

مقایسه روش های گرادیان تقلیل یافته و تصویر گرادیان

در اینجا به مقایسه دو روش گرادیان تقلیل یافته و تصویر گرادیان در مثال هایی میپردازیم وعملکرد آن ها را بررسی میکنیم.

مثال۸:

مسأله زیر را در نظر میگیریم که مسأله کمترین مربعات با متغیرهای کراندار است:

$$minimize \ \frac{1}{2}\|Ax-b\|_2^2$$
$$s.t. \quad 0 \le x_i \le 1$$

که در آن داریم:

$$A \to m \times n \ (m = 50, \ n = 100)$$
$$x_0 \to random \ \& \ feasible$$

و ماتریس b را هم به این صورت تعریف میکنیم که ابتدا یک جواب رندم برای مسأله در نظر میگیریم و سپس داریم:

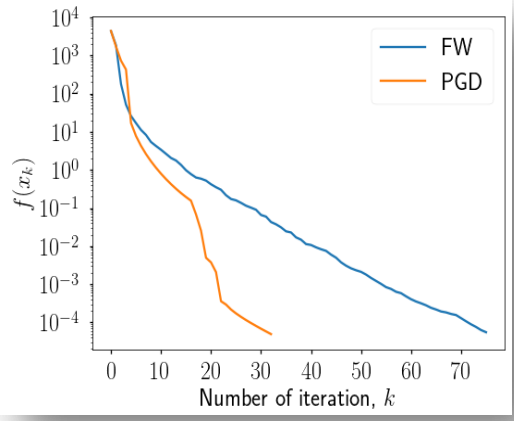
```
x_true = np.random.rand(n)
b = A.dot(x_true) + 0.01 * np.random.randn(m)
```

نقطه شروع را یک نقطه شدنی تصادفی در نظر میگیریم و نهایتا دو تابع تعریف میکنیم که هرکدام روش های تصویرگرادیان و گرادیان تقلیل یافته را انجام میدهند و روی مثال فوق اعمال میکنیم و داریم:

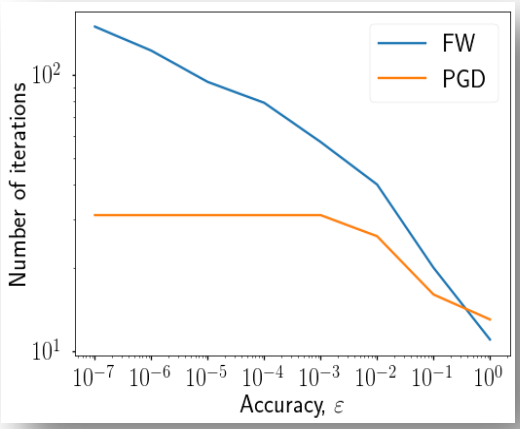
Resuts

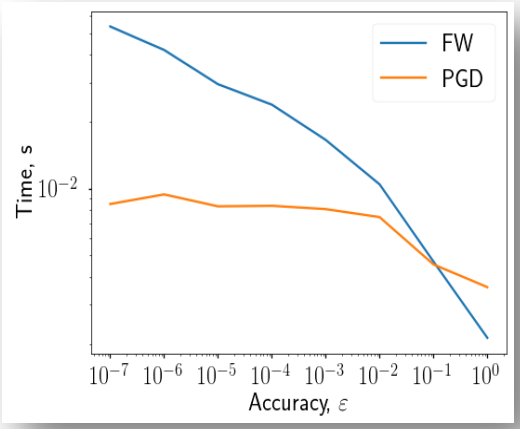
FW method	PG method
Convergence in 75 iterations Function value = 5.503265417124177e-05 Time = 31 ms ± 2.87 ms per loop	Convergence in 32 iterations Function value = 4.91178906574127 15e-05 Time = 8.89 ms ± 458 µs per loop

همچنین نمودار همگرایی دو روش نیز به صورت زیر است که همانطور که میبینیم در این مثال روش تصویر گرادیان عملکرد بهتری دارد و در تعداد تکرار کمتری به جواب همگرا شده است:



پس از آن به بررسی دقت دو روش میپردازیم و هم تعداد تکرار ها و هم زمان را بر حسب دقت بررسی میکنیم و نتایج بصورت زیر است که همانطور که میبینیم در هر صورت روش تصویر گرادیان دقت بهتری را در زمان و تعداد تکرار کمتر به ما میدهد:





مثال ۹:

حال مثال دیگری که میتوان در نظر گرفت، همان مسأله کمترین مربعات اما اینبار در سادک (سیمپلکس) است:

$$minimize \ \frac{1}{2}\|Ax-b\|_2^2$$
$$s.t. \quad \|x\|_1 \leq 1$$
$$x_i \geq 0$$

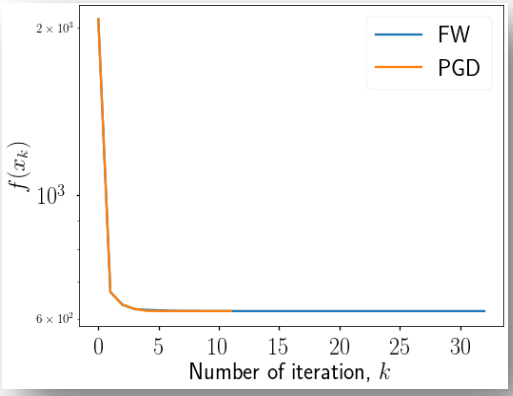
باز هم به همان روش مثال قبل ماتریس A و بردار b را تولید میکنیم و نقطه شروع را نیز شدنی تصادفی میگیریم و نتایج برای دو روش بصورت زیر است:

Resuts

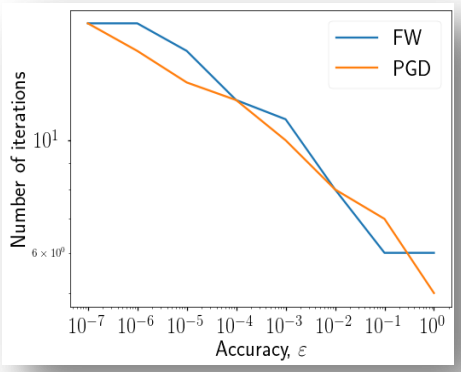
PG method
Convergence in 11 iterations
Function value = 619.1425429905358
Time = 6.58 ms ± 222 µs per loop

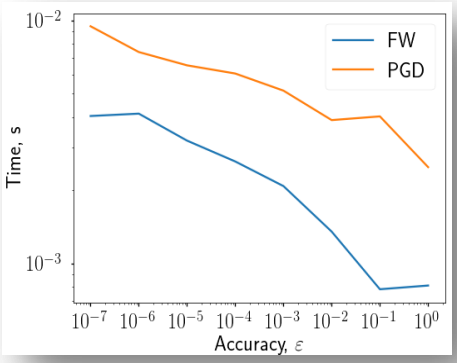
FW method
Convergence in 32 iterations
Function value = 619.1425573630355
Time = 3.33 ms ± 327 µs per loop

در اینجا روش تصویر گرادیان در تعداد تکرار کمتر اما در زمان بیشتری به جواب میرسد و نمودار همگرایی بصورت زیر است:



همچنین زمان و تعداد تکرار را برحسب دقت های لازم مورد بررسی دقت میدهیم و نتایج بصورت زیر است که از نظر زمانی روش گرادیان تقلیل یافته عملکرد بهتری دارد و در تعداد تکرار نیز تاحدی شبیه به هم هستند اما روش تصویر گرادیان کمی بهتر عمل میکند:





نتیجه گیری

نهایتا از تحلیل ها و بررسی های انجام شده، نتایج زیر حاصل میشوند:

- ✓ عملی ترین روش های اولیه، روش تصویر گرادیان و روش گرادیان تقلیل یافته هستند.
- ✓ این دو روش بصورت بنیادی کاربردی از روش تندترین کاهش بر رویه تعریف شده توسط قیود هستند.
- ✓ می توان انتظار داشت آهنگ همگرایی این دو روش تقریبا مساوی باشد. این آهنگ توسط ویژه مقدارهای هسی لاگرانژی محدود به زیرفضای مماس بر قیود مؤثر تعیین میشود.
- ✓ بنظر می رسد گرادیان تقلیل یافته روش بهتری باشد؛ به آسانی ترمیم میشود تا از ازدحام اجتناب شود و در هر تکرار محاسبات کمتری نیاز دارد و در زمان کمتری از روش تصویر گرادیان احتمالا همگرا می شود.

استفاده از نرم افزار

کدهای python دو روش تصویر گرادیان و گرادیان کاهش یافته پیوست شده است که بصورت زیر هستند.

کد پایتون دو روش گرادیان کاهش یافته و تصویر گرادیان بصورت تابع هایی نوشته شده است و نهایتا تابع دیگری تعریف شده که با گرفتن اسم روش و ورودی های دیگر بصورت زیر، مسئله را حل میکند:

```
methods[m_name].solve(x0=x0, max_iter=max_iter, tol=tol, disp=1)
```

که در آن ورودی ها زیر مسأله هایی بصورت تابع برای یکی از روشهای کاهش گرادیان و گرادیان کاهش بافته میباشد و همچنین تعداد حداکثر تکرار ها و نقطه شروع و تولرانس خطا را نیز بعنوان ورودی میگیریم. ورودی های مثال ها نیز در بخش مربوط به مثال ها گفته شد. نهایتا خروجی این دو روش شامل تعداد تکرار ها تا رسیدن به پاسخ و مقدارتابع هدف نهایی و مقدار گرادیان میباشد. پس از آن نیز زمان اجرای هر لوپ الگوریتم را نیز چاپ میکنیم.

کدهای پایتون بصورت کامل در زیر آمده است:

کد مثال۸ (Python):

ابتدا کتابخانه های موردنیاز فراخوانی میشوند:

```
import numpy as np
import liboptpy.base_optimizer as base
import liboptpy.constr_solvers as cs
import liboptpy.step_size as ss
import matplotlib.pyplot as plt
%matplotlib inline
plt.rc("text", usetex=True)
fontsize = 24
figsize = (8, 6)
import seaborn as sns
sns.set_context("talk")
from tqdm import tqdm
```

صورتبندی یک مسأله بهینه سازی را انجام میدهیم:

```
def func(x, A, b):
    return 0.5 * np.linalg.norm(A.dot(x) - b)**2

f = lambda x: func(x, A, b)

def grad_f(x, A, b):
    grad = -A.T.dot(b)
    grad = grad + A.T.dot(A).dot(x)
    return grad
```

```
grad = lambda x: grad_f(x, A, b)
```

حال به تعریف تابع برای روش گرادیان کاهش یافته میپردازیم:

```
def linsolver(gradient):
    x = np.zeros(gradient.shape[0])
    pos_grad = gradient > 0
    neg_grad = gradient < 0
    x[pos_grad] = np.zeros(np.sum(pos_grad == True))
    x[neg_grad] = np.ones(np.sum(neg_grad == True))
    return x
```

در اینجا نیز تابع روش تصویر گرادیان تعریف شده است:

```
def projection(y):
    return np.clip(y, 0, 1)
```

حال به مقداردهی پارامترهای مسئله خود میپردازیم با توجه به مثال:

```
m = 50
n = 100
A = np.random.randn(m, n)
x_true = np.random.rand(n)
b = A.dot(x_true) + 0.01 * np.random.randn(m)
methods = {"FW": cs.FrankWolfe(f, grad, linsolver,
ss.Backtracking(rule_type="Armijo", rho=0.5, beta=0.1, init_alpha=1.)),
           "PGD": cs.ProjectedGD(f, grad, projection,
ss.Backtracking(rule_type="Armijo", rho=0.5, beta=0.1, init_alpha=1.))
          }
x0 = np.random.randn(n)
max_iter = 300
tol = 1e-5
```

نهایتا مسئله را حل میکنیم و پاسخ بدست می آید:

```
for m_name in methods:
    print("\t", m_name)
    x = methods[m_name].solve(x0=x0, max_iter=max_iter, tol=tol, disp=1)
```

همگرایی تابع هدف را بررسی میکنیم برحسب تعداد تکرارهای انجام روش و نمودار آن را رسم میکنیم:

```
plt.figure(figsize=figsize)
for m_name in methods:
    plt.semilogy([f(x) for x in methods[m_name].get_convergence()],
label=m_name)
plt.legend(fontsize=fontsize)
plt.xlabel("Number of iteration, $k$", fontsize=fontsize)
plt.ylabel(r"$f(x_k)$", fontsize=fontsize)
plt.xticks(fontsize=fontsize)
_ = plt.yticks(fontsize=fontsize)
```

حال زمانی که هر تکرار(لوپ) برنامه طول میکشد را چاپ میکنیم و زمان دو روش را مقایسه میکنیم:

```
for key in methods:
    print("\t {}".format(key))
    %timeit methods[key].solve(x0, max_iter, tol)
```

حال نمودار وابستگی زمان و تعداد تکرارها را نسبت به دقت مورد نیاز رسم میکنیم و مقایسه میکنیم:

```
eps = [10**(-i) for i in range(8)]
time_pg = np.zeros(len(eps))
```

```
time_cg = np.zeros(len(eps))
iter_pg = np.zeros(len(eps))
iter_cg = np.zeros(len(eps))
pg = cs.ProjectedGD(f, grad, projection)
cg = cs.FrankWolfe(f, grad, linsolver, ss.Backtracking(rule_type="Armijo",
rho=0.5, beta=0.1, init_alpha=1.))
for i, tol in tqdm(enumerate(eps)):
    res = %timeit -o -q pg.solve(x0=x0, tol=tol, max_iter=100000)
    time_pg[i] = res.average
    iter_pg[i] = len(pg.get_convergence())
    res = %timeit -o -q cg.solve(x0=x0, tol=tol, max_iter=100000)
    time_cg[i] = res.average
    iter_cg[i] = len(cg.get_convergence())
plt.figure(figsize=figsize)
plt.loglog(eps, time_cg, label="FW")
plt.loglog(eps, time_pg, label="PGD")
plt.legend(fontsize=fontsize)
plt.xticks(fontsize=fontsize)
plt.yticks(fontsize=fontsize)
plt.xlabel(r"Accuracy, $\varepsilon$", fontsize=fontsize)
plt.ylabel(r"Time, s", fontsize=fontsize)
plt.figure(figsize=figsize)
plt.loglog(eps, iter_cg, label="FW")
plt.loglog(eps, iter_pg, label="PGD")
plt.legend(fontsize=fontsize)
plt.xticks(fontsize=fontsize)
plt.yticks(fontsize=fontsize)
plt.xlabel(r"Accuracy, $\varepsilon$", fontsize=fontsize)
plt.ylabel(r"Number of iterations", fontsize=fontsize)
```

کد مثال۹ (Python):

```
def func(x, A, b):
    return 0.5 * np.linalg.norm(A.dot(x) - b)**2

f = lambda x: func(x, A, b)

def grad_f(x, A, b):
    grad = -A.T.dot(b)
    grad = grad + A.T.dot(A).dot(x)
    return grad

grad = lambda x: grad_f(x, A, b)
m = 50
n = 100
A = np.random.randn(m, n)
x_true = np.random.rand(n)
b = A.dot(x_true) + 0.01 * np.random.randn(m)
def linsolver(gradient):
    x = np.zeros(gradient.shape[0])
    idx_min = np.argmin(gradient)
    if gradient[idx_min] > 0:
```

```
        x[idx_min] = 0
    else:
        x[idx_min] = 1
    return x
def projection(y):
    x = y.copy()
    if np.all(x >= 0) and np.sum(x) <= 1:
        return x
    x = np.clip(x, 0, np.max(x))
    if np.sum(x) <= 1:
        return x
    n = x.shape[0]
    bget = False
    x.sort()
    x = x[::-1]
    temp_sum = 0
    t_hat = 0
    for i in range(n - 1):
        temp_sum += x[i]
        t_hat = (temp_sum - 1.0) / (i + 1)
        if t_hat >= x[i + 1]:
            bget = True
            break
    if not bget:
        t_hat = (temp_sum + x[n - 1] - 1.0) / n
    return np.maximum(y - t_hat, 0)
methods = {
           "FW": cs.FrankWolfe(f, grad, linsolver,
ss.Backtracking(rule_type="Armijo", rho=0.5, beta=0.1, init_alpha=1.)),
           "PGD": cs.ProjectedGD(f, grad, projection,
ss.Backtracking(rule_type="Armijo", rho=0.5, beta=0.1, init_alpha=1.))
         }
x0 = np.random.randn(n)
max_iter = 300
tol = 1e-5
for m_name in methods:
    print("\t", m_name)
    x = methods[m_name].solve(x0=x0, max_iter=max_iter, tol=tol, disp=1)
plt.figure(figsize=figsize)
for m_name in methods:
    plt.semilogy([f(x) for x in methods[m_name].get_convergence()],
label=m_name)
plt.legend(fontsize=fontsize)
plt.xlabel("Number of iteration, $k$", fontsize=fontsize)
plt.ylabel(r"$f(x_k)$", fontsize=fontsize)
plt.xticks(fontsize=fontsize)
_ = plt.yticks(fontsize=fontsize)
for key in methods:
    print("\t {}".format(key))
    %timeit methods[key].solve(x0, max_iter, tol)
eps = [10**(-i) for i in range(8)]
time_pg = np.zeros(len(eps))
time_cg = np.zeros(len(eps))
```

```
iter_pg = np.zeros(len(eps))
iter_cg = np.zeros(len(eps))
pg = cs.ProjectedGD(f, grad, projection)
cg = cs.FrankWolfe(f, grad, linsolver, ss.Backtracking(rule_type="Armijo",
rho=0.5, beta=0.1, init_alpha=1.))
for i, tol in tqdm(enumerate(eps)):
    res = %timeit -o -q pg.solve(x0=x0, tol=tol, max_iter=100000)
    time_pg[i] = res.average
    iter_pg[i] = len(pg.get_convergence())
    res = %timeit -o -q cg.solve(x0=x0, tol=tol, max_iter=100000)
    time_cg[i] = res.average
    iter_cg[i] = len(cg.get_convergence())
plt.figure(figsize=figsize)
plt.loglog(eps, time_cg, label="FW")
plt.loglog(eps, time_pg, label="PGD")
plt.legend(fontsize=fontsize)
plt.xticks(fontsize=fontsize)
plt.yticks(fontsize=fontsize)
plt.xlabel(r"Accuracy, $\varepsilon$", fontsize=fontsize)
plt.ylabel(r"Time, s", fontsize=fontsize)
plt.figure(figsize=figsize)
plt.loglog(eps, iter_cg, label="FW")
plt.loglog(eps, iter_pg, label="PGD")
plt.legend(fontsize=fontsize)
plt.xticks(fontsize=fontsize)
plt.yticks(fontsize=fontsize)
plt.xlabel(r"Accuracy, $\varepsilon$", fontsize=fontsize)
plt.ylabel(r"Number of iterations", fontsize=fontsize)
```

کد مثال۴(MATLAB):

```
clc
clear
format short
% Function Definition:
syms x1 x2;
%Objective function
f = (x1-1)^2 + (x2-2)^2 -4;
%Constrains :
g(1) = x1+2*x2-5;       % x1+2*x2<=5
g(2) = 4*x1 +3*x2-6;    % 4*x1 +3*x2<=6
g(3) = 6*x1+x2-7;       % 6*x1+x2-7
g(4) = -x1;             % x1>=0
g(5) = -x2;             % x2>=0
N = 5; %number of constraints
%tolerances
eps1 = 0.001;
conv = 1; %initialize the convergance criteria
%Step 1: Initial Guess (Choose Initial Guesses):
i = 1;
x_1(i) = 1;
x_2(i) = 1;
%Save the gradient of given objective function
```

```
Search_dir = -gradient(f);
%Objective function value at initial guess
fun_value = (subs(f,[x1,x2], [x_1(i),x_2(i)]));
while conv > eps1
    %matrix for constratint at initial guess
    const(1) = (subs(g(1),[x1,x2], [x_1(i),x_2(i)]));
    const(2) = (subs(g(2),[x1,x2], [x_1(i),x_2(i)]));
    const(3) = vpa(subs(g(3),[x1,x2], [x_1(i),x_2(i)]),4);
    const(4) = (subs(g(4),[x1,x2], [x_1(i),x_2(i)]));
    const(5) = (subs(g(5),[x1,x2], [x_1(i),x_2(i)]));

    %Checking the optimum point [x_1,x_2] will violate the constraint or not
    if max(const) < 0 %not violated
        %calculate the search diection since point satisfies constraint
        S = subs(Search_dir,[x1,x2], [x_1(1),x_2(1)]);
    else %if constraint violate
        for j = 1:N
            if const(j)==0
                %calculate the projection matrix of violated constaraint
                N = (subs(gradient(g(j)),[x1,x2], [x_1(i),x_2(i)]));
                P = eye(2)-(N*inv(N'*N)*N');
                %Since point violate the constraint so calculate again
                %serach direction
                S = -((P*double(subs(gradient(f),[x1,x2],
[x_1(i),x_2(i)]))));
                break;
            end
        end
    end
    %calculating the step length
    if norm(S)== 0
        %If search direction is zero
        lambda = -inv(N'*N)*N'*double(subs(gradient(f),[x1,x2],
[x_1(i),x_2(i)]));
        if lambda>0
            %lambda more than zero then will get our optimum point
            optima = [x_1(i),x_2(i)];
            optimum = double(subs(f,[x1,x2], [x_1(i),x_2(i)]));
            break;
        end
    else %if search direction isn't zero
        S = S/norm(S);
        syms lambda
        const(1) = vpa((subs(g(1),[x1,x2], [x_1(i)+ lambda*S(1),
x_2(i)+lambda*S(2)])));
        const(2) = vpa((subs(g(2),[x1,x2], [x_1(i)+ lambda*S(1),
x_2(i)+lambda*S(2)])));
        lam2 = vpa(solve(const(1)==0,lambda),5);
        const(3) = vpa((subs(g(3),[x1,x2], [x_1(i)+ lambda*S(1),
x_2(i)+lambda*S(2)])));
        lam3 = vpa(solve(const(3)==0,lambda),5);
        const(4) = vpa((subs(g(4),[x1,x2], [x_1(i)+ lambda*S(1),
x_2(i)+lambda*S(2)])));
        lam4 = vpa(solve(const(4)==0,lambda),5);
        const(5) = vpa((subs(g(5),[x1,x2], [x_1(i)+ lambda*S(1),
x_2(i)+lambda*S(2)])));
        lam5 = vpa(solve(const(5)==0,lambda),5);
```

```
        lam = max(lam2,max(lam3,max(lam4,lam5)));
        lambd = max(lam);
        %put the maximum value of point in objective function
        func_lambda = vpa(subs(f,[x1,x2], [x_1(i)+ lambda*S(1),x_2(i)+
lambda*S(2)]));
        %differentiate the objective function respect to step length
        dfunc_lambda = diff(func_lambda,lambda);
        dfunc_lambda_lambd = vpa(subs(dfunc_lambda,[lambda],[lambd]));
        if dfunc_lambda_lambd>0
            lambda = vpa(solve(dfunc_lambda==0),5);
        else
            lambda = lambd;
        end
    end
    %new point
    x_1(i+1) = x_1(i)+lambda*S(1);
    x_2(i+1) = x_2(i)+lambda*S(2);
    fun_value1 = vpa(subs(f,[x1,x2], [x_1(i),x_2(i)]),6);
    fun_value2 = vpa(subs(f,[x1,x2], [x_1(i+1),x_2(i+1)]),6);
    %convergance criteria
    conv = (abs(fun_value1)-abs(fun_value2))/(abs(fun_value1));
    %increase the iteration number
    i = i+1
end
%% draw table in column view
Iter = 1:i;
X_coordinate = x_1';
Y_coordinate = x_2';
Iterations = Iter';
for i=1:length(X_coordinate)
    Objective_value(i) = double(subs(f,[x1,x2], [x_1(i),x_2(i)]));
end
Objective_value = Objective_value';
%table view
T = table(Iterations,X_coordinate,Y_coordinate, Objective_value);
%Output
fprintf('Initial Objective Function Value: %d\n\n',double(subs(f,[x1,x2],
[x_1(1),x_2(1)])));
if (conv < eps1)
    fprintf('Minimum succesfully obtained...\n\n');
end
fprintf('Number of Iterations for Convergence: %d\n\n', i);
fprintf('Point of Minima: [%d,%d]\n\n', x_1(i), x_2(i));
fprintf('Objective Function Minimum Value: %d\n\n', double(subs(f,[x1,x2],
[x_1(i),x_2(i)])));
%Plots
hold on;
%Contour Drawing
fcontour(f,[-5,5,-5,5],'MeshDensity',500, 'Fill', 'On');
[x1,x2] = meshgrid(-5:0.05:5);
Z1 =(x1>=0 & x2>=0 );
Z2 = (x1 +2*x2 <= 5) ;
Z3= (4*x1 +3*x2 <= 10);
Z4= (6*x1 +x2 <= 7);
contour(x1,x2,Z1,'-y','LineWidth',0.5)
contour(x1,x2,Z2,'-g','LineWidth',0.5)
contour(x1,x2,Z3,'-c','LineWidth',0.5)
```

```
contour(x1,x2,Z4,'-m','LineWidth',0.5)
for n = 1:i-1
    %for tracing the points in contour plot
    plot(x_1(n),x_2(n),'Og','LineWidth',2);
    plot(x_1(n+1),x_2(n+1),'Or','LineWidth',2);
    plot(x_1(n:n+1),x_2(n:n+1),'*-r')
    xlabel('x1')
    ylabel('x2')
    %title of the contour plot
    title('Rosens Gradient Projection method');
    %These annotation used for arrow with text, You can edit the arrow
position from
    %contour plot where you will get the position [x1,y1][x2,y2]
    annotation('textarrow',[0.724 0.691],[0.802 0.633],'Color','g','String',
'x1 + 2*x2 <=5');
    annotation('textarrow',[0.430 0.575],[0.697 0.738],'Color','m', 'String',
'6x1 + x2 <=7');
    annotation('textarrow',[0.764 0.792],[0.238 0.404],'Color','c','String',
'4x1 + 3x2 <=10');
    annotation('textarrow',[0.548 0.580],[0.335 0.511],'Color','y','String',
'x2 >=0');
    annotation('textarrow',[0.430 0.517],[0.592 0.638],'Color','y','String',
'x1 >=0');
    annotation('textarrow',[0.469 0.546],[0.445 0.6],'Color','r','String',
'Feasible region');
end
%Display the table in command window
disp(T)
```

منابع

- Nonlinear Programming: Theory and Algorithms**,** Mokhtar S. Bazaraa, Hanif D. Sherali, C. M. Shetty, 3rd edition

- Linear and Nonlinear Programming, David G. Luenberger, Yinyu Ye, 3rd edition