Distributed Optimization and Learning

دکتر کبیری

مقدمه

* Classification of optimization problems

$$\text{Optimization:}\quad \min_{x\in X} \underbrace{f(x)}_{\sum f_i} \longrightarrow \text{Solver}\begin{cases}\text{single computing unit} \Rightarrow \text{Centralized}\\ \text{multiple computing units} \Rightarrow \text{Decentralized}\end{cases}$$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

انگیزه‌های حل Decentralized : ① مشکلات Computational

← محدود بودن زمان و حجم حافظه

← حجم بالای داده و پیچیدگی محاسباتی

← محاسبات Parallel

با شکستن مسئله به مسئله‌های local، جواب‌های local به دست می‌آید

ولی به دنبال جواب global مسئله هستیم

② مشکلات Structural

← پخش بودن داده‌ها در منابع مختلف

← قابلیت ارتباط محدود بین منابع مختلف

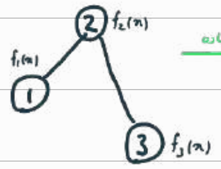
$$\min_{x_1,x_2} f_1(x_1)+f_2(x_2) \quad \text{where } x_1\in\mathbb{R}^{n_1},\ x_2\in\mathbb{R}^{n_2} \longrightarrow \text{Trivaly Decomposable}$$

$$\min_{x_1,x_2,y} f_1(x_1,y)+f_2(x_2,y) \quad \text{where } x_1\in\mathbb{R}^{n_1},\ x_2\in\mathbb{R}^{n_2},\ y\in\mathbb{R}^{n_3}$$

y: Complicating variable

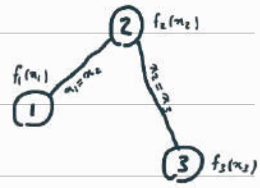
→ به سادگی قابل جداسازی نیست

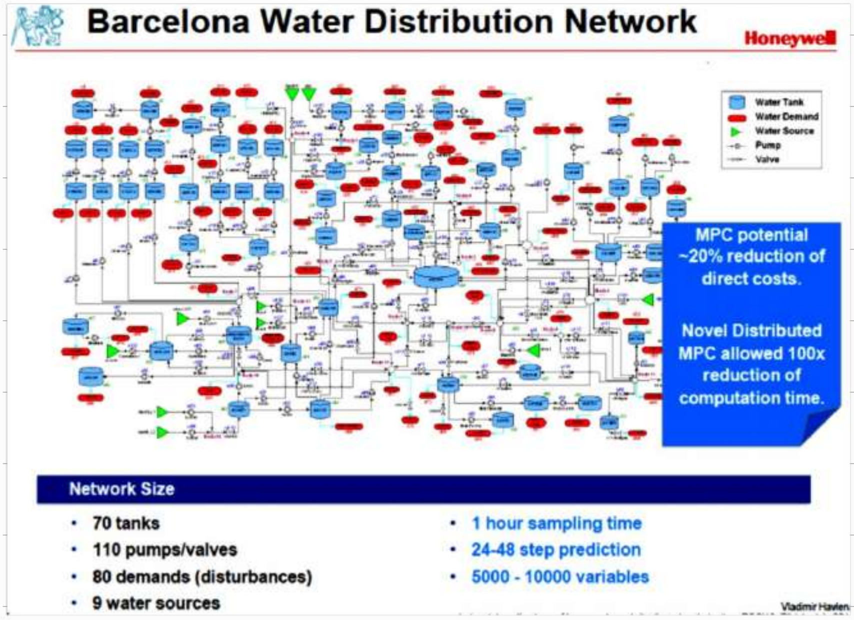
$$\min_{x} \; f_1(x) + f_2(x) + f_3(x)$$



ایده ساده

$$\min_{x_1, x_2, x_3} \; f_1(x_1) + f_2(x_2) + f_3(x_3)$$
$$\text{s.t.} \quad x_1 = x_2 \text{ and } x_2 = x_3$$





شبکه بهینه سازی شبکه آب شهر بارسلونا :

* مثال هایی از یادگیری توزیع شده

Markov Random Fields

گرافی است که به هر Node یک متغیر تصادفی نسبت داده شده

متغیر تصادفی هر Node فقط وابسته است به Node های همسایه و نسبت به سایر Node ها مستقل است

← کاربرد Image Denoising : هر پیکسل یک Node است که به پیکسل های اطراف متصل است

تابع هدف = تفاوت Solution و تصویر اصلی + $\underbrace{\text{پرش های تند بین مقادیر پیکسل ها}}_{\Rightarrow \text{smoothness}}$ → باید min شود

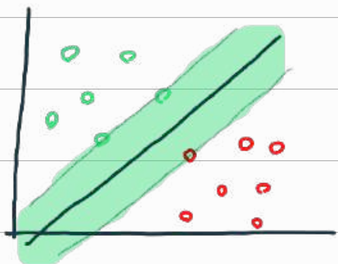
OFDMA Resource-Allocation → تخصیص باند در شبکه‌های مخابراتی

تابع هدف : مجموع Rate همه کاربران → باید Max شود

- ← از هر sub carrier یک کاربر می‌تواند استفاده کند → تداخل
- ← Rate هر کاربر از حدی بیشتر باشد
- ← توان ارسالی محدود است

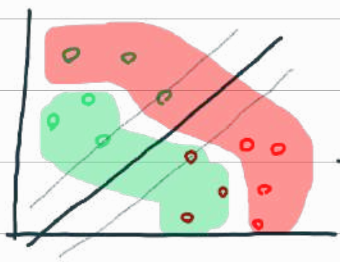
متغیرهای تصمیم‌گیری : Rate انتقال داده و توان ارسالی

Support Vector Machine



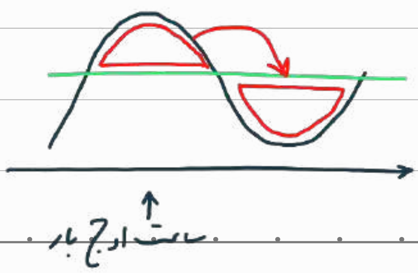
→ می‌خواهیم خط جداساز را پیدا کنیم به طوری که بیشترین Margin را داشته باشیم

نسخه Decentralized : همه داده‌ها در یک محل در دسترس نیستند



← Dataset های مختلف داده

Demand side management in smart grids



→ قیمت‌دهی به صورتی انجام شود که حجم بار در ساعت‌های مختلف تقسیم شود

هزینه تولید برق + مجموع رضایت کاربران در هر ساعت : تابع هدف

s.t. ← تولید > مجموع مصرف ← تداخل

Opinion Dynamics in Social Networks

$$x_i(t+1) = \frac{1}{d_i+k_i}\sum_{j\in\partial_i} x_j(t) + \frac{k_i}{d_i+k_i}\, x_i(0)$$ ← یکی از الگوریتم های اجماع است

opinion

stubbornness

مجموعه اطرافیان

← می‌توان دید نظرات به سمت نظرات Agent های Stubborn همگرا می‌شود

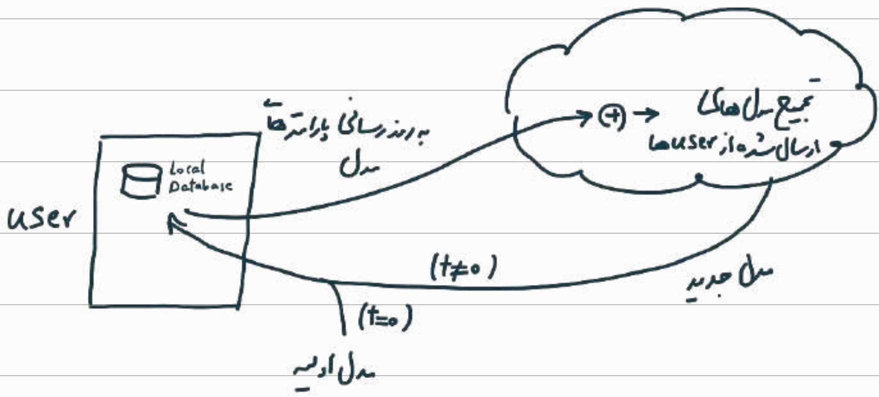
← با کنترل این Agent ها می‌توان کل شبکه را کنترل کرد!

Distributed Deep Learning / Federated Learning

داده‌ها بسیار توزیع شده هستند ← داده ها در گوشی ها هستند!

← user ها با مشوق ها ترغیب می‌شوند داده به اشتراک بگذارند

← Crowd Sourcing



بهینه سازی (Optimization)

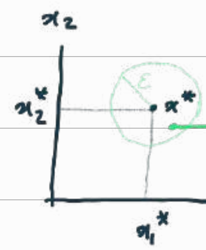
* تعاریف بهینگی

$f : R^n \to R$ ← هدف بهینه کردن تابع f اسکالر است

Feasible مجموعه : $D \subseteq R^n$, $x^* \in R^n$

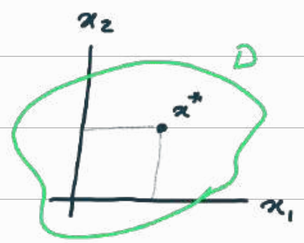
کمینه محلی (local min) : x^* را بر روی مجموعه D و برای تابع f یک کمینه محلی گویند اگر:

$$\exists \varepsilon > 0 : \forall x \in D \text{ with } \|x - x^*\| < \varepsilon : f(x^*) \leq f(x)$$



→ $f(x_1, x_2)$ → $f(x_1^*, x_2^*) \leq f(x_1, x_2)$

اگر تعریف فوق به ازای هر $x \in D$ برقرار باشد، x^* یک کمینه فراگیر (Global min) می باشد.



→ $f(x^*) \leq f(x)$

اگر روابط فوق به صورت $f(x^*) < f(x)$ برقرار باشد، در هر یک از دو حالت کمینه (محلی/فراگیر) اکید (strict) خواهد بود.

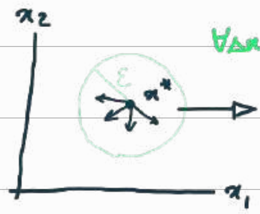
بیشینه تابع f، کمینه تابع $-f$ است.

/ /

* شرایط کمینگی

فرض: x^* کمینه محلی f است $\Rightarrow f(x^*+\Delta x) = f(x^*) + \left(\frac{\partial f}{\partial x}\Big|_{x=x^*}\right)^T \Delta x + \frac{1}{2}\Delta x^T\left(\frac{\partial^2 f}{\partial x^2}\Big|_{x=x^*}\right)\Delta x + HOT$

می خواهیم بررسی کنیم در یک همسایگی کوچک $\|\Delta x\| \ll 1$ حول x^* (کمینه محلی) بسط تیلور چه شرایطی را به دست می دهد



$\forall \Delta x \text{ st. } \|\Delta x\| < \varepsilon$

$\longrightarrow$ در هر جهت : $f(x^*+\Delta x) > f(x^*)$

فرض: مشتق تابع در x^* وجود دارد

$\longrightarrow \Delta x$ انتخاب : $\Delta x = -\eta\left(\frac{\partial f}{\partial x}\Big|_{x=x^*}\right) \qquad 0<\eta\ll 1$

تنظیم $\|\Delta x\|$

$$\Rightarrow f(x^*+\Delta x) = f(x^*) - \eta\left(\frac{\partial f}{\partial x}\Big|_{x=x^*}\right)^T\left(\frac{\partial f}{\partial x}\Big|_{x=x^*}\right) + O(\eta)$$

(در مقابل η^2 داریم) $\longrightarrow \lim_{\eta\to 0}\frac{O(\eta)}{\eta} = 0$

عبارات مرتبه بالاتر در مقابل بقیه عبارات وقتی $\eta \ll 1$ قابل صرف نظر است

$$y^Ty = [y_1 \cdots y_n]\begin{bmatrix} y_1 \\ \vdots \\ y_n\end{bmatrix} = y_1^2 + \cdots + y_n^2 = \|y\|^2$$

$$\Rightarrow f(x^*+\Delta x) = f(x^*) - \eta\left\|\frac{\partial f}{\partial x}\Big|_{x=x^*}\right\|^2 + O(\eta)$$

$$\exists \varepsilon>0 : \eta<\varepsilon \rightarrow -\eta\left\|\frac{\partial f}{\partial x}\Big|_{x=x^*}\right\|^2 + O(\eta) < 0 \Rightarrow f(x^*+\Delta x) < f(x^*) \quad \times$$

$\Leftarrow$ Δx به شکل گفته شده قابل تعریف نیست $\Leftarrow$ $\Delta x = 0$

شرط لازم مرتبه اول برای کمینگی

$\Leftarrow$ اگر x^* یک کمینه محلی برای f باشد و f در x^* مشتق پذیر باشد، آنگاه:

$$\boxed{\left.\frac{\partial f}{\partial x}\right|_{x=x^*} = 0}$$

به طور مشابه می‌توان نشان داد که این شرط برای نقطه بیشینه و زینی (Saddle) نیز برقرار است $\longrightarrow$ تمرین

در بعضی راستاها کمینه و در بعضی راستاها بیشینه

$$\Rightarrow f(x^* + \Delta x) = f(x^*) + 0 + \frac{1}{2}\Delta x^T \frac{\partial^2 f(x^*)}{\partial x^2}\Delta x + o(\|\Delta x\|^2)$$

$$\underset{ND}{H(x^*) < 0} \Rightarrow \exists \varepsilon : \|\Delta x\| < \varepsilon : \frac{1}{2}\Delta x^T H(x^*)\Delta x + o(\|\Delta x\|^2) < 0 \Rightarrow f(x^* + \Delta x) < f(x^*) \;\times$$

اگر $H(x^*)$ غیرمعین باشد بازهم به تناقض می‌رسیم چراکه در آن حالت به ازای برخی راستاها $\frac{1}{2}\Delta x^T H(x^*)\Delta x$ منفی خواهد بود

$\longrightarrow$ شرط لازم مرتبه دوم برای کمینگی: $\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x^*} = \underset{PSD}{H(x^*) \geqslant 0}$

شرط کافی مرتبه دوم: $\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x^*} = \underset{PD}{H(x^*) > 0}$

چون در این حالت $H \neq 0$، عبارت اول به عبارت دوم غلبه می‌کند

$$f(x^* + \Delta x) - f(x^*) = \underbrace{\frac{1}{2}\Delta x^T H(x^*)\Delta x}_{>0} + o(\|\Delta x\|^2) \underset{\exists \varepsilon : \|\Delta x\| < \varepsilon}{>} 0 \Rightarrow f(x^*) < f(x^* + \Delta x) \rightarrow \text{Local min}$$

شرایط مشابه برای تابع $-f$ شرایط بیشینگی را به دست می‌دهد.

بردار سطری

$$f(x^* + \Delta x) = f(x^*) + \nabla f(x^*)\,\Delta x + \Delta x^T H(x^*)\,\Delta x + HOT$$

$x^* \rightarrow$ اکسترمم $\Rightarrow \nabla f(x^*) = 0 \Rightarrow f(x^*+\Delta x) = f(x^*) + \Delta x^T H(x^*)\,\Delta x + HOT$

$H(x^*) > 0 \longrightarrow eig(H(x^*)) > 0$

فرض بردار ویژه‌های مستقل H :

$\longrightarrow$ Δx را به صورت ترکیب خطی بردارهای ویژه H می‌نویسیم

$$\Big(\sum_{i=1}^{n} \alpha_i v_i\Big)^T H(x^*) \Big(\sum_{i=1}^{n} \alpha_i v_i\Big) = \Big(\sum_{i=1}^{n} \alpha_i v_i\Big)^T \Big(\sum_{i=1}^{n} \alpha_i \lambda_i v_i\Big) = \sum_{i=1}^{n} \alpha_i^2 \underbrace{\lambda_i}_{>0} \|v_i\|^2 > 0$$

$H v_i = \lambda_i v_i$ $\qquad v_i^T v_j = 0 \quad i \neq j$

راستاهای مختلف Δx

وجود مقادیر ویژه مثبت و منفی ← بسته به اندازه تصویر Δx روی v_i های متناظر با λ_i های مثبت و منفی، $\Delta x^T H \Delta x$ می‌تواند مثبت یا منفی شود ← نقطه زینی

Δx در راستای بردار ویژه مربوط به مقدار ویژه صفر

وجود مقادیر ویژه صفر ← به ازای $\Delta x \neq 0$، $\Delta x^T H \Delta x$ صفر می‌شود ($\lambda_i = 0 \Rightarrow \alpha_i \neq 0, \alpha_j = 0, j \neq i$) $\Leftarrow$ min یا Max بودن وابسته به HOT می‌شود

Example

نقاط ایستا و نوع آن را برای تابع زیر به دست آورید

$$f(x) = x_1^2 + x_2^2 + 3x_1x_2 + 2x_2 - x_1 + 3$$

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 2x_1 + 3x_2 - 1 \\ 2x_2 + 3x_1 + 2 \end{pmatrix} = 0 \Rightarrow x^* = \begin{pmatrix} -\frac{8}{5} \\ \frac{7}{5} \end{pmatrix}$$

/ /

$$H(x) = \frac{\partial^2 f}{\partial x^2} = \begin{pmatrix} 2 & 3 \\ 3 & 2 \end{pmatrix} \qquad eig(H): \det(\lambda I - H) = 0$$

$$\Rightarrow (\lambda-2)^2 - 9 = 0 \Rightarrow \lambda - 2 = \pm 3 \Rightarrow \begin{cases} \lambda = 5 \\ \lambda = -1 \end{cases} \Rightarrow$$ x^* یک نقطه زینی است

End-Example

$$f(x) = x^T A x + b^T x + c$$

$$\Rightarrow \frac{\partial f}{\partial x} = (A + A^T)x + b = 0 \leftarrow$$ شرط لازم مرتبه اول $\xrightarrow{A \text{ معکوس پذیر}} x = -(A+A^T)^{-1} b$

$$\Rightarrow \frac{\partial^2 f}{\partial x^2} = A + A^T \begin{cases} PD: & \text{کمینه} \\ ND: & \text{بیشینه} \\ \text{غیرمعین}: & \text{زینی} \end{cases}$$

End-Example

$$f(x) = \frac{1}{2}x_2^2 + x_1 + x_2 = x^T \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}}_{A} x + (1 \quad 1)x \quad , \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\Rightarrow A + A^T = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} \nearrow \lambda = 0 \\ \searrow \lambda = 1 \end{matrix}$$

$$\frac{\partial f}{\partial x} = 0 \Rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} x + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0 \Rightarrow \begin{pmatrix} 1 \\ x_2 + 1 \end{pmatrix} = 0 \Rightarrow \begin{cases} 1 = 0 & \times \\ x_2 + 1 = 0 & \checkmark \end{cases} \rightarrow$$ شرط لازم مرتبه اول برقرار نیست

(علیرغم مشتق پذیر بودن در R^2)

$\Rightarrow$ این تابع دارای کمینه نیست

End-Example

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

$$= x^T \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} x + (1 \quad 1)\, x \quad , \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$A + A^T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \rightarrow (\lambda - 1)^2 - 1 = 0 \Rightarrow \lambda - 1 = \pm 1 \Rightarrow \begin{cases} \lambda = 0 \\ \lambda = 2 \end{cases} \rightarrow PSD$$

$$\frac{\partial f}{\partial x} = 0 \Rightarrow \begin{pmatrix} x_1 + x_2 + 1 \\ x_2 + x_1 + 1 \end{pmatrix} = 0 \Rightarrow x_1 + x_2 + 1 = 0 \checkmark$$

← شرط لازم مرتبه اول و مرتبه دوم برقرار است ولی شرط کافی برقرار نیست

$f(x) = \frac{1}{2}(x_1 + x_2 + 1)^2 \geq 0$, شرط لازم : $x_1 + x_2 + 1 = 0 \Rightarrow$ شرط لازم min را نتیجه می‌دهد

← f بی‌نهایت min در $x_1 + x_2 + 1 = 0$ در مقدار $f = 0$ دارد

---

* توابع Convex و Concave (محدب و مقعر)

تعریف : تابع f بر مجموعه محدب Ω محدب نامیده می‌شود اگر به ازای هر $x_1, x_2 \in \Omega$ و هر $0 \leq \alpha \leq 1$ داشته باشیم :

$$f(\alpha x_1 + (1-\alpha) x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

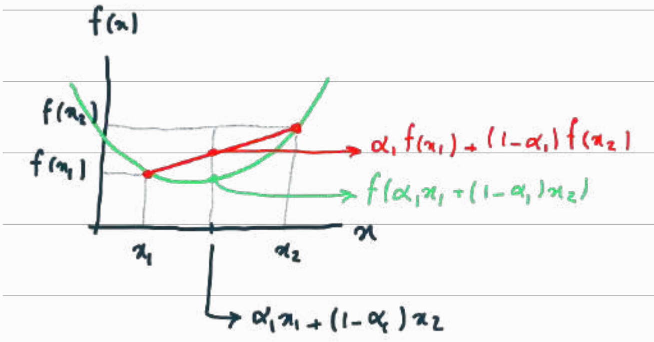
برای این که خط واصل هر دو نقطه وجود داشته باشد

مجموعه محدب Ω : $x_1, x_2 \in \Omega \Rightarrow \alpha x_1 + (1-\alpha) x_2 \in \Omega$

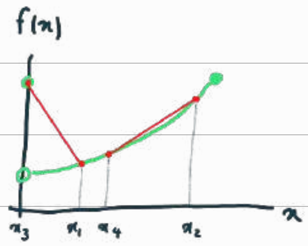
همچنین اگر به ازای $0 < \alpha < 1$ و $x_1 \neq x_2$ داشته باشیم :

$$\underbrace{f(\alpha x_1 + (1-\alpha) x_2)}_{\text{خط واصل } x_1 \text{ و } x_2} < \alpha f(x_1) + (1-\alpha) f(x_2)$$

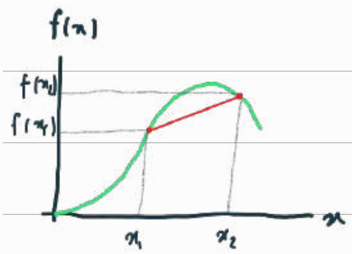
آنگاه f محدب اکید (Strictly Convex) نامیده می‌شود.



تابع محدب اکید:



تابع محدب با ناپیوستگی:



تابع غیر محدب (Non Convex):

← برای محدب بودن یک تابع، خط واصل بین هر دو نقطه باید بالاتر از منحنی قرار گیرد

تعریف: تابع f بر مجموعه محدب Ω مقعر (Concave) است اگر $f=-y$ محدب باشد

تعریف: تابع f بر مجموعه محدب Ω مقعر اکید (Strictly Concave) است اگر $f=-y$ مقعر اکید باشد

خواص توابع Convex: ① اگر $f_1,\dots,f_n$ توابع محدب و $i=1,\dots,n$ $a_i \geq 0$ باشند، آنگاه $\sum_{i=1}^{n} a_i f_i$ محدب است

اثبات:

محدب $f_1 \Rightarrow f_1(\alpha x_1+(1-\alpha)x_2) \leq \alpha f_1(x_1)+(1-\alpha) f_1(x_2) \quad \times a_1$

$\vdots \qquad \vdots \qquad \vdots$

محدب $f_n \Rightarrow f_n(\alpha x_1+(1-\alpha)x_2) \leq \alpha f_n(x_1)+(1-\alpha) f_n(x_2) \quad \times a_n$

$$\oplus \Rightarrow \sum_i a_i f_i(\alpha x_1+(1-\alpha)x_2) \leq \alpha \sum_i a_i f_i(x_1) + (1-\alpha)\sum_i a_i f_i(x_2) \checkmark$$

/ /

ارتباط بین توابع محدب و مجموعه‌های محدب

2 اگر f تابعی محدب بر مجموعه محدب Ω باشد آنگاه $\Gamma_c=\{x : x\in\Omega, f(x)\leq c\}$ یک مجموعه محدب است

اثبات:

$x_1\in\Gamma_c \Rightarrow f(x_1)\leq c$

$x_2\in\Gamma_c \Rightarrow f(x_2)\leq c$

$\alpha f(x_1)+(1-\alpha)f(x_2)\leq c$

$\Rightarrow f(\alpha x_1+(1-\alpha)x_2)\leq \alpha f(x_1)+(1-\alpha)f(x_2)\leq c$

$\Rightarrow \alpha x_1+(1-\alpha)x_2\in\Gamma_c$ ✓

f مشتق‌پذیر

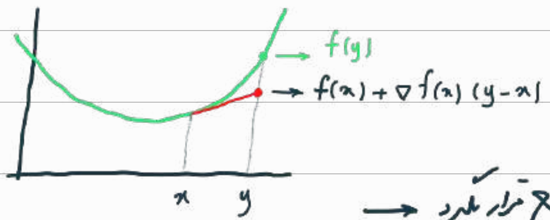
3 اگر $f\in C^1$ آنگاه f بر Ω محدب است اگر و فقط اگر $\forall x,y\in\Omega : f(y)\geq f(x)+\nabla f(x)(y-x)$

(محدب اکید) $(>)$

اثبات شهودی:

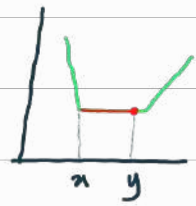
اثبات ریاضی در کتاب [illegible]

تابع محدب:

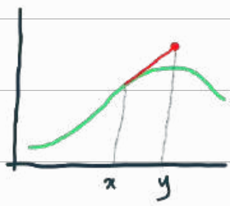


→ خطوط مماس زیر تابع قرار بگیرد

تابع محدب غیراکید ←



تابع غیرمحدب:



→ خطوط مماس بالای تابع قرار گرفته

توابع مشتق‌پذیر از مرتبه 2

4 اگر $f\in C^2$ و f تابعی محدب بر مجموعه محدب Ω باشد آنگاه ماتریس هسین f نیمه‌معین مثبت بر Ω است (و برعکس) ← شرط لازم و کافی

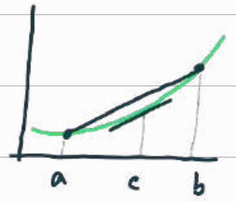
اثبات: براساس قضیه تیلور - قضیه مقدار میانگین مرتبه دوم:

$$f(y)=f(x)+\nabla f(x)(y-x)+\frac{1}{2}(y-x)^T H_f(x+\alpha(y-x))(y-x)$$

$y=x+(y-x)=x+\Delta x$

$\exists\, 0\leq\alpha\leq 1$ ← رابطه فوق به ازای یک α برقرار است

(یادآوری: قضیه مقدار میانگین مرتبه اول: $f'(c)=\frac{f(b)-f(a)}{b-a}$ $\quad \exists\, a<c<b$)



اثبات رفت:

$$H_f \geq 0 \Rightarrow (y-x)^T H_f(x+\alpha(y-x))(y-x) \geq 0$$

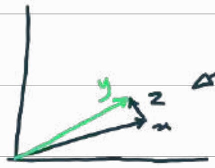
$$\xrightarrow{\text{رابطه اصلی}} f(y) \geq f(x) + \nabla f(x)(y-x) \Rightarrow$$ f محدب است

اثبات برگشت:

برقرار نیست $H_f \geq 0 \Rightarrow \exists z, x: \; z^T H_f(x) z < 0$

$\longrightarrow (y-x):$ در راستای z باشد

به اندازه کافی کوچک باشد $\rightarrow x+\alpha(y-x) \sim x$

$\Rightarrow$  $\leftarrow \|z\| \to 0 \Rightarrow y-x=z$

$$z^T H_f(x) z < 0 \xrightarrow{\text{پیوستگی ماتریس } H_f \text{ نسبت به } x} z^T H_f(x+\varepsilon) z < 0 \;, \|\varepsilon\| \to 0$$

$$\xrightarrow{\|z\|\to 0} \frac{1}{2} z^T H_f(x+\alpha z) z < 0 \Rightarrow f(y) < f(x) + \nabla f(x)(y-x)$$

اگر ماتریس هسین معین مثبت باشد ($H_f > 0$) آنگاه تابع محدب اکید است $\Leftarrow$ $H_f > 0 \Rightarrow$ محدب اکید

رابطه برگشت برقرار نیست

5 اگر تابع f بر مجموعه محدب Ω محدب باشد، آنگاه هر کمینه محلی (local) f یک کمینه فراگیر (global) است

اثبات:

فرض: x^*: کمینه محلی برای f $\Rightarrow \exists y \in \Omega : f(y) < f(x^*)$

(غیر فراگیر)

$$\alpha f(x^*) + (1-\alpha) f(x^*) = f(x^*)$$

$$\Rightarrow f(\alpha y + (1-\alpha)x^*) \leq \alpha f(y) + (1-\alpha) f(x^*) < f(x^*) \quad , \quad 0<\alpha\leq 1$$

$\alpha \to 0 \Rightarrow$ $\alpha y + (1-\alpha)x^*$ یک همسایگی x^* است

$\hookrightarrow f(\alpha y + (1-\alpha)x^*) < f(x^*) \Rightarrow$ ✗ x^* حتی کمینه محلی هم نیست

6. مجموعه نقاط کمینه تابع f بر مجموعه محدب Ω محدب است

$M \subset \Omega$

اثبات:

$x,y \in M \Rightarrow x,y \in \Omega \xRightarrow{\Omega \text{ محدب}} \alpha x + (1-\alpha) y \in \Omega$

هر کمینه f محدب سراسری است

$$\Rightarrow f(\alpha x + (1-\alpha)y) \underset{f\ \text{محدب}}{\leq} \alpha f(x) + (1-\alpha) f(y) = f(x) = f(y)$$

چون x و y کمینه هستند $f(\dots) < f(x)=f(y)$ نمی‌تواند برقرار باشد $\Rightarrow f(\alpha x + (1-\alpha)y) = f(x) = f(y)$

کمینه است

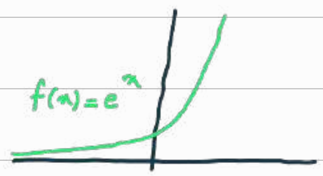
$\Rightarrow \alpha x + (1-\alpha) y \in M$ ✓

7. تابع محدب اکید اگر کمینه داشته باشد آن کمینه یکتاست

اثبات: فرض کنیم یکتا نباشد $\Rightarrow x,y \in M \Rightarrow \alpha x + (1-\alpha)y \in M$

$$\Rightarrow f(x) = f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y) = f(x) \quad ✗$$

تابع محدب اکید لزوماً کمینه ندارد:



$\leftarrow H_f = e^x > 0 \leftarrow$ محدب اکید است

ولی کمینه ندارد

8 فرض کنید تابع محدب و مشتق‌پذیر f بر مجموعه محدب Ω موجود است. شرط لازم و کافی برای این که $x^*\in\Omega$ یک کمینه فراگیر برای f باشد این است که: $\nabla f(x^*)=0$

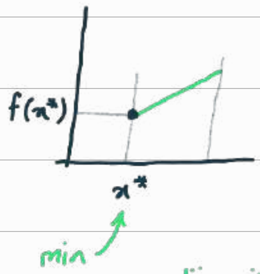
ویژگی بسیار مفیدی برای بهینه‌سازی است ←

اثبات: شرط لازم بودن برای بهینگی قبلاً نشان داده شد

شرط کافی بودن:

تابع محدب: $\forall y\in\Omega \quad f(y)\geqslant f(x)+\nabla f(x)(y-x)$

$$\nabla f(x^*)=0 \Rightarrow f(y)\geqslant f(x^*) \checkmark$$



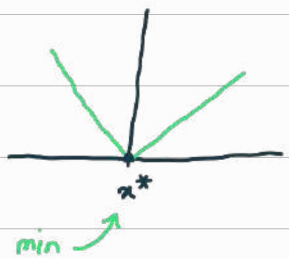
$\rightarrow f(x)=ax+b \Rightarrow H_f=0\geqslant 0 \Rightarrow$ محدب است

چون در x^* مشتق وجود ندارد علی‌رغم min بودن

شرط $\nabla f(x^*)=0$ برقرار نیست (می‌توان ∇f را در راستای دلخواه در نظر گرفت (مشابه کاری که در اثبات انجام شد))

$$f(y)\geqslant f(x^*)+\nabla f(x^*)(y-x^*) \xrightarrow{f(y)\geqslant f(x^*)} \underbrace{\nabla f(x^*)}_{a>0}\underbrace{(y-x^*)}_{y>x^*\Rightarrow +}\geqslant 0$$

→ برای مشتق راست که وجود دارد برقرار است



$\rightarrow$ Convex است، مشتق‌پذیر نیست

$$\Rightarrow \begin{cases} y>x^* \Rightarrow \nabla f^+(x^*)>0 \Rightarrow \nabla f(x^*)(y-x^*)>0 \\ y<x^* \Rightarrow \nabla f^-(x^*)<0 \Rightarrow \nabla f(x^*)(y-x^*)>0 \end{cases} \checkmark$$

← در توابع Convex، کمینه می‌تواند در نقاط مرز یا نقاط غیرمشتق‌پذیر باشد

Example

توابع مربعی:

$f(x) = \frac{1}{2} x^T P_{n\times n} x + q^T_{1\times n} x + r$ $\quad P = P^T \longleftarrow$ قسمت غیرمتقارن P حذف می‌شود

$$\Rightarrow \nabla f^T = \frac{1}{2}(P + P^T)x + q = Px + q = 0 \Rightarrow Px = -q$$

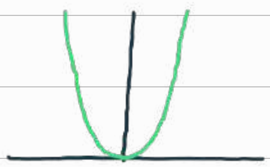
$$\nabla^2 f = H = P$$

$\Rightarrow \begin{cases} P \succeq 0 \Rightarrow f \text{ محدب است} \Rightarrow P \text{, Full Rank نیست} \Rightarrow \\ P \succ 0 \Rightarrow f \text{ محدب اکید است} \Rightarrow x^* = -P^{-1}q \end{cases}$

(سطر اول، ادامه:) $Px = -q$ جواب ندارد یا بی‌نهایت جواب دارد

EndExample

$f(x) = x^4$



$\rightarrow$ محدب اکید

$$\Rightarrow f'(x) = 4x^3 \Rightarrow f''(x) = 12x^2 \Rightarrow f''(0) = 0 \Rightarrow H_f \succeq 0$$

← در این مثال با این که ماتریس هسین نیمه معین مثبت است تابع f محدب اکید است.

$\Rightarrow \begin{cases} H_f \succ 0 \Rightarrow \text{محدب اکید} \\ \text{محدب اکید} \not\Rightarrow H_f \succ 0 \end{cases}$

End Example

$$f(x,y) = \frac{x^2}{y} \qquad y > 0$$

$$\nabla^T f(x,y) = \begin{pmatrix} \frac{2x}{y} \\ \frac{-x^2}{y^2} \end{pmatrix} \Rightarrow \nabla^2 f(x,y) = \begin{pmatrix} \frac{2}{y} & \frac{-2x}{y^2} \\ \frac{-2x}{y^2} & \frac{2x^2}{y^3} \end{pmatrix} = H(x,y) = \frac{2}{y^3} \begin{pmatrix} y^2 & -xy \\ -xy & x^2 \end{pmatrix}$$

$$y > 0 \Rightarrow \frac{2}{y^3} > 0 \quad , \quad y^2 > 0 \quad , \quad \det(H) = x^2y^2 - x^2y^2 = 0 \Rightarrow \underset{(PSD)}{H \succeq 0} \rightarrow$$ محدب

$$\nabla^T f(x,y) = \begin{pmatrix} \frac{2x}{y} \\ \frac{-x^2}{y^2} \end{pmatrix} = 0 \Rightarrow x = 0 \,,\, y > 0 \Rightarrow$$ $(0,y)$ نقاط کمینه فراگیر تابع $f(x,y)$ می‌باشند

end

* الگوریتم‌های بهینه‌سازی

به دست آوردن حل بسته برای مسئله بهینه‌سازی همیشه ممکن نیست ⇐ از الگوریتم‌های iterative استفاده می‌شود

$$\min_x f(x) \longrightarrow x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x^* = \underset{x}{\mathrm{Argmin}}\, f(x)$$

هدف رسیدن به نقطه بهینه کننده f به صورت تکراری است

الگوریتم مرتبه اول تندترین شیب (Steepest Descent / SD):

از تقریب مرتبه اول بسط تیلور استفاده می‌شود

$$\underbrace{f(x+\Delta x) = f(x) + f_x^T \Delta x}_{\text{تقریب مرتبه اول}} + \frac{1}{2} \Delta x^T H(x) \Delta x + HOT$$

$\rightarrow$ در مرحله k ام: $f(x_k+\Delta x_k) \simeq f(x_k) + f_x^T(x_k)\Delta x_k$ $\rightarrow$ ترتیب مرتبه اول f در مرحله k

$$\begin{cases} \Delta x_k = x_{k+1} - x_k \\ x_{k+1} = x_k + \Delta x_k \end{cases}$$ $\Rightarrow$ با انتخاب مقدار Δx_k مقدار x_{k+1} تعیین می‌شود

$$\underbrace{f(x_k+\Delta x_k)}_{x_{k+1}} \simeq f(x_k) + \underbrace{f_x^T(x_k)\Delta x_k}_{<0} \xRightarrow{\Delta x \text{ به اندازه کافی کوچک}} f(x_{k+1}) < f(x_k)$$

اندازه و جهت Δx_k باید طوری تعیین شود که این موارد برقرار شود

$$\longrightarrow \Delta x_k = \underset{\text{اندازه}}{\lambda_k}\, \underset{\text{بردار یکه جهت}}{u_k}$$

تندترین شیب $\Leftarrow$ باشد $180°$، $f_x(x_k)$ و Δx_k زاویه بین $\Rightarrow u_k = \dfrac{-f_x(x_k)}{\|f_x(x_k)\|}$ $\Rightarrow f_x^T(x_k)\Delta x_k < 0$

می‌توان λ_k را کوچک انتخاب کرد تا $f_x^T(x_k)\Delta x_k$ نسبت به H.O.T. غالب شود و $f(x_{k+1}) < f(x_k)$ برقرار شود.
در صورت امکان می‌توان λ_k را به صورت بهینه به دست آورد:

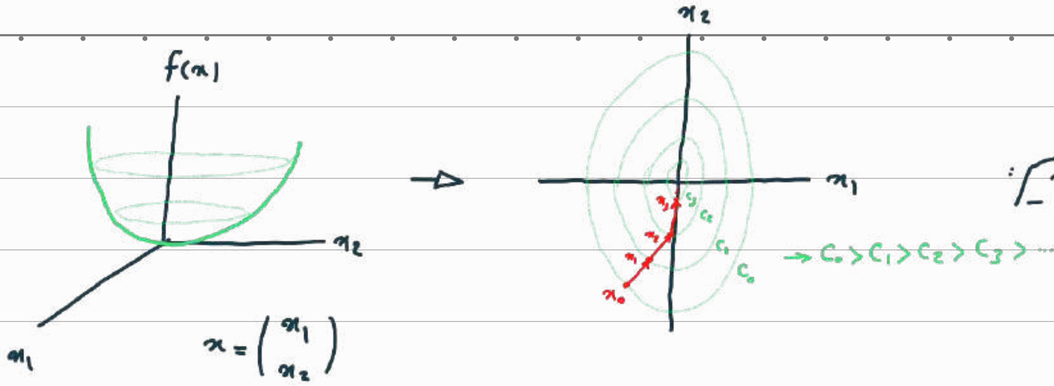
$$\lambda_k^* = \underset{\lambda_k}{\text{Argmin}}\; f(\underbrace{x_k + \lambda_k u_k}_{x_{k+1}})$$

برای حل بهینه‌سازی λ_k^* می‌توان از الگوریتم‌های Line Search (Backtracking Line Search, ...) هم استفاده کرد
(بهینه‌سازی اسکالر است و ساده‌تر است) $\leftarrow$ مطالعه

$$\Rightarrow x_{k+1} = x_k + \lambda_k u_k$$

در ابتدای الگوریتم λ_k بزرگ انتخاب می‌شود و با نزدیک شدن به نقطه بهینه λ_k کوچک انتخاب می‌شود تا به نقطه بهینه همگرا شویم

/ /



لزوماً در یک مرحله به نقطه بهینه نمی‌رسیم:

همگرایی روش SD : برای توابع محدب و مشتق‌پذیر با انتخاب مناسب گام به روز رسانی (λ_k) الگوریتم SD به مقدار کمینه تابع همگرا می‌شود

λ_k انتخاب :

- $\lambda_k = \dfrac{\|f_x(x_n)\|}{L}$ $\longrightarrow$ ثابت لیپشیتز: $\|f(x) - f(y)\| \le L\|x - y\|$

می‌تواند حد بالای مشتق انتخاب شود

- Backtracking Line Search

در نزدیکی نقطه بهینه الگوریتم کند می‌شود و نمی‌خواهیم بی‌نهایت iteration داشته باشیم

تست خاتمه محاسبات : الگوریتم متوقف می‌شود اگر حداقل یکی از معیارهای زیر برآورده شود :

1. $\|\nabla f(x_k)\| \le \varepsilon_\nabla$

برای m تکرار این شرط برقرار بماند

2. $|f(x_i) - f(x_{i-1})| < \varepsilon_f \qquad i = k, k-1, \dots, k-m$

3. $\|x_i - x_{i-1}\| < \varepsilon_x \qquad i = k, k-1, \dots, k-m$

4. $\lambda_k \le k\lambda_0 \qquad 0 < k < 1$

5. a = Number of iterations $\longrightarrow$ مثلاً فقط صد iteration الگوریتم پیش برود

Example

الگوریتم SD برای توابع مربعی :

$$f(x) = \frac{1}{2}x^T A x + b^T x + c \qquad \overset{\text{متقارن}}{\underset{PD}{A > 0}} \Rightarrow \text{Strictly Convex}$$

$$f_x(x) = Ax + b \quad \Rightarrow x_{k+1} = x_k - \lambda_k \left(\frac{Ax_k + b}{\|Ax_k + b\|} \right)$$

$$f(x_k + \lambda_k u_k) = \frac{1}{2}(x_k + \lambda_k u_k)^T A (x_k + \lambda_k u_k) + b^T(x_k + \lambda_k u_k) + c$$

$$\Rightarrow \frac{\partial f}{\partial \lambda_k} = \left(\frac{\partial f}{\partial (x_k + \lambda_k u_k)} \right)^T \frac{\partial (x_k + \lambda_k u_k)}{\partial \lambda_k} = [A(x_k + \lambda_k u_k) + b]^T u_k = 0$$

$$\Rightarrow \underbrace{x_k^T A u_k}_{= u_k^T A x_k} + \lambda_k u_k^T A u_k + \underbrace{b^T u_k}_{= u_k^T b} = 0 \quad , \quad H_k = u_k^T A u_k > 0 \rightarrow \text{نتیجه ب } \lambda_k \text{ مثبت است}$$

$$\Rightarrow \lambda_k = \frac{-u_k^T (Ax_k + b)}{u_k^T A u_k} = \frac{-u_k^T f_x(x_k)}{u_k^T A u_k} = \frac{f_x^T(x_k) f_x(x_k)}{f_x^T(x_k) A f_x(x_k)} \|f_x(x_k)\|$$

$$\Rightarrow x_{k+1} = x_k + \Delta x_k = x_k + \lambda_k u_k = x_k - \frac{f_x^T(x_k) f_x(x_k)}{f_x^T(x_k) A f_x(x_k)} f_x(x_k)$$

End of Example

$$f(x) = x_1^2 + 25x_2^2 = \frac{1}{2}x^T \begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix} x \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$\Rightarrow f_x(x) = \begin{pmatrix} 2x_1 \\ 50x_2 \end{pmatrix} \quad , \quad H = A > 0$$

$$x_0 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \Rightarrow f(x_0) = 104 \quad , \quad f_x(x_0) = \begin{pmatrix} 4 \\ 100 \end{pmatrix}$$

$$x_1 = x_0 - \frac{(4 \;\; 100)\begin{pmatrix} 4 \\ 100 \end{pmatrix}}{(4 \;\; 100)\begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix}\begin{pmatrix} 4 \\ 100 \end{pmatrix}} \begin{pmatrix} 4 \\ 100 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 0.02003 \begin{pmatrix} 4 \\ 100 \end{pmatrix} = \begin{pmatrix} 1.92 \\ -0.003 \end{pmatrix} \Rightarrow f(x_1) = 3.69$$

یعنی f از 104 به 3.69 در یک iteration

Example

حل دستگاه معادلات :

$$\begin{cases} 3x_1 - \cos(x_2 x_3) - \frac{3}{2} = 0 \\ 4x_1^2 - 625x_2^2 + 2x_2 - 1 = 0 \\ \exp(-x_1 x_2) + 20x_3 + \frac{10\pi - 3}{3} = 0 \end{cases}$$

$\Rightarrow$ خطا :
$$\begin{cases} e_1(x) = 3x_1 - \cos(x_2 x_3) - \frac{3}{2} \\ e_2(x) = 4x_1^2 - 625x_2^2 + 2x_2 - 1 \\ e_3(x) = \exp(-x_1 x_2) + 20x_3 + \frac{10\pi - 3}{3} \end{cases} \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$\Rightarrow F(x) = \frac{1}{2}\left(e_1^2(x) + e_2^2(x) + e_3^2(x)\right)$$

$F(x) = 0 \Rightarrow$ حل دستگاه

$F(x) \to 0 \Rightarrow$ حل تقریبی دستگاه

اگر بخواهیم مستقیم حل کنیم، با نوشتن شرط لازم مرتبه اول به دستگاه معادلات اولیه می‌رسیم که قابل حل نیست

← از الگوریتم های iterative استفاده می‌کنیم

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow F(x_0) = \frac{1}{2}\left((-2.5)^2 + (-1)^2 + (10.472)^2\right) = 58.456 \quad , \quad F_x(x_0) = \begin{pmatrix} -7.5 \\ -2 \\ 209.44 \end{pmatrix}$$

ضریب $\frac{1}{\|F_x\|}$ می‌تواند داخل λ_k فرض شود

$$x_1 = x_0 - \lambda_0 F_x(x_0) \qquad \Rightarrow x_1 = x_0 - \lambda_0 \begin{pmatrix} -7.5 \\ -2 \\ 209.44 \end{pmatrix}$$

λ_0 انتخاب : $F(x_1) < F(x_0) \Rightarrow \lambda_0 = 0.001$

$$\Rightarrow x_1 = \begin{pmatrix} 0.0075 \\ 0.002 \\ -0.20944 \end{pmatrix} \Rightarrow F(x_1) = 23.306 < 58.456 = F(x_0) \checkmark$$

end

روش SD : { مزیت : سریع به جواب کمینه نزدیک می‌شود
ایراد : همگرایی کند پس از نزدیک شدن به جواب کمینه ← می‌توان از یک روش مکمل مثل روش نیوتن استفاده کرد

روش مرتبه دوم نیوتن :

$$f(x_k + \Delta x_k) \simeq f(x_k) + \nabla f(x_k)\Delta x_k + \frac{1}{2}\Delta x_k^T H(x_k)\Delta x_k$$ ← با انتخاب Δx_k مناسب این عبارت Quadratic را کمینه می‌کنیم

$$\frac{\partial f(x_k + \Delta x_k)}{\partial \Delta x_k} = \nabla^T f(x_k) + H(x_k)\Delta x_k = 0$$

فرض : $H(x_k) > 0$ ← شرط کافی مرتبه 2 ← H^{-1} وجود دارد

$$\Rightarrow \underline{\Delta x_k = -H^{-1}(x_k)\nabla^T f(x_k)} \Rightarrow x_{k+1} = x_k - H^{-1}(x_k)\nabla^T f(x_k)$$

نیاز به معکوس گیری از H_k ⇐ محاسبات زیاد

شرایط همگرایی روش نیوتن پیچیده‌تر از روش SD است (Bounded و لیپشیتز بودن H و به قدر کافی نزدیک بودن x_0 به x^* و ...)

Example

$$f(x) = \frac{1}{2}x^T A x + b^T x + c$$

$$\Rightarrow \nabla^T f(x) = Ax + b, \quad H(x) = A,$$ دلخواه x_0

$$x_1 = x_0 - A^{-1}(Ax_0 + b) = x_0 - x_0 - A^{-1}b = -A^{-1}b = x^*$$ → به کمینه یک‌تای تابع رسیدیم!

← برای تابع درجه دو الگوریتم نیوتن در یک مرحله به معادله کمینه تابع می‌رسد ← سرعت روش نیوتن از روش SD بیشتر است

end

یک ایده برای استفاده از مزایای روش‌های SD و نیوتن: استفاده از SD در مراحل اولیه و نزدیک شدن به جواب کمینه و سپس استفاده از روش نیوتن جهت تسریع در همگرایی به نقطه کمینه

↳ در نزدیکی جواب کمینه شکل تابع به تابع Quadratic نزدیک می‌شود

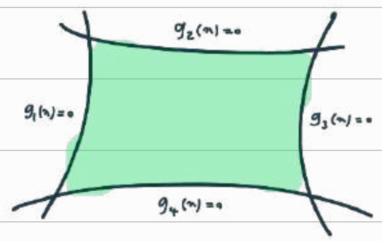
* بهینه‌سازی مقید

minimize $f(x)$

$x \in \Omega \subseteq R^n$ → فرض می‌کنیم x^* نقطه داخلی Ω است

قیود تساوی و نامساوی را به مسئله اضافه می‌کنیم :

$$h_1(x) = 0 \qquad g_1(x) \le 0$$
$$h_2(x) = 0 \qquad g_2(x) \le 0$$
$$\vdots \qquad \vdots$$
$$h_m(x) = 0 \qquad g_p(x) \le 0$$



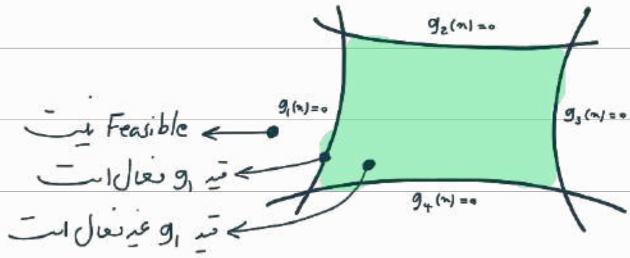
با قیود نامساوی می‌توان ناحیه Feasible را مشخص کرد

تعداد قیود تساوی از تعداد متغیرها کمتر است (در فضای R^3 وجود 3 قید یک نقطه را مشخص می‌کند!) ⇐ $m < n$

فرض: f و g_j و h_i ها پیوسته و مشتق‌پذیر از مرتبه دوم هستند ($i=1,\dots,m$ و $j=1,\dots,p$)

ابتدا فقط قیود تساوی را بررسی می‌کنیم (اهمیت قیود نامساوی هم در حالت تساوی‌شان است)

قید غیرفعال: $g_j(x^*)<0$ ← $g_j(x^*) \le 0$ → قید فعال: $g_j(x^*)=0$



تعریف: به مجموعه‌ای از x ها که در همه قیود صدق کنند Feasible یا شدنی می‌گویند.

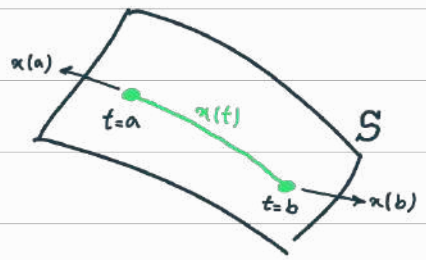
قیود تساوی: با فرض استقلال قیود تساوی، مجموعه قیود یک ابرصفحه با بعد $n-m$ را تشکیل می‌دهند

R^3 , $x_1+x_2+x_3=0 \Rightarrow 3-1=2 \longrightarrow$ صفحه در فضای R^3

با این قید یک وابستگی داریم و با داشتن هر دو متغیر از x_1 و x_2 و x_3 سومی به دست می‌آید

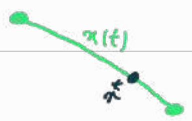
- صفحه (ابرصفحه) مشخص شده با قیود را با S نمایش می‌دهیم ← جواب بهینه داخل S است

- یک مسیر بر صفحه S را با مجموعه نقاط $x(t)\in S$ نشان می‌دهیم که در بازه $a\le t\le b$ با پارامتر t پارامتریزه شده است.

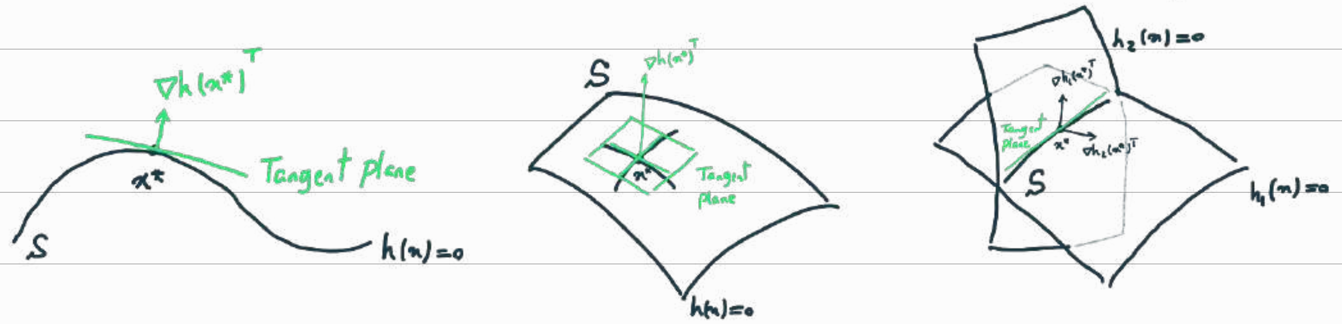


- یک مسیر مشتق‌پذیر است اگر $\dot{x}=\frac{dx}{dt}$ وجود داشته باشد و مشتق‌پذیر از مرتبه دوم است اگر $\ddot{x}(t)$ وجود داشته باشد

- مسیر $x(t)$ از نقطه x^* بر S عبور می‌کند اگر وجود داشته باشد $t^*\in\{t: a\le t\le b\}$ به طوری که $x^*=x(t^*)$



صفحه مماسی: اگر همهٔ مسیرهای مشتق‌پذیر از S که از x^* عبور می‌کند را در نظر بگیریم، صفحهٔ مماسی در x^* به مجموعهٔ همهٔ مشتق‌های همهٔ مسیرها در x^* گفته می‌شود



تعریف: به نقطهٔ x^* که $h(x^*)=0$ را برآورده می‌کند رگولار گویند اگر بردارهای گرادیان $\nabla h_1(x^*), \dots, \nabla h_m(x^*)$ مستقل خطی باشند

قضیه: در نقطهٔ رگولار x^* بر صفحهٔ S که با $h(x^*)=0$ مشخص شده است صفحهٔ مماسی به صورت زیر تعریف می‌گردد:

$M = \{y : \nabla h(x^*)\, y = 0\}$ ⟵ y هایی که بردار گرادیان بر آنها عمود است

اثبات کامل در کتاب لیونبرگر

اثبات: مسیر روی S: $x(t) \Rightarrow h(x(t)) = 0 \xRightarrow{\text{مشتق}} \nabla h(x(t))\, \underbrace{\dot{x}(t)}_{\text{مشتق مسیر}}\Big|_{t=t^*} = 0$

$$\begin{cases} h_1 = 0 \\ \vdots \\ h_m = 0 \end{cases} \qquad \nabla h(x) = \begin{bmatrix} \nabla h_1(x) \\ \vdots \\ \nabla h_m(x) \end{bmatrix}_{m \times n}$$

همهٔ بردارهای گرادیان بر همهٔ مسیرها (x دلخواه) و در نتیجه بر صفحهٔ مماسی عمودند

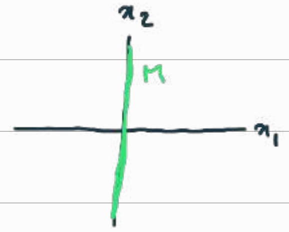
$\nabla h_i(x^*)$ های مستقل خطی $\Rightarrow \text{Rank}(\nabla h(x^*)) = m \Rightarrow$ بعد فضای پوچی $= n-m$

در زیرفضای عمود برهم

⇐ صفحهٔ مماسی فضای متعامد بر سطرهای ماتریس گرادیان است

Example

$h(x_1, x_2) = x_1 = 0$



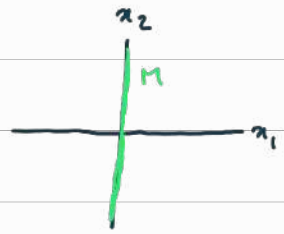
صفحه مماس بر نقطه دلخواه $x^*=(0,x_2^*)$ محور x_2 خواهد بود

$$\nabla h(x_1,x_2) = (1 \quad 0) \Rightarrow (1 \quad 0)y = 0$$

$$\Rightarrow (1 \quad 0)\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = 0 \Rightarrow y_1 = 0 \Rightarrow M = \{y : (1 \quad 0)y = 0\} = \text{محور } x_2$$

End Example

$h(x_1, x_2) = x_1^2 = 0$



$$\nabla h(x^*) = (2x_1^* \quad 0)$$

$$x^* \in h \Rightarrow x_1^* = 0 \Rightarrow \nabla h(x^*) = (0 \quad 0) \longrightarrow$$ بردار گرادیان مستقل نیست

$$\forall y \in \mathbb{R}^2 \Rightarrow \nabla h(x^*)y = 0 \Rightarrow M = \mathbb{R}^2 \; \times$$

چون x^* رگولار نیست نمی‌توان با این رابطه صفحه مماس را به دست آورد

end

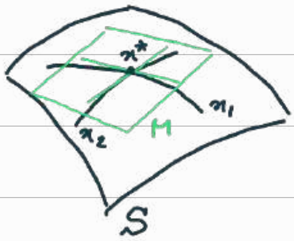
لم: اگر x^* نقطه رگولار قید $h(x)=0$ باشد و همچنین نقطه بحرانی محلی برای f با توجه به قیود باشد، آنگاه به ازای هر $y \in \mathbb{R}^n$ اگر $\underbrace{\nabla h(x^*)y = 0}_{\text{صفحه مماس}}$ آنگاه $\nabla f(x^*)y = 0$ خواهد بود.

∇f هم بر صفحه مماس عمود است

مسیر دلخواه گذرنده از x^*

اثبات: x^* اکسترمم محلی f است $\Rightarrow \frac{d}{dt} f(x(t))\Big|_{t=t^*} = 0 \Rightarrow \nabla f(x(t^*))\,\dot{x}(t^*) = 0$

صفحه مماس



$\Leftarrow$ بردار $\nabla f(x^*)$ بر همهٔ شتاب‌های همهٔ مسیرهای گذرنده از x^* در x^* عمود است

$\Leftarrow \nabla f(x^*)$ بر همهٔ بردارهای y که در عبارت $\nabla h(x^*)y=0$ صدق می‌کند که همان صفحه مماس می‌باشد عمود است

$\nabla f(x^*)$ به $\Rightarrow \{\nabla h_1(x^*), \dots, \nabla h_m(x^*)\}$ وابسته است

صفحه مماس : $n-m$

بردارهای گرادیان $\{\nabla h_1 \cdots \nabla h_m\}$: یک زیرفضا با بعد m و عمود بر صفحه مماس

کل فضای عمود بر صفحه مماس را span می‌کند

شرط لازم بهینگی

قضیه : اگر x^* یک کمینه محلی f با توجه به قیود $h(x)=0$ باشد و x^* یک نقطه رگولار برای این قیود باشد ، آنگاه بردار $\lambda \in \mathbb{R}^m$ وجود دارد به طوری که :

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) = 0$$

∇f به ∇h_i ها وابسته است و به صورت ترکیب خطی آن‌ها قابل بیان است : $\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x^*) = 0$

می‌توانیم شرط لازم بهینگی را نسبت به تابع زیر که لاگرانژین گفته می‌شود بیان کنیم :

$$\underbrace{\mathcal{L}(x,\lambda) = f(x) + \lambda^T h(x)}_{\lambda:\ \text{ضرایب لاگرانژ}} \Rightarrow \begin{cases} \dfrac{\partial \mathcal{L}}{\partial x} = 0 \Rightarrow \nabla f(x) + \lambda^T \nabla h(x) = 0 & \rightarrow \text{شرط لازم بهینگی} \\[2ex] \dfrac{\partial \mathcal{L}}{\partial \lambda} = 0 \Rightarrow h(x) = 0 & \rightarrow \text{قیود} \end{cases}$$

متغیرها : $\lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix}$ و $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ $\leftarrow$ $n+m$ مجهول $\Leftarrow$ با حل $n+m$ معادله $\dfrac{\partial \mathcal{L}}{\partial x}=0$ و $\dfrac{\partial \mathcal{L}}{\partial \lambda}=0$ به دست می‌آیند

/ /

در واقع چون به صفحهٔ $h(x)=0$ محدود شده‌ایم شرط لازم (صفر شدن مشتق) را روی همهٔ سیرهای روی صفحه می‌نویسیم که منجر به عمود شدن ∇f بر صفحه می‌شود.

Example

$$\text{Maximize } x_1x_2+x_2x_3+x_1x_3 \quad \text{s.t. } x_1+x_2+x_3=3 \quad \equiv \quad \text{minimize } -x_1x_2-x_2x_3-x_1x_3=f(x) \quad \text{s.t. } x_1+x_2+x_3=3$$

$$\Rightarrow \begin{cases} -(x_2+x_3,\ x_1+x_3,\ x_1+x_2)+\lambda(1\ \ 1\ \ 1)=0 \\ x_1+x_2+x_3=3 \end{cases}$$

$$\Rightarrow \begin{cases} x_2+x_3=\lambda \\ x_1+x_3=\lambda \xrightarrow{+} 2(x_1+x_2+x_3)=3\lambda \Rightarrow 3\lambda=6 \Rightarrow \lambda=2 \\ x_1+x_2=\lambda \xrightarrow{-} x_1=x_2=x_3 \Rightarrow x_1=x_2=x_3=1 \end{cases}$$

End Example

$$\text{Maximize } xyz \quad \Rightarrow \text{minimize } f=-xyz$$

$$\text{s.t. } xy+yz+xz=\frac{c}{2} \qquad c>0$$

$$\nabla^T f+\lambda\nabla^T h=0 \Rightarrow \begin{cases} \begin{array}{c} ① \\ ② \\ ③ \end{array}\begin{pmatrix} -yz \\ -xz \\ -xy \end{pmatrix}+\lambda\begin{pmatrix} y+z \\ x+z \\ x+y \end{pmatrix}=0 \\ xy+yz+xz=\frac{c}{2} \end{cases} \Rightarrow \left.\begin{cases} x① - y② \rightarrow \lambda(x-y)z=0 \\ y② - z③ \rightarrow \lambda(y-z)x=0 \\ z③ - x① \rightarrow \lambda(z-x)y=0 \end{cases}\right) (*)$$

$$①+②+③ \Rightarrow -(xy+yz+zx)+2\lambda(x+y+z)=0 \Rightarrow \underbrace{2\lambda(x+y+z)=\frac{c}{2}}_{(**)} \Rightarrow \lambda\neq 0$$

فرض: $x=0 \xrightarrow{②} \lambda z=0 \xrightarrow{\lambda\neq 0} z=0 \xrightarrow{①} \lambda y=0 \xrightarrow{\lambda\neq 0} y=0$

به همین شکل: $\begin{cases} y=0 \Rightarrow x=z=0 \\ z=0 \Rightarrow x=y=0 \end{cases}$ $\xrightarrow{xy+yz+zx=\frac{c}{2}\neq 0} x\neq 0,\ y\neq 0,\ z\neq 0$

$\Rightarrow (*) \Rightarrow x=y=z \Rightarrow$ قید: $x^2+x^2+x^2=\frac{c}{2} \Rightarrow x=\pm\sqrt{\frac{c}{6}}$

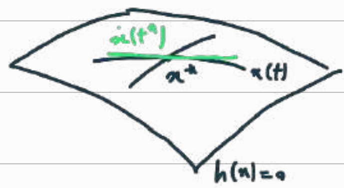
$(**) \Rightarrow \lambda = \pm\frac{1}{2}\sqrt{\frac{c}{6}} \Rightarrow \begin{cases} x=y=z=\pm\sqrt{\frac{c}{6}} \\ \lambda=\pm\frac{1}{2}\sqrt{\frac{c}{6}} \end{cases}$ ← مثبت‌ها با هم و منفی‌ها با هم رخ می‌دهند

end

شرایط لازم و کافی مرتبه دوم: شرط لازم مرتبه دوم: اگر x^* کمینه نسبی f مقید به $h(x)=0$ و رگولار نسبت به قیود باشد و $M=\{y: \nabla h(x^*)y=0\}$ صفحه مماس بر x^* باشد، آنگاه ماتریس $L(x^*)=\nabla^2 f(x^*)+\lambda^T\nabla^2 h(x^*)=F(x^*)+\lambda^T H(x^*)$ بر M نیمه‌مثبت معین است.

$\forall y\in M:\ y^T L(x^*)y \geq 0$

اثبات:



→ f در x^* روی هر مسیر $x(t)$ دلخواه کمینه محلی است

$$\Rightarrow \left.\frac{d^2 f}{dt^2}\right|_{t=t^*} \geq 0 \Rightarrow \frac{d}{dt}\left(\nabla f(x(t))\,\dot{x}(t)\right) \geq 0$$

$$\Rightarrow \dot{x}^T(t)\,\nabla^2 f(x(t))\,\dot{x}(t) + \nabla f(x(t))\,\ddot{x}(t) \geq 0 \quad \text{I}$$

$$h(x)=0 \Rightarrow \underset{1\times m}{\lambda^T}\ \underset{m\times 1}{h(x)}=0 \xrightarrow{\text{مشتق}} \lambda^T \underset{m\times n}{\nabla h(x(t))}\ \dot{x}(t)=0$$

$$\Rightarrow \sum_{i=1}^{m} \lambda_i \nabla h_i(x(t))\, \dot{x}(t) = 0$$

$$\xrightarrow{\text{مشتق}} \dot{x}^T(t) \underbrace{\sum_{i=1}^{m} \lambda_i \nabla^2 h_i(x(t))}_{\lambda^T \nabla^2 h(x(t))\ :\ \text{notation}} \dot{x}(t) + \sum \lambda_i \nabla h_i(x(t))\, \ddot{x}(t) = 0$$

$$\Rightarrow \dot{x}^T \lambda^T \nabla^2 h(x(t))\, \dot{x}(t) + \lambda^T \nabla h(x(t))\, \ddot{x}(t) = 0 \quad \text{II}$$

$$\Rightarrow \text{I} + \text{II} \geq 0 \Rightarrow \dot{x}^T(t)\left[\nabla^2 f(x(t)) + \lambda^T \nabla^2 h(x(t))\right]\dot{x}(t) + \cancel{\left[\nabla f(x(t)) + \lambda^T \nabla h(x(t))\right]}\, \ddot{x}(t)\Big|_{t=t^*} \geq 0$$

0 (شرط لازم مرتبه اول)

$$\Rightarrow \dot{x}^T(t^*)\left[\underbrace{\nabla^2 f(x^*)}_{F} + \lambda^T \underbrace{\nabla^2 h(x^*)}_{H}\right]\dot{x}(t^*) \geq 0$$

$$\Rightarrow \dot{x}^T(t^*)\left[F(x^*) + \lambda^T H(x^*)\right]\dot{x}(t^*) \geq 0$$

← برای مشتق همه مسیرها باید برقرار باشد ← کل صفحه مماس را شامل می‌شود

$$\Rightarrow y^T\left[F(x^*) + \lambda^T H(x^*)\right]y \geq 0 \quad \forall y \in M \quad \Rightarrow y^T L(x^*)\, y \geq 0 \quad \forall y \in M$$

شرط کافی مرتبه دوم: اگر x^* یک نقطه رگولار بر قید $h(x)=0$ باشد به طوری که $h(x^*)=0$ و وجود داشته باشد $\lambda \in R^m$ به طوری که $\nabla f(x^*) + \lambda^T \nabla h(x^*) = 0$ آنگاه اگر $L(x^*)$ بر $M = \{y : \nabla h(x^*)\, y = 0\}$ معین مثبت باشد ($y^T L(x^*)\, y > 0 \ \forall y \in M$) آنگاه x^* یک کمینه محلی اکید برای تابع f مقید به $h(x)=0$ می‌باشد.

Example

Maximize $x_1x_2 + x_2x_3 + x_1x_3 \quad \Rightarrow$ minimize $f = -x_1x_2 - x_2x_3 - x_1x_3$

s.t. $x_1 + x_2 + x_3 = 3$

→ شرط لازم مرتبه اول در شکل های قبل به دست آمد

$$\nabla^T f = -\begin{pmatrix} x_2+x_3 \\ x_1+x_3 \\ x_1+x_2 \end{pmatrix} \Rightarrow \nabla^2 f = -\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

$$\nabla^T h = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \Rightarrow \nabla^2 h = 0 \qquad \Rightarrow \nabla^2 f(x^*) + \underbrace{\lambda^T \nabla^2 h}_{0} = -\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

→ دترمینان کهادهای اصلی مقدم : $0 \quad 1 \quad -(1+1)=-2 \Rightarrow$ نامعین

باید علامت $L(x^*)$ را بر صفحه های بررسی کنیم

$$M = \{y : (1 \ 1 \ 1)y = 0\} = \{y : y_1+y_2+y_3 = 0\}$$

$$y^T L(x^*) y = -[y_1 \ y_2 \ y_3] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = -[y_1 \overbrace{(y_2+y_3)}^{-y_1} + y_2 \overbrace{(y_1+y_3)}^{-y_2} + y_3 \overbrace{(y_1+y_2)}^{-y_3}]$$

$$= -[-y_1^2 - y_2^2 - y_3^2] = y_1^2 + y_2^2 + y_3^2 > 0 \quad \forall y \in M, \ y \neq 0$$

⇐ نقطه به دست آمده از شرط لازم مرتبه اول کمینه f (= بیشینه تابع خواسته شده) را به دست می دهد

end

قیود نامساوی

Minimize $f(x)$

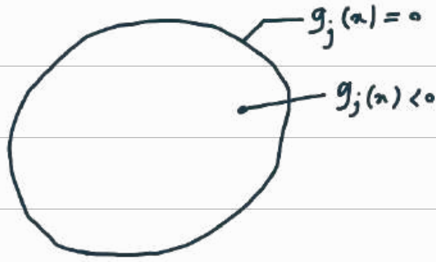
s.t. $\underset{m\times 1}{h(x)} = 0 \quad \underset{p\times 1}{g(x)} \leq 0 \qquad f, g, h \in C'$

پیوسته مشتق پذیر

فعال بودن یا فعال نبودن قید نامساوی : $g_j(x) \leq 0$

$g_j(x) = 0$: قید فعال ← مثل قید تساوی با آن برخورد می شود

$g_j(x) < 0$: قید غیرفعال ← اثری روی بهینه سازی ندارد



← اگر نقطه بهینه خارج مرز باشد Feasible نیست

← اگر نقطه بهینه روی مرز باشد با تبدیل قید تساوی برخورد می‌شود

← اگر نقطه بهینه داخل مرز باشد، مرز محدودیتی روی آن اعمال نمی‌کند

Example

$\min (x-1)^2$

$x>1$ → $x^*=1$: شدنی نیست ⇐ مسئله پاسخ ندارد و نسبت به هر نقطه داخل مرز نقطه‌ای کوچک‌تر نیز وجود دارد ← بدون حضور مرز وجود قید نقشی ندارد

$\min x^2$

$x<1$ → $x^*=0$ ⇐ قید نقشی در به دست آمدن جواب ندارد و تنها Feasible بودن یا نبودن پاسخ را مشخص می‌کند

$\min x^2$

$x\geq 1$ → $x^*=1$ ⇐ قید فعال در به دست آمدن جواب نقش دارد

end

← با داشتن قید نامساوی اکید مسئله را به صورت عادی حل می‌کنیم و تنها Feasible بودن جواب ها را با آن چک می‌کنیم

تعریف: فرض کنید x^* نقطه‌ای است که $g(x^*)\leq 0$ و $h(x^*)=0$ را برآورده می‌کند (شدنی است) و J مجموعه اندیس‌هایی از j باشد که در آن $g_j(x^*)=0$ می‌باشد (قید فعال) آنگاه x^* نسبت به قیدهای فوق رگولار است اگر $\nabla h_i(x^*)$ و $\nabla g_j(x^*)$ به ازای هر $1\leq i\leq m$ و $j\in J$ مستقل خطی باشند.

/ /

قضیه Karush-kuhn-Tucker : اگر x^* یک کمینه نسبی برای مسئله بهینه‌سازی

$$\begin{cases} \text{Minimize } f(x) \\ \quad s.t. \quad h(x)=0 \ , \ g(x)\leq 0 \end{cases}$$

باشد و همچنین x^* یک نقطه رگولار برای قیود مسئله باشد آنگاه $\lambda \in R^m$ و $\mu \in R^p$ و $\mu \geq 0$
وجود دارند به طوری که:

$$\begin{cases} \underset{1\times n}{\nabla f(x^*)} + \underset{1\times m}{\lambda^T}\underset{m\times n}{\nabla h(x^*)} + \underset{1\times p}{\mu^T}\underset{p\times n}{\nabla g(x^*)} = 0 \\ \underset{1\times p}{\mu^T}\underset{p\times 1}{g(x^*)} = 0 \\ h(x^*)=0 \end{cases}$$

شرط لازم مرتبه اول

اثبات: قیود نامساوی فعال مشابه قیود مساوی با ضرایب لاگرانژ مانند لاگرانژین می‌شوند
قیود نامساوی غیرفعال ضریب صفر می‌گیرند و وارد لاگرانژین نمی‌شوند

$$\mu^T g(x^*) = \sum_{j=1}^{p} \mu_j g_j(x^*) = 0$$

قیود فعال: $g_j(x^*)=0$

قیود غیرفعال: $g_j(x^*)<0 \ , \ \mu_j \geq 0 \Rightarrow \mu_j = 0$

← چون x^* کمینه نسبی تابع f با توجه به قیود است می‌توان گفت که کمینه نسبی f با توجه به قیود فعال می‌باشد. بنابراین برای قیود فعال همان رابطه لاگرانژ برقرار است ($\mu_j \geq 0$ برای قیود فعال g_j و $j\in J$ خواهد بود)
برای سایر قیود غیرفعال (که نباید در رابطه لاگرانژ ظاهر شوند) باید داشته باشیم $\mu_j = 0$ ($j \notin J$) برای همین برای همه داریم:

$$\begin{cases} g_j(x^*)<0 \Rightarrow \mu_j = 0 \\ g_j(x^*)=0 \Rightarrow \mu_j \geq 0 \end{cases}$$

که رابطه $\mu^T g(x^*)=0$ این خاصیت را برآورده می‌کند ($\mu \geq 0$)

تفاوت قیود نامساوی فعال و قیود مساوی
این است که $\mu \geq 0$ باید برقرار باشد اما برای λ شرطی وجود ندارد

اثبات $\mu \geq 0$: فرض می‌کنیم $g_k(x^*)=0$ و $\mu_k < 0$

D: صفحه مماس حاصل از سایر قیود فعال در x^*

$$\Rightarrow D = \{ y : \nabla h(x^*) y = 0 ,\ \nabla g_j\, y = 0 \ ,\ j \in J \ ,\ j \neq k \}$$

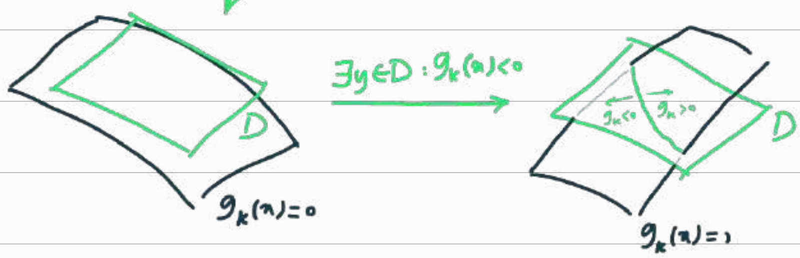
$$\rightarrow x = x^* + \alpha \underbrace{y}_{y \in D} \qquad 0 < \alpha \ll 1 \quad \Rightarrow \exists y \in D : g_k(x) < 0$$

اگر این y وجود نداشته باشد، در هر جهت در D حول x^* در $g_k(x)=0$ می‌مانیم که در

این صورت $\nabla g_k(x^*)$ نیز بر صفحه مماس عمود است ($g_k(x^*)=0$ مماس بر D است)

پس $\nabla g_k(x^*)$ و سایر بردارهای گرادیان سازنده D وابسته خطی می‌شوند و

در این صورت x دیگر نقطه رگولار نیست ✗



$$\Rightarrow g_k(x^* + \alpha y) < 0 \quad \Rightarrow \underbrace{\cancel{g_k(x^*)}}_{0} + \underbrace{\nabla^T g_k(x)(\alpha y)}_{\alpha \ll 1 \Rightarrow \text{این عبارت علامت را تعیین می‌کند}} + O(\alpha) < 0$$

$$\Rightarrow \nabla g_k^T \alpha y < 0 \xrightarrow{\alpha > 0} \nabla g_k^T(x^*) y < 0$$

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu_{-k}^T \nabla g_{-k}(x^*) + \mu_k \nabla g_k(x^*) = 0$$

همه اندیس‌ها به جز k

$$\xrightarrow{\times y} \nabla f(x^*) y + \underbrace{\lambda^T \nabla h(x^*) y}_{y \in D \Rightarrow 0} + \underbrace{\mu_{-k}^T \nabla g_{-k}(x^*) y}_{y \in D,\ \mu_j = 0 \ \Rightarrow 0} + \underbrace{\mu_k}_{<0} \underbrace{\nabla g_k(x^*) y}_{<0} = 0$$

$$\Rightarrow \nabla f(x^*)\,y < 0 \xRightarrow{\text{بسط تیلور}} f(x^*+\alpha y) < f(x^*) \Rightarrow$$ x^* یک کمینه نسبی با توجه به قیود نیست.

روی مرز جایی است و $g_k(x)<0$ در نتیجه Feasible است

$\Leftarrow \mu_k \geq 0$

Example

$$\text{Minimize } \; 2x_1^2 + 2x_1x_2 + x_2^2 - 10x_1 - 10x_2$$
$$\text{s.t. } \; x_1^2 + x_2^2 \leq 5$$
$$3x_1 + x_2 \leq 6$$

$$\Rightarrow \begin{pmatrix} 4x_1 + 2x_2 - 10 & 2x_1 + 2x_2 - 10 \end{pmatrix} + \begin{pmatrix} \mu_1 & \mu_2 \end{pmatrix} \begin{pmatrix} 2x_1 & 2x_2 \\ 3 & 1 \end{pmatrix} = 0$$

$$\Rightarrow \begin{cases} 4x_1 + 2x_2 - 10 + 2\mu_1 x_1 + 3\mu_2 = 0 \\ 2x_1 + 2x_2 - 10 + 2\mu_1 x_2 + \mu_2 = 0 \end{cases}$$

$$\begin{cases} \mu_1(x_1^2 + x_2^2 - 5) = 0 & \mu_1 \geq 0 \\ \mu_2(3x_1 + x_2 - 6) = 0 & \mu_2 \geq 0 \end{cases} \equiv \mu_1(x_1^2+x_2^2-5) + \mu_2(3x_1+x_2-6) = \mu^T g(x) = 0$$

1 هیچ یک از قیدها فعال نباشد

$$\mu_1 = \mu_2 = 0 \Rightarrow \begin{cases} 4x_1 + 2x_2 - 10 = 0 \\ 2x_1 + 2x_2 - 10 = 0 \end{cases} \Rightarrow \begin{cases} x_1 = 0 \\ x_2 = 5 \end{cases} \Rightarrow g_1(0,5) = 25 - 5 = 20 \not< 0$$

2 قید اول فعال باشد و قید دوم فعال نباشد

تمرین: حل معادلات

$$\mu_2 = 0 \Rightarrow \begin{cases} 4x_1 + 2x_2 - 10 + 2\mu_1 x_1 = 0 \\ 2x_1 + 2x_2 - 10 + 2\mu_1 x_2 = 0 \\ x_1^2 + x_2^2 - 5 = 0 \end{cases} \Rightarrow x_1 = 1,\; x_2 = 2,\; \mu_1 = 1 \geq 0$$

$g_2(1,2) = 3+2-6 \leqslant 0$ ✓

3 قید دوم فعال و قید اول غیرفعال باشد

$$\mu_1 = 0 \Rightarrow \begin{cases} 4x_1 + 2x_2 - 10 + 3\mu_2 = 0 \\ 2x_1 + 2x_2 - 10 + \mu_2 = 0 \\ 3x_1 + x_2 = 6 \end{cases} \Rightarrow x_1 = \frac{2}{5}, \quad \mu_2 = \frac{-2}{5} < 0, \quad x_2 = 6 - \frac{6}{5}$$

✗

4 هر دو قید فعال باشند

$$\begin{cases} 4x_1 + 2x_2 - 10 + 2\mu_1 x_1 + 3\mu_2 = 0 \\ 2x_1 + 2x_2 - 10 + 2\mu_1 x_2 + \mu_2 = 0 \\ x_1^2 + x_2^2 - 5 = 0 \\ 3x_1 + x_2 = 6 \end{cases} \Rightarrow \mu_1, \mu_2 < 0$$

✗

← تمرین

← یک مسئله می‌تواند چند کمینه محلی داشته باشد (در چند حالت فعال / غیرفعال بودن قیدها جواب داشته باشیم)

end

مشتق پذیر مرتبه 2

شرط لازم مرتبه دوم: اگر توابع $f, g, h \in C^2$ باشند و x^* نقطه رگولار قیود (فعال) باشد و همچنین x^* یک کمینه نسبی f مقید به قیود باشد، آنگاه ضرایب $\mu \geqslant 0$ و $\mu \in R^p$ و $\lambda \in R^m$ وجود دارند به طوری که

$$L(x^*) = \nabla^2 f(x^*) + \lambda^T \nabla^2 h(x^*) + \mu^T \nabla^2 g(x^*) \longrightarrow \text{notation: } \lambda^T \nabla^2 h = \sum_{i=1}^{m} \lambda_i \nabla^2 h_i, \quad \mu^T \nabla^2 g = \sum_{j=1}^{p} \mu_j \nabla^2 g_j$$

بر صفحه مماس مربوط به قیود فعال در x^* نیمه مثبت معین (PSD) است:

$$\forall y \in M' \quad y^T L(x^*) y \geqslant 0$$

$$M' = \{ y : \nabla h(x^*) y = 0, \ \nabla g_j(x^*) y = 0 \quad \forall j \in J \}$$

← مجموعه قیود نامساوی فعال

/ /

شرط کافی مرتبه دوم: اگر توابع $f,g,h \in C^2$ (مشتق‌پذیر مرتبه 2) و x^* نقطه رگولار قیود باشد، شرط کافی برای این که x^* یک کمینه نسبی اکید برای مسئله بهینه‌سازی مقید (تساوی و نامساوی) باشد این است که وجود داشته باشد $\lambda \in \mathbb{R}^m$ و $\mu \in \mathbb{R}^p$ به طوری که:

$$\mu \geq 0$$

$$\mu^T g(x^*) \geq 0$$

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) = 0$$

و $L(x^*)$ بر M' معین مثبت (PD) باشد یا به عبارتی:

$$\forall y \in M' \quad y^T L(x^*) y > 0$$

Example

ادامه مثال قبل:

$$\begin{cases} x_1^* = 1 \\ x_2^* = 2 \end{cases} \qquad \begin{cases} \mu_1 = 1 \\ \mu_2 = 0 \end{cases}$$

$$L(x^*) = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} + 1 \times \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 2 & 4 \end{bmatrix} \longrightarrow \text{کهادهای اصلی مقدم} : 6 \ , \ 24$$

$\Rightarrow L(x^*) > 0 \Rightarrow$ روی صفحه مماس هم PD است $\Rightarrow$ x^* نقطه کمینه است

End Example

$$\text{Minimize} \quad \frac{1}{2}(x_1^2 + x_2^2)$$

$$\text{s.t.} \quad x_1 + x_2 - 2 = 0$$

$$x_1 \geq 0 \longrightarrow -x_1 \leq 0$$

$$x_2 \geq 0 \longrightarrow -x_2 \leq 0$$

$$\Rightarrow \begin{cases} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ 1 \end{pmatrix} + (\mu_1 \quad \mu_2) \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = 0 \\ \mu_1(-x_1) + \mu_2(-x_2) = 0 \\ \mu_1, \mu_2 \geq 0 \end{cases} \Rightarrow \begin{cases} x_1 + \lambda - \mu_1 = 0 \\ x_2 + \lambda - \mu_2 = 0 \\ -\mu_1 x_1 = 0 \\ -\mu_2 x_2 = 0 \\ x_1 + x_2 - 2 = 0 \end{cases}$$

1 هردو قید نامساوی غیرفعال

$$\mu_1 = \mu_2 = 0 \Rightarrow \begin{cases} x_1 + \lambda = 0 \\ x_2 + \lambda = 0 \\ x_1 + x_2 = 2 \end{cases} \xrightarrow{\oplus} \overbrace{x_1 + x_2}^{2} + 2\lambda = 0 \Rightarrow \lambda = -1 \Rightarrow x_1 = x_2 = 1 > 0 \checkmark$$

$$L(x^*) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + (-1)\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} > 0 \Rightarrow$$ $x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ کمینه است

2 قید اول فعال و قید دوم غیرفعال

$$x_1 = 0, \mu_2 = 0 \xrightarrow{x_1 + x_2 = 2} x_2 = 2 \xrightarrow{x_2 + \lambda = 0} \lambda = -2 \xrightarrow{x_1 + \lambda - \mu_1 = 0} \mu_1 = -2 < 0 \times$$

3 قید اول غیرفعال و قید دوم فعال

$$x_2 = 0, \mu_1 = 0 \xrightarrow{x_1 + x_2 = 2} x_1 = 2 \xrightarrow{x_1 + \lambda = 0} \lambda = -2 \xrightarrow{x_2 + \lambda - \mu_2 = 0} \mu_2 = -2 < 0 \times$$

4 هردو قید فعال

$$x_1 = x_2 = 0 \Rightarrow x_1 + x_2 = 0 \neq 2 \times$$

end

بهینه‌سازی محدب : $\min f(x)$

$x \in \Omega \longrightarrow$ محدب : $x_1, x_2 \in \Omega \Rightarrow \alpha x_1 + (1-\alpha) x_2 \in \Omega$

$\Rightarrow f(\alpha x_1 + (1-\alpha) x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$

نکته: در بهینه‌سازی مقید اگر مجموعه مشخص شده توسط قیود تساوی و نامساوی را Ω بنامیم، اگر مجموعه Ω محدب باشد و همچنین تابع $f(x)$ محدب باشد، ویژگی‌های بیان شده در بهینه‌سازی محدب در حالت وجود قیود تساوی و نامساوی نیز برقرار می‌باشد.

Minimize $f(x)$

$$\Omega \begin{cases} g_j(x) \leq 0 & j = 1, \dots, p \\ h_i(x) = 0 & i = 1, \dots, m \end{cases}$$

اشتراک چند مجموعه محدب، یک مجموعه محدب است $\Leftarrow$ اگر همه قیود محدب باشند Ω محدب می‌شود

دانستیم $\Gamma = \{x : g_j(x) \leq \beta\}$ محدب است اگر $g_j(x)$ محدب باشد $\Leftarrow$ توابع $g_j(x)$ می‌بایست محدب باشند

$h_i(x) = 0$:
- $h_i(x) \leq 0 \Rightarrow$ $h_i(x)$ باید محدب باشد
- و $-h_i(x) \leq 0 \Rightarrow$ $-h_i(x)$ باید محدب باشد $\Rightarrow$ $h_i(x)$ باید مقعر باشد

$\Rightarrow$ $h_i(x)$ هم باید محدب و هم مقعر باشد

$\rightarrow$
- محدب $h_i \Rightarrow \nabla^2 h(x) \geq 0$
- مقعر $h_i \Rightarrow -\nabla^2 h(x) \geq 0$

$\Rightarrow \nabla^2 h(x) = 0 \Rightarrow$ $h(x)$ باید affine باشد $\Rightarrow h(x) = Ax + B$

شرایط لازم و کافی بهینگی (KKT) برای بهینه‌سازی محدب:

$$L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$$
$$= f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{p} \mu_j g_j(x)$$

بردار (x^*, λ^*, μ^*) پاسخ بهینه و ضرایب لاگرانژ بهینه برای مسئله بهینه‌سازی محدب می‌باشند اگر و تنها اگر شرایط زیر برقرار باشد:

$$\begin{cases} h(x^*) = 0 \ , \ g(x^*) \leq 0 & \rightarrow \text{Primal Feasibility} \quad (x: \text{Primal متغیر}) \\ \mu^* \geq 0 & \rightarrow \text{dual Feasibility} \quad (\mu, \lambda: \text{dual متغیرها}) \\ \nabla_x L(x^*, \lambda^*, \mu^*) = \nabla f(x^*) + \lambda^{*T} \nabla h(x^*) + \mu^{*T} \nabla g(x^*) = 0 & \rightarrow \text{Lagrangian optimality} \\ \mu^{*T} g(x^*) = 0 & \rightarrow \text{Complementary Slackness} \end{cases}$$

یکتایی جواب در مسائل Strictly Convex در این حالت هم برقرار است (با رسیدن به اولین جواب می‌دانیم پاسخ global است و نیاز به بررسی حالات دیگر نیست)

* دوگانی (Duality)

Minimize $f_0(x)$

s.t. $f_i(x) \leq 0 \quad i = 1, \dots, m$ $\longrightarrow$ f_0 مقدار بهینه (optimal value): p^*

$h_i(x) = 0 \quad i = 1, \dots, p$

$D = \left(\bigcap_{i=0}^{m} \operatorname{dom} f_i\right) \cap \left(\bigcap_{i=1}^{p} \operatorname{dom} h_i\right)$ $\longrightarrow$ اشتراک دامنه توابع $\rightarrow x \in D \subseteq \mathbb{R}^n$

(در حل بهینه‌سازی نقشی ندارد

ولی باید چک شود جواب در D باشد)

مسئله دوگان (Dual) : مسئله اصلی (Primal) ← مسئله بهینه‌سازی بدون قید + مسئله بهینه‌سازی Convex

← اگر مسئله اصلی non-Convex و مقید باشد می‌توان آن را به یک مسئله بدون قید و یک مسئله Convex تبدیل کرد تا حل ساده‌تر باشد

← حتی برای مسائل Convex این تبدیل می‌تواند مزیت محاسباتی داشته باشد

← این تبدیل می‌تواند به حل توزیع شده مسئله کمک کند

Lagrangian : $L(x, \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x)$

/ /

ممکن است min به صورت حدی وجود داشته باشد ← → infimum (بزرگترین حد پایین)

→ Lagrange dual function : $g(\lambda,\mu) = \inf_{x\in D} L(x,\lambda,\mu)$

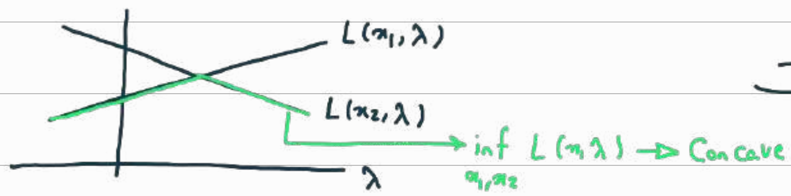
قیدی ایجاد نمی‌کند

$\min_{x>0} x^2$ → جواب ندارد $\inf_{x>0} x^2 = 0$ → جواب دارد (در x غیر شدنی) $\inf e^{-x} = 0 \to x \to \infty$

$$\Rightarrow g(\lambda,\mu) = \inf_{x\in D} L(x,\lambda,\mu) = \inf_{x\in D} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x) \right)$$

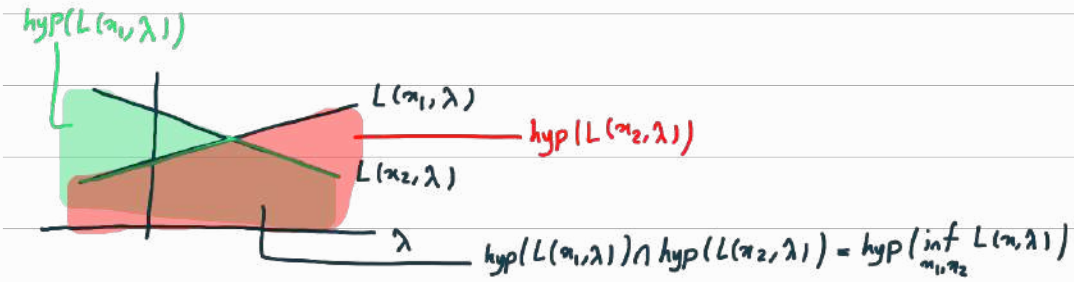
ممکن است این تابع Bounded below نباشد و مسئله جواب نداشته باشد. در این صورت نمی‌توان از این روش استفاده کرد.

- $g(\lambda,\mu)$ یک تابع Concave است:



$g(\lambda,\mu)$ نسبت به λ ، μ خطی (affine) است

مجموعه نقاط زیر یک تابع : Hypograph → $hyp(f) = \{(x,r) \in D \times \mathbb{R} \mid f(x) \geq r\}$



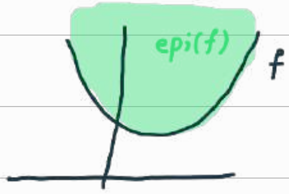
f is Concave iff hyp(f) is Convex

$L(x_i,\lambda,\mu)$ ها affine هستند ⇒ hyp شان محدب است ⇒ اشتراک hyp ها محدب است ⇒ hyp تابع inf محدب است

← inf شان مقعر است

مجموعه نقاط بالای یک تابع : Epigraph $\rightarrow$ $epi(f) = \{(x, r) \in D \times \mathbb{R} \mid f(x) \le r\}$

f is Convex iff epi(f) is Convex



- Lower bound property : $g(\lambda, \mu) \le p^*$ if $\lambda \ge 0$

فرض کنید $\tilde{x}$ یک پاسخ شدنی و $\lambda \ge 0$ باشد ← $f_0(\tilde{x}) \ge L(\tilde{x}, \lambda, \mu) = f_0(\tilde{x}) + \sum_{i=1}^{m} \underbrace{\lambda_i}_{\ge 0} \underbrace{f_i(\tilde{x})}_{\le 0} + \sum_{i=1}^{p} \mu_i \underbrace{h_i(\tilde{x})}_{0}$

$$L(\tilde{x}, \lambda, \mu) \ge \inf_{x \in D} L(x, \lambda, \mu) = g(\lambda, \mu) \Rightarrow g(\lambda, \mu) \le f_0(\tilde{x})$$

از آنجایی که رابطه بالا به ازای هر $\tilde{x}$ شدنی برقرار است بنابراین شامل x^* شدنی و کمینه کننده $f_0(x)$ نیز می‌شود

$$\Rightarrow g(\lambda, \mu) \le f_0(x^*) = p^*$$

The (Lagrange) dual Problem : Maximize $g(\mu, \lambda)$

s.t. $\lambda \ge 0$

g مقعر و شرط $\lambda \ge 0$ محدب است ← مسئله dual محدب است

$g(\lambda, \mu) \le p^*$ ← با بیشینه کردن g به p^* تا حد ممکن نزدیک می‌شویم

← مسئله فوق Convex است و حل آن بهترین حد پایینی برای p^* را به دست می‌دهد

d^* : dual مقدار بهینه مسئله $\Rightarrow$ duality gap $= p^* - d^*$
- $d^* \le p^*$: weak duality
- $d^* = p^*$: strong duality

/ /

weak duality همواره برقرار است و strong duality (حالت خاص weak duality) برای مسئله Convex تحت شرایط سادهای برقرار است و همچنین برای برخی مسائل non-Convex نیز می‌تواند برقرار باشد.

مسئله Primal

Constraint Qualification (CQ) $\xrightarrow{\text{معروف‌ترین}}$ Slater's Constraint Qualification

minimize $f_0(x)$

s.t. $f_i(x) \leq 0 \quad i=1,\dots,m$

$Ax = b$

interior: (غیر مرزی) نقاط درونی

$$\exists \tilde{x} \in \text{int}\, D : f_i(\tilde{x}) < 0 \quad i=1,\dots,m \quad , \quad A\tilde{x} = b$$

در این صورت مسئله Convex اصطلاحاً Strictly Feasible است و شرایط CQ برقرار است.

بنابراین برای مسئله Convex با احراز شرایط CQ، strong duality برقرار خواهد بود $\leftarrow d^* = p^*$

اثبات در کتاب بوید

x^*, λ^*, μ^* به دست آمده در مسئله dual و strong duality لزوماً x^*, λ^*, μ^* مسئله Primal نیستند $\Leftarrow$ شرایط KKT برای مسئله Convex باید بررسی شود

$\lambda^* \geq 0$ ✓Dual Feasibility

جایگذاری λ^*, μ^* به دست آمده

$$g(\lambda,\mu) = \inf_{x \in D} \{L(x,\lambda,\mu)\} \Rightarrow x^* = \underset{x \in D}{\text{Argmin}}\ L(x,\lambda^*,\mu^*)$$ ✓ Lagrange optimality

x^* به دست آمده لزوماً یکتا نیست، در این حالت ممکن است x های دیگری هم به دست بیاید که شروط Primal Feasibility و Complementary slackness را نداشته باشند

← اگر x^* به دست آمده از رابطه فوق شرایط Feasibility و Complementary Slackness را برآورده کند در این صورت پاسخ بهینه مسئله Primal خواهد بود.

یک حالت خاص این است که مسئله Primal جواب بهینه یکتا داشته باشد (مثلاً Strictly Convex باشد). در این صورت حل مسئله dual پاسخ بهینه مسئله Primal را به دست خواهد داد. در غیر این صورت حتماً باید شرایط KKT احراز شود.

پاسخ بهینه مسئله Primal

$$p^* = f_0(x^*) \underset{\text{strong duality}}{=} g(\lambda^*, \mu^*) = \inf_x \left\{ f_0(x) + \sum_{i=1}^{m} \lambda_i^* f_i(x) + \sum_{i=1}^{p} \mu_i^* h_i(x) \right\}$$

$$= f_0(\tilde{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\tilde{x}) + \sum_{i=1}^{p} \mu_i^* h_i(\tilde{x}) \longrightarrow$$ $\tilde{x}$ به دست آمده لزوماً x^* نیست

$\tilde{x}$ شدنی $\Rightarrow$ $$f_0(\tilde{x}) + \sum_{i=1}^{m} \underbrace{\lambda_i^*}_{\geq 0} \underbrace{f_i(\tilde{x})}_{\leq 0} + \sum_{i=1}^{p} \mu_i^* \overbrace{h_i(\tilde{x})}^{0} \leq f_0(\tilde{x})$$

اگر پاسخ مسئله یکتا باشد: $$f(x^*) = f_0(\tilde{x}) = f_0(\tilde{x}) + \underbrace{\sum_{i=1}^{m} \lambda_i^* f_i(\tilde{x})}_{0 \Rightarrow \text{Complementary slackness} \checkmark} + \sum_{i=1}^{p} \mu_i^* \cancel{h_i(\tilde{x})}$$ $\Rightarrow$ $\tilde{x}$ نیز یک پاسخ بهینه برای مسئله Primal خواهد بود

Feasibility $\Rightarrow 0$

اگر پاسخ مسئله یکتا باشد Feasibility و Complementary Slackness باید چک شود تا $\tilde{x}$ به دست آمده پاسخ بهینه مسئله Primal باشد

← در این حالت شروط روی متغیرهای Dual (λ^*, μ^*) هم رعایت شده و در نتیجه این متغیرها برابر ضرایب لاگرانژ بهینه مسئله Primal هستند.

Example

$$\min \sum_{i=1}^{n} f_i(x_i) \qquad f_i(x_i) = \frac{1}{2} c_i x_i^2 \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$
$$\text{s.t.} \quad a^T x = b$$

$$g(\mu) = \min_{x} \left\{ \sum_{i=1}^{n} \frac{1}{2} c_i x_i^2 + \mu (b - a^T x) \right\} = \mu b + \min_{x} \left\{ \sum_{i=1}^{n} \frac{1}{2} c_i x_i^2 - \mu \sum_{i=1}^{n} a_i x_i \right\}$$

$$= \mu b + \min_{x} \left\{ \sum_{i=1}^{n} \left\{ \frac{1}{2} c_i x_i^2 - \mu a_i x_i \right\} \right\} = \mu b + \sum_{i=1}^{n} \min_{x_i} \left\{ \frac{1}{2} c_i x_i^2 - \mu a_i x_i \right\}$$

$$\Rightarrow x_i^* = \frac{a_i \mu}{c_i} \Rightarrow g(\mu) = \mu b + \sum_{i=1}^{n} \left[\frac{1}{2} \frac{a_i^2}{c_i} \mu^2 - \frac{a_i^2}{c_i} \mu^2 \right] = \mu b - \frac{\mu^2}{2} \underbrace{\sum_{i=1}^{n} \frac{a_i^2}{c}}_{t} = b\mu - \frac{1}{2} t \mu^2$$

$$\max_{\mu} \; b\mu - \frac{1}{2} t \mu^2 \Rightarrow \mu^* = \frac{b}{t} \qquad \Rightarrow \underline{x_i^* = \frac{a_i \mu^*}{c_i} = \frac{a_i b}{c_i t}}$$

برای این نوع مسائل که حل بسته دارند معمولاً از روش مستقیم استفاده می‌شود. روش Dual را زمانی که حل الگوریتمی نیاز باشد یا بخواهیم از زیرگرادیان استفاده کنیم به کار می‌بریم.

ارتباط نقطه زینی تابع لاگرانژ و مقادیر بهینه مسئله Primal و Dual:

نقطه زینی ⇐ { نسبت به x min ، نسبت به μ, λ Max } ⟵ می‌توان از به دست آوردن نقطه زینی از روش‌های دیگر حل الگوریتم‌های iterative و gradient based برای حل مسئله بهینه‌سازی استفاده کرد.

نقطه زینی: جفت بردار $x^* \in X$ و $z^* \in Z$ یک نقطه زینی برای تابع φ می‌باشند اگر:

$$\underbrace{\varphi(x^*, z) \le \varphi(x^*, z^*)}_{\text{نسبت به } z \text{ در } x^* \text{ Max}} \le \varphi(x, z^*) \qquad \forall x \in X, \; \forall z \in Z$$

$$\underbrace{\varphi(x^*, z^*) \le \varphi(x, z^*)}_{\text{نسبت به } x \text{ در } z^* \text{ min}}$$

به طور معادل می‌توان گفت $x^* \in X$ و $z^* \in Z$ یک نقطه زینی برای تابع φ است اگر و فقط اگر:

$$\sup_{z \in Z} \varphi(x^*, z) = \varphi(x^*, z^*) = \inf_{x \in X} (x, z^*)$$ ⟵ شرط برای $\forall x$ و $\forall z$ نیاز نیست

لم: اگر جفت (x^*,z^*) یک نقطه زینی برای تابع φ باشد آنگاه تساوی minimax برقرار است:

$$\sup_{z\in Z}\ \inf_{x\in X}\ \varphi(x,z) = \inf_{x\in X}\ \sup_{z\in Z}\ \varphi(x,z)$$ → ترتیب اعمال sup, inf مهم نیست

اثبات:

minimax inequality: $\forall \bar{z}\in Z \quad \varphi(x,\bar{z}) \leqslant \sup_{z\in Z} \varphi(x,z)$

$$\xRightarrow{\inf} \forall \bar{z}\in Z \quad \inf_{x\in X} \varphi(x,\bar{z}) \leqslant \inf_{x\in X}\ \sup_{z\in Z} \varphi(x,z)$$

چون به ازای هر $\bar{z}$ برقرار بود به ازای z سوپریمم کننده هم برقرار است $\Rightarrow \underline{\sup_{\bar{z}\in Z}\ \inf_{x\in X} \varphi(x,\bar{z}) \leqslant \inf_{x\in X}\ \sup_{z\in Z} \varphi(x,z)}$ (I)

در هر شرایطی برقرار است ← $\boxed{\text{minimax inequality}}$

Example

$f(x,y) = x+y \qquad S=\{(x,y) \mid x=0,1,2 \ ,\ y=0,1,2 \ ,\ |x-y|<2\}$

$\sup_y \to y=2 \to \inf_x \xrightarrow{|x-y|<2} x=1 \Rightarrow \inf_x \sup_y f = 1+2=3$ ✓

$\inf_x \to x=0 \to \sup_y \to y=1 \Rightarrow \sup_y \inf_x f = 0+1 = 1$

end

اگر (x^*,z^*) نقطه زینی باشد: $\sup_z \varphi(x^*,z) = \varphi(x^*,z^*) = \inf_x \varphi(x,z^*)$

$$\Rightarrow \inf_x\ \sup_z \varphi(x,z) \leqslant \sup_z \varphi(x^*,z) = \varphi(x^*,z^*) = \inf_x \varphi(x,z^*) \leqslant \sup_z\ \inf_x \varphi(x,z)$$

$$\Rightarrow \inf_{x} \sup_{z} \varphi(x,z) \leq \sup_{z} \inf_{x} \varphi(x,z) \quad \text{II}$$

$$\text{I},\text{II} \Rightarrow \inf_{x} \sup_{z} \varphi(x,z) = \sup_{z} \inf_{x} \varphi(x,z) \quad \blacksquare$$

قضیه: (x^*,z^*) نقطه زینی تابع φ است اگر و فقط اگر تساوی minimax برقرار باشد و x^* و z^* به ترتیب پاسخ مسائل زیر باشند:

شرط لازم و کافی

$$\begin{cases} x^* = \operatorname{Argmin}_{x} \sup_{z} \varphi(x,z) \\ z^* = \operatorname{Argmax}_{z} \inf_{x} \varphi(x,z) \end{cases}$$

اثبات: اگر x^* و z^* پاسخ مسائل بهینه‌سازی فوق باشند داریم:

$$\overbrace{\sup_{z} \inf_{x} \varphi(x,z) = \inf_{x} (x,z^*)}^{z^* = \operatorname{Argmax} \cdots} \leq \varphi(x^*,z^*) \leq \overbrace{\sup_{z} \varphi(x^*,z) = \inf_{x} \sup_{z} \varphi(x,z)}^{x^* = \operatorname{Argmin} \cdots} \quad (*)$$

حال اگر تساوی minimax برقرار باشد، تساوی در سراسر نامعادلات فوق برقرار است

$$\Rightarrow \inf_{x} (x,z^*) = \varphi(x^*,z^*) = \sup_{z} \varphi(x^*,z)$$ $\Rightarrow$ (x^*,z^*) طبق تعریف نقطه زینی تابع φ می‌باشد

اثبات عکس: اگر (x^*,z^*) نقطه زینی تابع φ باشد، در لم فوق ثابت شد تساوی minimax برقرار است. همچنین بر اساس نامعادله $(*)$ تساوی در سراسر نامعادله برقرار خواهد شد که در نتیجه آن x^* و z^* پاسخ مسائل بهینه‌سازی فوق خواهند بود:

$$\Rightarrow \underbrace{\sup_{z} \inf_{x} \varphi(x,z) = \inf_{x} (x,z^*)}_{x^* = \operatorname{Argmin}_{x} \sup_{z} \varphi} = \varphi(x^*,z^*) = \underbrace{\sup_{z} \varphi(x^*,z) = \inf_{x} \sup_{z} \varphi(x,z)}_{z^* = \operatorname{Argmax}_{z} \inf_{x} \varphi}$$

/ /

تمرین: چند تایی (x^*, λ^*, μ^*) یک نقطه زینی برای تابع لاگرانژین $L(x, \lambda, \mu)$ است اگر و فقط اگر x^* و (λ^*, μ^*) به ترتیب پاسخ بهینه مسئله Primal و Dual باشند و همچنین Strong duality برقرار باشد.

اگر نقطه (x^*, λ^*, μ^*) به دست آمده زینی باشد می‌گوییم x^* و (λ^*, μ^*) پاسخ بهینه مسائل Primal و Dual هستند و Strong duality برقرار است.

$$L(x, \lambda, \mu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \mu_i h_i(x)$$

$$\Rightarrow \sup_{\lambda \geq 0, \mu} L(x, \lambda, \mu) = \begin{cases} f_0(x) & \rightarrow \text{Feasible } x: \begin{cases} h_i(x) = 0 \\ f_i(x) \leq 0 \;,\; \lambda \geq 0 \end{cases} \\ \infty & \rightarrow \text{نقض هرکدام از قیود} \end{cases}$$

$$\Rightarrow \inf_{x} \sup_{\lambda \geq 0, \mu} L(x, \lambda, \mu) = p^*$$

به ازای x های Feasible انتخاب می‌شود چون در غیر این صورت این مقدار ∞ می‌شود. در نتیجه حل این مسئله منجر به حل مسئله Primal می‌شود

$$\Rightarrow x^* = \operatorname{Argmin}_{x} \sup_{\lambda \geq 0, \mu} L(x, \lambda, \mu) \rightarrow \text{جواب مسئله Primal}$$

تعریف مسئله dual: $\sup_{\lambda \geq 0, \mu} \inf L(x, \lambda, \mu) = d^*$

$$\Rightarrow (\lambda^*, \mu^*) = \operatorname{Argmax}_{\lambda \geq 0, \mu} \inf_{x} L(x, \lambda, \mu) \rightarrow \text{جواب مسئله dual}$$

اگر Strong duality برقرار باشد آنگاه $p^* = d^*$ و طبق روابط فوق:

$$\inf_{x} \sup_{\lambda \geq 0, \mu} L(x, \lambda, \mu) = \sup_{\lambda \geq 0, \mu} \inf_{x} L(x, \lambda, \mu) \rightarrow \text{تساوی minimax برقرار است}$$

تساوی minimax $\equiv$ strong duality

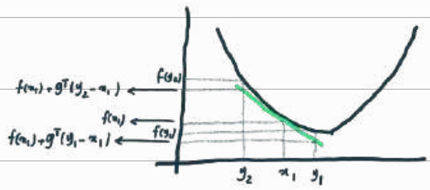
و بنابراین (x^*, λ^*, μ^*) یک نقطه زینی برای تابع لاگرانژین خواهد بود (و برعکس اگر (x^*, λ^*, μ^*) نقطه زینی باشد تساوی minimax که معادل strong duality است برقرار است)

* زیرگرادیان (subgradient)

نوعی تعمیم است به روش گرادیان برای توابع مشتق ناپذیر (عموماً برای بهینه‌سازی توابع Convex استفاده می‌شود چون در همه نقاط برای توابع Convex قابل تعریف است)

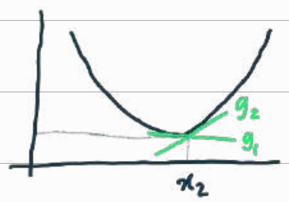
تعریف: بردار g یک زیرگرادیان برای تابع $f: X \subseteq \mathbb{R}^n \to \mathbb{R}$ در x است اگر:

$$f(y) \geq f(x) + g^T(y-x) \quad \forall y \in X$$



→ شیب خط ($g^T \equiv$) معادل زیرگرادیان است (در توابع Convex)

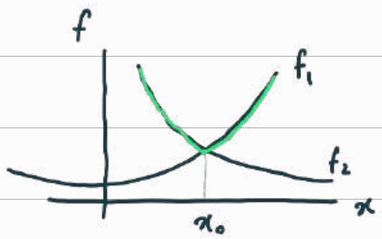
← در توابع Concave شیب بالای خطی است : super gradient



→ تابع f در x_2 مشتق‌پذیر نیست ← هر شیبی بین g_1 و g_2 می‌تواند subgradient باشد

Example

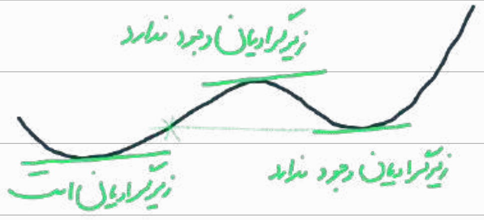
$$f = \max(f_1, f_2)$$



g : زیرگرادیان تابع f

$$\begin{cases} f_1(x) > f_2(x) \Rightarrow g = \nabla f_1(x) \\ f_2(x) > f_1(x) \Rightarrow g = \nabla f_2(x) \\ f_1(x) = f_2(x) \Rightarrow g \in [\nabla f_2(x_0) \quad \nabla f_1(x_0)] \end{cases}$$

end



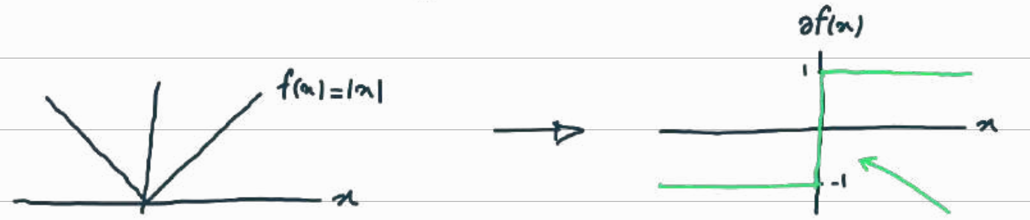
برای توابع Convex زیرگرادیان در تمام نقاط وجود دارد

که ممکن بر توابع محدب در تمام نقاط زیر آن است

زیر دیفرانسیل (subdifferential): به مجموعه زیرگرادیان های تابع f در x زیر دیفرانسیل f در x می گویند و با $\partial f(x)$ نمایش می دهند

$$\partial f(x) = \{ g : f(z) \geq f(x) + g^T(z-x) \quad \forall z \in X \}$$

$$f : X \subseteq \mathbb{R}^n \longrightarrow \mathbb{R} \qquad g \in \mathbb{R}^n$$



کل بازه [-1 1] در صفر جزو زیردیفرانسیل است

با استفاده از روش زیرگرادیان:
- می توان با توابع غیرمشتق پذیر کار کرد
- همگرایی اتفاق می افتد
- ممکن است در هر iteration مقدار تابع هدف کم نشود ولی در نهایت همگرا می شود
- Compatible است با روش های Decomposition و Distributed
- در صورت قید به خوبی عمل می کند ← Projected Subgradient

نیاز داریم زیرگرادیان در تمام دامنه تابع وجود داشته باشد ←

الگوریتم زیرگرادیان برای بهینه سازی مسائل محدب:

محدب

$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$ ⟶ فعلاً حالت بدون قید را در نظر می گیریم

$\Rightarrow$ initial $x_0 \in \mathbb{R}^n$

$$x_{k+1} = x_k - \alpha_k g_k$$

step size ← ; → یک زیرگرادیان تابع f در مرحله k (در x_k)

/ /

به دلیل نقاط مشتق ناپذیر و وجود چند زیرگرادیان در آن‌ها (مثلاً $x=0$ در $|x|$)

در روش steepest Descent همواره در جهت کاهش f حرکت می‌کردیم اما در روش زیرگرادیان اینطور نیست

بهترین جواب در k مرحله اول : $f_k^{best} = \min\limits_{i=1,\dots,k} f(x_i)$ $\longrightarrow$ لزوماً $f(x_k)$ نیست

step size : (1) ثابت : $\alpha_k = \alpha$

(2) طول گام ثابت : $\|x_{k+1} - x_k\| = \gamma \Rightarrow \alpha_k = \dfrac{\gamma}{\|g_k\|}$

$$\|x_{k+1} - x_k\| = \|\frac{\gamma}{\|g_k\|} g_k\| = \gamma$$

(3) جمع مربعات محدود و مجموع نامحدود : $\sum\limits_{k=1}^{\infty} \alpha_k^2 < \infty \quad \sum\limits_{k=1}^{\infty} \alpha_k = \infty$

non Summable

$\lim\limits_{k\to\infty} \alpha_k = 0$

Squared Summable

مثال $\longrightarrow \alpha_k = \dfrac{1}{k}$

انتخاب α_k به این شکل باعث همگرایی به نقطه بهینه می‌شود

(4) جمع و مجموع مربعات نامحدود : $\lim\limits_{k\to\infty} \alpha_k = 0 \quad \sum\limits_{k=1}^{\infty} \alpha_k^2 = \infty \quad \sum\limits_{k=1}^{\infty} \alpha_k = \infty$

مثال $\longrightarrow \alpha_k = \dfrac{1}{k^{0.5}}$

← تعیین همگرایی و همگرایی به مقدار بهینه برای این الگوریتم را بررسی می‌کنیم

مفروضات (همگرایی) : (1) پاسخ بهینه x^* وجود دارد به طوری که : $f(x^*) = f^* = \inf\limits_{x} f(x) > -\infty$

(2) $\|g\|_2 \leq G \quad \forall g \in \partial f \quad \forall x \in \mathbb{R}^n$ $\leftarrow$ norm زیرگرادیان تابع g در همه نقاط محدود باشد

3 فاصله نقطه اولیه (شرایط اولیه) و نقطه بهینه (x^*) محدود باشد: $\|x_0 - x^*\| \le R$

← اگر norm شرایط اولیه محدود انتخاب شود نرم x^* باید محدود باشد

تحلیل همگرایی الگوریتم زیرگرادیان:

$$\|x_{k+1} - x^*\|_2^2 = \|x_k - \alpha_k g_k - x^*\|_2^2 = (x_k - \alpha_k g_k - x^*)^T (x_k - \alpha_k g_k - x^*)$$

$$= \|x_k - x^*\|^2 - 2\alpha_k g_k^T (x_k - x^*) + \alpha_k \|g_k\|^2$$

$$\alpha_k g_k^T (x_k - x^*) = \alpha_k (x_k - x^*)^T g_k$$

تعریف زیرگرادیان: $f(y) \ge f(x) + g^T(y - x) \longrightarrow y \to x^*,\ x \to x_k$

$$\Rightarrow g_k^T(x^* - x_k) \le f(x^*) - f(x_k)$$

$$\Rightarrow \|x_{k+1} - x^*\|^2 \le \|x_k - x^*\|^2 - 2\alpha_k \big(f(x_k) - f(x^*)\big) + \alpha_k^2 \|g_k\|^2$$

روی همه مراحل از 1 تا k، $\sum$ اعمال می‌کنیم:

$$\Rightarrow \sum_{i=1}^{k} \|x_{i+1} - x^*\| \le \sum_{i=1}^{k} \|x_i - x^*\| - 2\sum_{i=1}^{k} \alpha_i \big(f(x_i) - f(x^*)\big) + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2$$

$$\xrightarrow{\|x_i - x^*\| \text{ ها با هم ساده می‌شوند}} \|x_{k+1} - x^*\|^2 \le \|x_1 - x^*\|^2 - 2\sum_{i=1}^{k} \alpha_i \big(f(x_i) - f(x^*)\big) + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2$$

$$\xrightarrow{\|x_{k+1} - x^*\|^2 \ge 0} \|x_1 - x^*\|^2 - 2\sum_{i=1}^{k} \alpha_i \big(f(x_i) - f(x^*)\big) + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2 \ge 0$$

$$\xrightarrow{f(x_i) \ge f_k^{best}} \sum_{i=1}^{k} \alpha_i \big(f(x_i) - f(x^*)\big) \ge \sum_{i=1}^{k} \alpha_i \big(f_k^{best} - f^*\big)$$

$$\Rightarrow \underbrace{\|x_1 - x^*\|^2}_{\text{فرض: } \leq R^2} - 2\sum_{i=1}^{k} \alpha_i \left(f_k^{best} - f^*\right) + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2 \geq 0$$

$$\Rightarrow \left(2\sum_{i=1}^{k} \alpha_i\right)\left(f_k^{best} - f^*\right) \leq R^2 + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2$$

$$\Rightarrow f_k^{best} - f^* \leq \frac{R^2 + \sum_{i=1}^{k} \alpha_i^2 \|g_i\|^2}{2\sum_{i=1}^{k} \alpha_i}$$

فرض: $\|g_i\| \leq G$

1 α ثابت : $f_k^{best} - f^* \leq \dfrac{R^2 + G^2 \sum_{i=1}^{k} \alpha^2}{2\sum_{i=1}^{k} \alpha} = \dfrac{R^2 + kG^2\alpha^2}{2k\alpha}$

$k \to \infty \Rightarrow f_k^{best} - f^* \leq \dfrac{G^2\alpha}{2}$ $\longrightarrow$ الگوریتم همگرا می‌شود ولی در بازه‌ای در اطراف جواب باقی می‌ماند

← α کوچک‌تر این حد را کوچک می‌کند، اما همگرایی را کند می‌کند

2 طول گام ثابت : $f_k^{best} - f^* \leq \dfrac{R^2 + \sum_{i=1}^{k} \frac{\gamma^2}{\cancel{\|g_i\|^2}} \cancel{\|g_i\|^2}}{\sum_{i=1}^{k} \frac{2}{\|g_i\|}\gamma} \leq \dfrac{R^2 + \sum_{i=1}^{k}\gamma^2}{\frac{2}{G}\sum_{i=1}^{k}\gamma} = \dfrac{R^2 + k\gamma^2}{\frac{2k\gamma}{G}}$

فرض: $\|g_i\| \leq G \rightarrow \|g_i\|$

$k \to \infty \Rightarrow f_k^{best} - f^* \leq \dfrac{\gamma G}{2}$ $\longrightarrow$ با γ کوچک‌تر به جواب نزدیک‌تر می‌شویم اما با گام‌های کوچک‌تری به آن می‌رسیم

3 $\sum_{i=1}^{k} \alpha_k = \infty, \ \sum_{i=1}^{k} \alpha_k^2 < \infty$: $f_k^{best} - f^* \leq \dfrac{R^2 + G^2 \sum_{i=1}^{k} \alpha_i^2}{2\sum_{i=1}^{k} \alpha_i}$

$k \to \infty \Rightarrow f_k^{best} - f^* \leq 0 \xrightarrow{f_k^{best} \geq f^*} f_k^{best} = f^*$

← دقیقاً به جواب بهینه همگرا می‌شود

4 $\sum_{i=1}^{k} \alpha_k = \infty, \ \sum_{i=1}^{k} \alpha_k^2 = \infty, \ \lim_{k \to \infty} \alpha_k = 0$ $\rightarrow$ اثبات در Lecture Note های بوید

مثال → $\alpha_k = \frac{1}{k^{0.5}} \Rightarrow \lim_{T\to\infty} \frac{\sum_{k=1}^{T} \frac{1}{k}}{\sum_{k=1}^{T} \frac{1}{k^{0.5}}} = 0$ → با نوشتن این عبارت به T و T+1 می‌توان نزولی بودن آن را دید

$k \to \infty \Rightarrow f_k^{best} \to f^*$ → در این حالت هم به جواب بهینه می‌رسیم

← ویژگی های subgradient :
1. روش مرتبه اول
2. نسبت به روش های مانند نیوتن (مرتبه دوم) کندتر است

مزایا :
1. قابل استفاده برای توابع مشتق ناپذیر
2. محاسبات ساده
3. قابلیت مناسب برای استفاده در الگوریتم های توزیع شده غیرمتمرکز
4. قابلیت استفاده در حالت مقید

روش زیرگرادیان تصویر شده (Projected subgradient method) :

$$\min f(x)$$
$$\text{s.t. } x \in D \qquad D \subseteq \mathbb{R}^n, \text{ Convex}$$ → قید Convex داریم

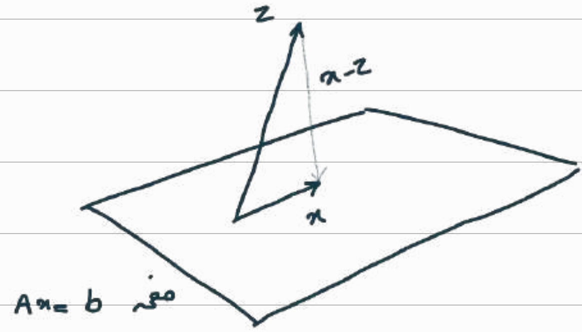
→ $x_{k+1} = Pr_D(x_k - \alpha_k g_k)$ → نقطه ای از مجموعه D که نزدیکترین فاصله را از بردار $x_k - \alpha_k g_k$ دارد انتخاب می‌کنیم

تعریف Projection :
$$\Pi_D(x) = Pr_D(x) = \underset{\tilde{x} \in D}{\text{Arg min}} \underbrace{\|x - \tilde{x}\|}_{\text{معمولاً norm اقلیدسی}}$$

Example

$$\min f(x)$$
$$\text{s.t. } Ax = b$$

$$\Rightarrow x_{k+1} = Pr_{Ax=b}\underbrace{(\underbrace{x_k - \alpha_k g_k}_{z})}_{x}$$



$$\min \ \|x - z\|^2$$
$$\text{s.t.} \ Ax = b$$

$\longrightarrow$ هدف بدست آوردن بردار x به طوری است که فاصله x از بردار z کمینه شود به طوری که x متعلق به صفحه $Ax=b$ باشد

$$\xrightarrow{\text{از تابع لاگرانژ استفاده می‌کنیم}} L(x, \lambda) = \|x - z\|^2 + \lambda^T(Ax - b)$$

$$\frac{\partial L}{\partial x} = 2(x - z) + A^T\lambda = 0 \quad (*) \quad \xrightarrow{Ax} 2(\overset{b}{\overline{Ax}} - Az) + AA^T\lambda = 0 \Rightarrow \lambda = 2(AA^T)^{-1}(Az - b)$$

باید معکوس پذیر باشد $\leftarrow$ A دارای رتبه سطری کامل باشد $\leftarrow$ قیدها مستقل از هم باشند و تناقضی باهم نداشته باشند

$$\xRightarrow{\text{جایگذاری } \lambda \text{ در } (*)} \underset{\downarrow \atop x_{k+1}}{x} = \underset{\downarrow \atop x_k - \alpha_k g_k}{z} - A^T(AA^T)^{-1}(Az - b) \quad (**)$$

$\longrightarrow$ مسئله محدب است پس شرط مرتبه اول لازم و کافی است

$$\Rightarrow x_{k+1} = x_k - \alpha_k g_k - A^T(AA^T)^{-1}(A(x_k - \alpha_k g_k) - b) = x_k - \alpha_k g_k - A^T(AA^T)^{-1}(\cancel{b} - A\alpha_k g_k - \cancel{b})$$
$$= x_k - \alpha_k g_k - A^T(AA^T)^{-1}(-A\alpha_k g_k)$$

این ماتریس به راه حل اصلی اضافه شد

$$\Rightarrow x_{k+1} = x_k - \alpha_k \underbrace{[I - A^T(AA^T)^{-1}A]\, g_k}_{Pr_{N(A)}(g_k)}$$

$$N(A): Ax = 0$$

$$\xrightarrow[z \leftarrow g_k,\ b \leftarrow 0]{(**)} Pr_{N(A)}(g_k) = g_k - A^T(AA^T)^{-1}(Ag_k - 0) = (I - A^T(AA^T)^{-1}A)\, g_k$$

end

همگرایی الگوریتم زیرگرادیان تصویر شده: با فرضیات مشابه و stepsize مشابه:

در هر مرحله الگوریتم

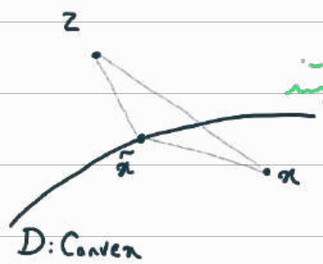
$$\begin{cases} z_{k+1} = x_k - \alpha_k g_k \\ x_{k+1} = \Pi_D(z_{k+1}) \end{cases}$$

اپراتور تصویرسازی (Π_D) یک اپراتور Non expansive است:

$$\|\Pi_D(z) - x\| \le \|z - x\| \quad \forall x \in D$$

چون فاصله z به D در $\tilde{x}$ نباید D را قطع کند

اگر قطع کند نقاط روی قطعه بین دو نقطه عضو D نیستند



D: Convex ⇒ $\angle z\tilde{x}x > 90^\circ$ ⇒ ضلع zx بزرگترین ضلع مثلث است

$x^* \in D$: Convex

$$\|x_{k+1} - x^*\|^2 = \|\Pi_D(z_{k+1}) - x^*\|^2 \le \|z_{k+1} - x^*\|^2$$

⟵ بنابراین از این مرحله به بعد اثبات همگرایی مشابه حالت الگوریتم زیرگرادیان بدون قید خواهد بود و در شرایط مشابه و با فرض Convex بودن مجموعه قیود الگوریتم زیرگرادیان تصویر شده همگرا خواهد شد

* حل مسئله dual به کمک روش گرادیان تصویر شده

Primal: $\min\ f_0(x)$ s.t. $f_i(x) \le 0 \quad i = 1, \dots, m$ ⟶ dual $\max\ g(\lambda)$ s.t. $\lambda \ge 0$

$$L(x, \lambda) = f_0 + \sum_{i=1}^{m} \lambda_i f_i(x) \Rightarrow g(\lambda) = \inf_x L(x, \lambda) \Rightarrow \begin{cases} \max\ g(\lambda) \\ \text{s.t. } \lambda \ge 0 \end{cases} \Rightarrow \begin{cases} \min\ -g(\lambda) \\ \text{s.t. } \lambda \ge 0 \end{cases}$$

قبلاً نشان دادیم $g(\lambda)$ نسبت به λ، Concave است و بنابراین $-g(\lambda)$ تابعی Convex است.

Convex optimization ⇐ از روش زیرگرادیان استفاده می‌کنیم

قیود $f_i(x) \leq 0$ ممکن است ساده نباشند و در نتیجه تصویرسازی روی آنها نیز ساده نباشد. پس الگوریتم زیرگرادیان تصویرشده را روی مسئله dual که قیود affine دارد اعمال می‌کنیم.

← مزیت حل مسئله dual نسبت به primal و مزیت استفاده از الگوریتم گرادیان تصویرشده

$h_k \in \partial(-g(\lambda_k))$

$\lambda_{k+1} = [\lambda_k - \alpha_k h_k]_+ \quad , \quad (x)_+ = Max(x, 0)$ → تصویرسازی روی مقادیر + به این معنی است که اگر مقدار به‌دست آمده مثبت بود خود آن مقدار و در غیر این صورت مقدار صفر لحاظ خواهد شد

← تصویرسازی روی $\lambda \geq 0$ ساده است

$$-g(\lambda) = \underbrace{\sup_x -(f_0(x) + \lambda_1 f_1(x) + \dots + \lambda_m f_m(x))}_{-g(\lambda) = -\inf_x L(x,\lambda) = \sup_x -L(x,\lambda)}$$

$\Rightarrow$ زیرگرادیان $-L(x,\lambda)$ نسبت به λ : $h(-L(x,\lambda)) = -[f_1(x), \dots, f_m(x)]^T \quad \forall x$

$$\sup_x -L(x,\lambda) \rightarrow x^* \quad \Rightarrow \quad h_k = -[f_1(x^*(\lambda_k)), \dots, f_m(x^*(\lambda_k))]^T$$

← به دلیل خطی بودن $L(x,\lambda)$ نسبت به λ زیرگرادیان هم به سادگی محاسبه می‌شود

بهینه سازی غیرمتمرکز (Decentralized Optimization)

فرض می‌کنیم مجموعه‌ای از n عامل / Node به شکل $N=\{1,\dots,n\}$ داریم که هر یک از عامل / Node ها دارای تابع هدف f_i $i=1,\dots,n$ و متغیر تصمیم‌گیری x_i $i=1,\dots,n$ می‌باشند.

هدف حل مسئله های بهینه سازی به اشکال ذیل می‌باشد:

$$\underset{x_1,\dots,x_n,y}{\text{Minimize}} \sum_{i=1}^{n} f_i(x_i, y)$$

⟵ (Complicating Variable) متغیر تداخلی ⟵ y) متغیر Primal) مربوط به هیچ کدام از Agent ها نیست

$$\text{Minimize} \sum_{i=1}^{n} f_i(x_i)$$

$$\text{s.t.} \sum_{i=1}^{n} x_i \le C$$

⟵ قید تداخل ایجاد می‌کند ⟵ باعث می‌شوند نتوان مسئله را Decompose کرد

⟵ با استفاده از روش ضرایب لاگرانژ، λ (متغیر dual) مربوط به هیچ کدام از Agent ها نیست

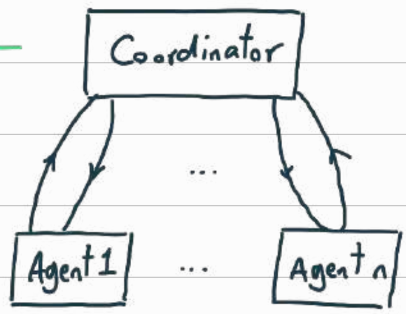
$\vdots$

دلایل عدم حل متمرکز مسائل فوق:

1. گستردگی جغرافیایی و عدم امکان تبادل اطلاعات در حجم گسترده (برای ارسال اطلاعات تابع Agent ها)
2. حجم محاسباتی مسئلهٔ متمرکز و مسئلهٔ مقیاس پذیری (افزایش تعداد Agent ها و در نتیجه افزایش بعد بردار تصمیم‌گیری حل متمرکز مسئله را سخت می‌کند)
3. اطلاعات خصوصی کاربران (Agent ها ممکن است نخواهند اطلاعاتشان را در اختیار Agent بالادستی قرار دهند)

شمای کلی حل مسئله بهینه سازی غیرمتمرکز:

⟵ چون متغیر / قید تداخلی داریم وجود Coordinator نیاز است که ارتباط Agent ها با Coordinator را برقرار کند (راه حل دیگر ایجاد ارتباط بین Agent هاست)



Example

Resource Allocation:

x_i : مقدار تخصیص (کالا/توان/...) به عامل i

$\sum_{i=1}^{n} x_i \leq C$: محدودیت منابع

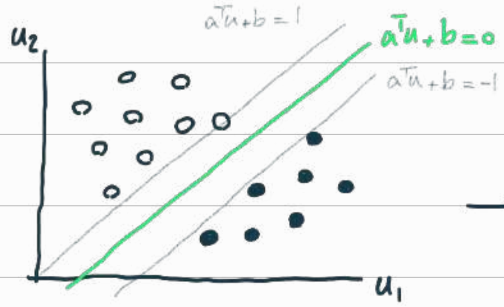
f_i : میزان ترجیح/رضایت (Utility) عامل i از دریافت x_i → Concave معمولاً :

هر Agent دوست دارد مقدار بیشتری از x_i داشته باشد

$$\Rightarrow \text{Maximize } \sum_{i=1}^{n} f_i(x_i)$$
$$\text{s.t. } \sum_{i=1}^{n} x_i \leq C$$

End Example

حل غیرمتمرکز مسئله SVM :



$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$

→ کوچکتر $\|a\|$ ⇐ بزرگتر Margin ⇐ بیشتر generalization

نمی‌خواهیم $\|a\|$ بیش از حد کوچک شود که Data point های بیشتری اشتباه تشخیص داده شوند

Data points: $\{(z_j, y_j),\ j = 1, \dots, p\}$ → p داده

→ label داده : $\bullet\ y_j = -1$, $\circ\ y_j = 1$

→ بردار Feature داده jام : $z_j \in \mathbb{R}^2$

$$\Rightarrow \underset{a,b}{\text{minimize}} \quad \underbrace{\frac{\lambda}{2}\|a\|^2}_{\equiv \text{Maximize Margin}} + \sum_{j=1}^{p} \underbrace{\text{Max}(0, 1-y_j(a^T z_j + b))}_{\text{Violate‌ها از Margin}}$$

Hard Constraint: $y_j(a^T z_j + b) \geq 1 \Rightarrow \begin{cases} y_j = 1 \rightarrow a^T z_j + b \geq 1 \\ y_j = -1 \rightarrow a^T z_j + b \leq -1 \end{cases}$

Soft Constraint

داده‌هایی که درست classify شدند $\rightarrow y_j(a^T z_j + b) \geq 1 \Rightarrow \text{Max}(0, 1-y_j(a^T z_j + b)) = 0$

داده‌هایی که درست classify نشدند $\rightarrow y_j(a^T z_j + b) < 1 \Rightarrow \text{Max}(0, 1-y_j(a^T z_j + b)) = 1-y_j(a^T z_j + b)$

مسئله بهینه‌سازی تعداد داده‌هایی که اشتباه classify می‌شوند و فاصله آن‌ها را کمینه می‌کند

فرض کنید مجموعه داده‌ها در m محل مختلف توزیع شده‌اند

مجموعه داده‌های محل i ام : $\{(z_l, y_l), \; l \in S_i\}$ $\Rightarrow \bigcup_{i=1}^{m} S_i = \{1, \dots, P\}, \; \bigcap_{i=1}^{m} S_i = \phi$

S_i: index set

(مجموعه اندیس داده‌های محل i ام)

$$\Rightarrow \underset{a,b}{\text{minimize}} \sum_{i=1}^{m} \underbrace{\left(\frac{\lambda}{2m}\|a\|^2 + \sum_{l \in S_i} \text{Max}(0, 1-y_l(a^T z_l + b))\right)}_{f_i(x), \; x = (a, b)}$$

$$\Rightarrow \underset{x}{\text{minimize}} \sum_{i=1}^{m} f_i(x)$$

$\longrightarrow$ x: متغیر تصمیم ، f_i: حاوی اطلاعات Private گره i (Node)

هدف : با حل مسائل local و ارتباط با یک Coordinator مسئله اصلی حل شود

end

/ /

* Decomposition in Optimization

→ Convex Optimization 2 - Boyd

• مسائل جدایی‌پذیر :

$$\begin{array}{ll} \underset{x_1, x_2}{\text{Minimize}} & f_1(x_1) + f_2(x_2) \\ \text{s.t.} & x_1 \in C_1,\ x_2 \in C_2 \end{array} \equiv \left(\underset{x_1 \in C_1}{\text{Minimize}}\ f_1(x_1) \right) + \left(\underset{x_2 \in C_2}{\text{Minimize}}\ f_2(x_2) \right)$$

تداخل در متغیر تصمیم‌گیری یا قید وجود ندارد ⇐ به صورت بدیهی جدایی‌پذیر هستند

به جز اپراتور + (جمع) این مسئله برای سایر اپراتورهای غیر نزولی نیز جدایی‌پذیر است

غیر نزولی نسبت به هر کدام از آرگومان‌ها

مثال: اپراتور Max یا Min

$$\to \underset{\substack{x_1 \in C_1 \\ x_2 \in C_2}}{\text{minimize}}\ \Psi(f_1(x_1), f_2(x_2)) \equiv \Psi\left(\underset{x_1 \in C_1}{\text{minimize}}\ f_1(x_1),\ \underset{x_2 \in C_2}{\text{minimize}}\ f_2(x_2)\right)$$

مثلاً این قضیه برای اپراتور - (تفریق) برقرار نیست

Example

$$\min\ x_1^2 - x_2^2$$

$$0 \leq x_1 \leq 1$$

$$0 \leq x_2 \leq 2$$

$$\Rightarrow x_1 \to \min\ ,\quad x_2 \to \text{Max} \Rightarrow \begin{array}{l} \min\ x_1^2 - x_2^2 = -4 \\ 0 \leq x_1 \leq 1 \\ 0 \leq x_2 \leq 2 \end{array}$$

$$\left(\min_{0\le x_1\le 1} x_1^2\right) - \left(\min_{1\le x_2\le 2} x_2^2\right) = -1 \neq -4$$

end

• مسائل با متغیر تداخلی (Complicating / Coupling Variable)

$$\underset{x=(x_1,x_2,y)}{\text{minimize}}\ f(x) = f_1(x_1,y) + f_2(x_2,y)$$ → برای سادگی حالت دو متغیره را در نظر می‌گیریم، به حالت n متغیره قابل تعمیم است

$$\underset{u,w}{\text{minimize}}\ f(u,w) = \min_u \min_w f(u,w)$$

$$\Rightarrow \underset{x_1,x_2,y}{\text{minimize}}\ f(x_1,x_2,y) = \min_y \overbrace{\min_{x_1,x_2} f(x_1,x_2,y)}^{\text{مسئله جدایی‌پذیر نسبت به } x_1, x_2}$$

Coordinator کمینه‌سازی روی y را انجام می‌دهد

Primal Decomposition: ایده این است که ابتدا کمینه‌سازی نسبت به متغیرهای شخصی در سطح پایین (عامل / Node ها) انجام می‌شود سپس حاصل به سطح بالاتر (هماهنگ‌کننده / Coordinator) ارسال می‌شود تا کمینه‌سازی نسبت به متغیر تداخلی انجام شده و در نهایت f کمینه گردد.

Problem 1

$$P_1:\ \underset{x_1}{\text{minimize}}\ f_1(x_1,y) = \varphi_1(y)$$

$$P_2:\ \underset{x_2}{\text{minimize}}\ f_2(x_2,y) = \varphi_2(y)$$

$$\xrightarrow[\text{حل می‌شود}]{\text{حل مسئله در سطح بالاتر}} \underset{y}{\text{minimize}}\ \varphi_1(y) + \varphi_2(y)$$

به این نوع جداسازی Primal Decomposition می‌گویند چرا که $\underbrace{\text{co}}_{\text{Coordinator}}$ یک متغیر Primal را تعیین می‌کند

/ /

← نیاز است Agent ها تابع بهینه (φ_i) به CO ارسال کنند

← ممکن است ارسال φ_i، Privacy عامل i را دچار مشکل کند (در حالتی که متغیر شخصی وجود نداشته باشد عامل ها باید خود f_i را ارسال کنند)

برای این که مسئله ارتباطی بین عامل ها در سطح پایین و CO تسهیل شود می توانیم الگوریتمی پیشنهاد دهیم که صرفاً مقادیر عددی بین عامل ها و CO منتقل شود و همچنین Privacy توابع هدف عامل ها حفظ گردد.

فرض کنیم زیرمسائل P_1 و P_2 توسط عامل ها حل شده است.

الگوریتم Primal Decomposition:

1. شروع با مقدار اولیه y_0 و $k=0$

2. $g_1^k \in \partial\varphi_1(y_k)$, $g_2^k \in \partial\varphi_2(y_k)$ — زیردیفرانسیل

   ← اگر مسئله Convex باشد روش زیرگرادیان تضمین همگرایی دارد

3. به روز رسانی متغیر تداخلی توسط CO به روش زیرگرادیان: $y_{k+1} = y_k - \alpha_k (g_1^k + g_2^k)$
   (α_k به روش های استاندارد بیان شده انتخاب می شود)

   زیرگرادیان تابعی که CO باید کمینه کند

4. $k=k+1$

5. بازگشت به مرحله 1 تا شرط توقف

← subproblem ها باید قابل حل باشند
$\varphi_1 + \varphi_2$ باید Convex باشد تا همگرایی الگوریتم تضمین شود

Dual Decomposition: ایده اصلی معادل سازی مسئله با تعریف متغیرهای جدید و اضافه کردن قید (اجماع / Consensus) است

$$\text{minimize } f(x) = f_1(x_1, y) + f_2(x_2, y) \equiv \text{minimize } f(x) = f_1(x_1, y_1) + f(x_2, y_2)$$
$$\text{s.t. } y_1 = y_2$$

← متغیر Dual ← تداخل به قید منتقل شد

/ /

لاگرانژین مسئله معادل $\longrightarrow$ $L(x_1, y_1, x_2, y_2, \lambda) = f_1(x_1, y_1) + f_2(x_2, y_2) + \lambda^T(y_1 - y_2)$

$$= \underbrace{\left[f_1(x_1, y_1) + \lambda^T y_1 \right]}_{L_1(x_1, y_1, \lambda)} + \underbrace{\left[f_2(x_2, y_2) - \lambda^T y_2 \right]}_{L_2(x_2, y_2, \lambda)}$$

مسئله قابل جداسازی شد

بنابراین برای به دست آوردن تابع dual کافی است زیرمسائل زیر را حل نماییم

$$\text{dual subproblems} \begin{cases} g_1(\lambda) = \inf\limits_{x_1, y_1} L_1(x_1, y_1, \lambda) = \inf f_1(x_1, y_1) + \lambda^T y_1 \\ g_2(\lambda) = \inf\limits_{x_2, y_2} L_2(x_2, y_2, \lambda) = \inf\limits_{x_2, y_2} f_2(x_2, y_2) - \lambda^T y_2 \end{cases}$$

تابع dual : $g(\lambda) = g_1(\lambda) + g_2(\lambda)$

dual problem : $\underset{\lambda}{\text{Maximize}}\ g(\lambda) \equiv \underset{\lambda}{\text{Maximize}}\ g_1(\lambda) + g_2(\lambda)$ $\longrightarrow$ توسط CO در سطح بالاتر حل می‌شود

مشابه حالت قبل به دلیل حفظ Privacy و کاهش حجم اطلاعات ارسالی از ارسال توابع خودداری می‌کنیم

- برای حل الگوریتمی مسئله (جهت حفظ حریم خصوصی عامل‌ها و تسهیل ارتباطات) هر عامل لازم است یک زیرگرادیان از تابع $-g_i(\lambda)$ را محاسبه کند و به CO ارسال کند. همچنین لازم است CO در هر مرحله آخرین مقدار به روز رسانی شده λ را محاسبه و برای عامل‌ها ارسال کند.

g_i : Concave $\longrightarrow$ $-g_i$: Convex $\longrightarrow$ زیرگرادیان دارد

زیرگرادیان $-g_i(\lambda)$:

$$i=1 \Rightarrow -g_1(\tilde{\lambda}) = -\inf_{x_1, y_1} \left(f_1(x_1, x_2) + \tilde{\lambda}^T y_1 \right) = -f(x_1^*(\tilde{\lambda}), y_1^*(\tilde{\lambda})) - \tilde{\lambda}^T y_1^*(\tilde{\lambda})$$

$$\geq -f(x_1^*(\lambda), y_1^*(\lambda)) - \tilde{\lambda}^T y_1^*(\lambda) + \lambda^T y_1^*(\lambda) - \lambda^T y_1^*(\lambda)$$

با تغییر λ

تغییر از x_1^* و y_1^* باعث بزرگ شدن عبارت (= کوچک شدن منفی عبارت) می‌شود

$$= \underbrace{-f_1(x_1^*(\lambda), y_1^*(\lambda)) - \lambda^T y_1^*(\lambda)}_{-g_1(\lambda)} + (-y_1^*(\lambda))(\tilde{\lambda} - \lambda)$$

$$\Rightarrow -g_1(\tilde{\lambda}) \geq -g_1(\lambda) + (-y_1^*(\lambda))(\tilde{\lambda} - \lambda) \longrightarrow$$ تعریف زیرگرادیان

$$\xrightarrow{\text{طبق تعریف}} -y_1^*(\lambda) \in \partial(-g_1(\lambda))$$

$$\xrightarrow{\text{به طریق مشابه}} i=2 \Rightarrow \dots \Rightarrow y_2^*(\lambda) \in \partial(-g_2(\lambda))$$

Agent ها این مقادیر را برای CO ارسال می‌کنند

$$\Rightarrow (y_2^*(\lambda) - y_1^*(\lambda)) \in \partial(-g(\lambda))$$

الگوریتم : Dual Decomposition

1. مقداردهی اولیه λ_0 و $k=0$

2. حل زیرمسئله‌های عامل‌ها:

$$(x_1^*(\lambda_k), y_1^*(\lambda_k)) = \underset{x_1, y_1}{\mathrm{Argmin}} \left(f_1(x_1, y_1) + \lambda_k^T y_1 \right)$$

$$(x_2^*(\lambda_k), y_2^*(\lambda_k)) = \underset{x_2, y_2}{\mathrm{Argmin}} \left(f_2(x_2, y_2) - \lambda_k^T y_2 \right)$$

و ارسال $-y_1^*(\lambda_k)$ و $y_2^*(\lambda_k)$ به CO

3. به‌روزرسانی متغیر Dual توسط CO:

$$\lambda_{k+1} = \lambda_k - \alpha_1 (y_2^*(\lambda_k) - y_1^*(\lambda_k))$$

4. $k = k+1$

5. برو به مرحله 2 تا شرط توقف.

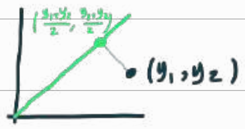
در این روش عامل‌ها بهینه‌سازی ندارند (که بخواهند تابعی را به دست بیاورند) و الگوریتم به صورت کاملاً عملی قابل اجراست، همچنین تضمین همگرایی نیز وجود دارد

← در هر مرحله یک حد پایین از جواب بهینه را خواهیم داشت: $g(\lambda) \le f^*$

← در مراحل الگوریتم پاسخ‌ها لزوماً Feasible نیستند: $y_1 \neq y_2$

اگر بخواهیم در مراحل میانی مقادیر شدنی از متغیرهای y_1 و y_2 داشته باشیم:

$y_1 = y_2$ تصویرسازی در مجموعه $\longrightarrow \quad \hat{y} = \frac{y_1 + y_2}{2}$



در این صورت داریم: $f^* \le f_1(x_1, \hat{y}) + f_2(x_2, \hat{y})$ ← با تغییر y_1 و y_2 به‌کلی از بین می‌رود

می‌توان مقدار x_1 و x_2 را هم توسط بهینه‌سازی برحسب $\hat{y}$ به‌روز کرد: $\hat{x}_i = \underset{x_i}{\mathrm{Argmin}}\ f_i(x_i, \hat{y}) \quad i=1,2$

که باعث بهبود پاسخ و یا عدم تغییر آن خواهد شد و داریم: $f^* \le f_1(\hat{x}_1, \hat{y}) + f_2(\hat{x}_2, \hat{y}) \le f_1(x_1, \hat{y}) + f_2(x_2, \hat{y})$

پاسخ sub optimal

همگرایی الگوریتم dual decomposition و بازیابی متغیرهای Primal:

از آنجا که مسئله dual نسبت به λ محدب می‌باشد، اگر α_k در روش زیرگرادیان به شکل مناسب انتخاب شود در این صورت خواهیم داشت:

$k \to \infty \Rightarrow \lambda_k \to \lambda^*$ ← λ قطعاً همگرا می‌شود

حال اگر مسئله بهینه‌سازی محدب باشد و شرایط CQ برقرار باشد در این صورت duality gap برابر صفر خواهد بود و λ^* همان ضریب لاگرانژ بهینه مسئله خواهد بود.

/ /

$$(x_1^*(\lambda_k), y_1^*(\lambda_k)) = \underset{x_1, y_1}{\text{Argmin}}\ f_1(x_1, y_1) + \lambda_k^T y_1$$

$$(x_2^*(\lambda_k), y_2^*(\lambda_k)) = \underset{x_2, y_2}{\text{Argmin}}\ f_2(x_2, y_2) - \lambda_k^T y_2$$

از آنجا که λ_k به یک λ^* همگرا می‌شود، مقادیر x_1^* و y_1^* و x_2^* و y_2^* نیز همگرا خواهند شد.

اگر مسئله بهینه‌سازی صرفاً Convex باشد در این صورت از آنجا که متغیرهای x_i^*, y_i^* $i=1,2$ لزوماً یکتا نیستند بنابراین الگوریتم لزوماً به مقدار بهینه و شدنی x_i^* و y_i^* همگرا نخواهد شد و به تبع آن شرط $y_1 = y_2$ نیز در نهایت لزوماً برقرار نخواهد شد

شرایط kkt لزوماً برقرار نیست

اما اگر مسئله بهینه‌سازی strictly Convex باشد، آنگاه پاسخ x_i^*, y_i^* $i=1,2$ یکتا می‌باشد و در این صورت الگوریتم به این مقادیر همگرا شده و شرط $y_1 = y_2$ نیز در نهایت برقرار خواهد شد.

جواب Feasible است و قید $y_1 = y_2$ را برقرار می‌کند

مقایسه روش‌های PD و DD:

Primal	dual
✗ محاسبه تابع $\varphi_i(y)$ و سپس محاسبه $g_i^k \in \partial \varphi_i(y_k)$	✓ محاسبه $y_i(\lambda_k)$ از حل بهینه‌سازی با امکان حل عددی
✗ نیاز به Convex بودن φ_iها برای وجود زیرگرادیان و همگرایی	✓ مسئله dual یک مسئله Convex است و λ قطعاً همگرا می‌شود
تعیین یک تغییر بهینه‌سازی تداخلی توسط CO	تنظیم تغییر بهینه‌سازی تداخلی به طور غیرمستقیم و از طریق قیمت مجازی (virtual price)
✓ مقادیر شدنی در همه مراحل الگوریتم	✗ مقادیر شدنی تنها در صورت همگرایی و محدب اکید بودن مسئله یا با تمهیدات بیان شده (تقویت‌سازی)

قیمت : λ ، تولید : y_2 ، مصرف : y_1 → قیمت Virtual است چون پرداخت نمی‌شود و فقط فرضی است (اگر پرداخت می‌شد مانند توابع هدف می‌شد)

⇒ { کاهش تولید → کاهش قیمت → کاهش مصرف
کاهش مصرف → افزایش قیمت → کاهش تولید

/ /

- Decomposition with Constraints

$$\text{minimize} \quad f_1(x_1) + f_2(x_2)$$
$$\text{s.t.} \quad x_1 \in C_1, \quad x_2 \in C_2$$
$$h_1(x_1) + h_2(x_2) \leq 0 \qquad h_i : R^n \to R^p$$

→ اگر در تابع هدف هم تداخل داشته باشیم از ایده‌های بخش قبل استفاده می‌کنیم

p → تعداد قیدها

→ قید تداخلی (مثلاً محدودیت منابع)

Primal Decomposition:

بردار $t \in R^p$ را در نظر می‌گیریم

$$\to \text{minimize} \quad f_1(x_1) + f_2(x_2)$$
$$\text{s.t.} \quad x_1 \in C_1, \quad x_2 \in C_2$$
$$h_1(x_1) \leq t, \quad h_2(x_2) \leq -t_1$$

→ هر متغیر به یک بانک می‌شود t ⇐ بهینه باید پیدا شود t

→ غیر بهینه مسئله قابل جداسازی است ⇐ را CO تعیین می‌کند t

← در حالت عادی این شروط بلای مسئله اصلی کافی هستند اما در نقطه بهینه لازم و کافی می‌شوند

⇒ حل زیرمسئله‌ها توسط عامل‌ها :

$$\begin{cases} P_1: \underset{x_1 \in C_1}{\text{minimize}} \quad f_1(x_1) \quad \to \varphi_1(t) \\ \qquad \text{s.t.} \quad h_1(x_1) \leq t \\ P_2: \underset{x_2 \in C_2}{\text{minimize}} \quad f_2(x_2) \quad \to \varphi_2(t) \\ \qquad \text{s.t.} \quad h_2(x_2) \leq -t \end{cases}$$

حال تنها مسئله این است که مقدار t به صورت بهینه تعیین شود که این کار توسط CO انجام خواهد شد.

$$\underset{t}{\text{minimize}} \quad (\varphi_1(t) + \varphi_2(t))$$

حال مانند حالات قبل به کمک یک الگوریتم عددی مبتنی بر زیرگرادیان نیاز به ارسال توابع φ_1 و φ_2 از طرف عامل‌ها به CO نخواهد بود و فقط مقادیر عددی زیرگرادیان های φ_1 و φ_2 در هر مرحله برای CO ارسال می‌شود و CO بر اساس روش زیرگرادیان مقدار متغیر t را به‌روز کرده و برای عامل‌ها ارسال می‌کند.

/ /

← φ_1 و φ_2 باید برحسب t قابل محاسبه باشند
← φ_1 و φ_2 باید برحسب t محدب باشند ← خواهیم دید در مسائل Convex مشکلی ایجاد نمی‌کند

الگوریتم Primal Decomposition: حل زیرمسائل P_1 و P_2 توسط عامل‌ها و به دست آمدن توابع $\varphi_1(t)$ و $\varphi_2(t)$

1. مقداردهی اولیه t_0 و $k=0$
2. محاسبه $\begin{cases} d_k^1 \in \partial\varphi_1(t_k) \\ d_k^2 \in \partial\varphi_2(t_k) \end{cases}$
3. به‌روزرسانی t_k: $t_{k+1} = t_k - \alpha_t (d_k^1 + d_k^2)$
4. $k = k+1$
5. برو به مرحله 2 تا شرط توقف

برای مسائل محدب (f_i محدب و h_i محدب و مجموعه C_i محدب ، $i=1,2$) خواهیم داشت:

امکان استفاده از duality

$$\varphi_i(t) = \sup_{\lambda_i \geq 0} \inf_{x_i \in C_i} \left(f_i(x_i) + \lambda_i^T (h_i(x_i) \mp t) \right) \qquad i=1,2$$

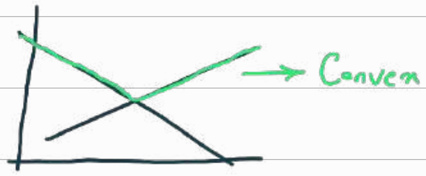
t در حل بهینه‌سازی نسبت به x نقشی ندارد ⇐ x_i^* فقط وابسته به λ است ← sup مجموع توابع خطی نسبت به t باقی می‌ماند

⇐ بیشترین شیب (ضریب t) را می‌خواهیم ← $\mp \lambda_i^*$

$$\Rightarrow d_k^1 = -\lambda_1^*(t_k) \quad , \quad d_k^2 = \lambda_2^*(t_k)$$

→ زیرگرادیان‌ها ضریب لاگرانژ زیرمسئله‌ها هستند ← به حل بسته نیاز نداریم

با حل inf که به t وابسته نیست، φ_1 به صورت sup توابع affine (نسبت به t) به دست می‌آید



$\Leftarrow$ φ نسبت به t محدب است

$\Leftarrow$ برای مسائل محدب محاسبه زیرگرادیان می‌تواند به صورت عددی حل شود در φ_1 های به دست آمده محدب هستند

$$\varphi_1(\tilde{t}) = \sup_{\lambda_1 \geq 0} \inf_{x_1 \in C_1} \left(f_1(x_1) + \lambda_1^T (h_1(x_1)) - \tilde{t}\right) \geq \inf_{x_1 \in C_1} \left(f_1(x_1) + \lambda_1^{*T}(t)(h_1(x_1) - \tilde{t})\right)$$

$\lambda_1^*(\tilde{t}) \xrightarrow{\tilde{t} \to t}$

$$= \inf_{x_1 \in C_1} \left(f_1(x_1) + \lambda_1^{*T}(t)(h_1(x_1) - \tilde{t})\right) + \lambda_1^{*T}(t)t - \lambda_1^{*T}(t)t$$

$$= \inf_{x_1 \in C_1} \left(f_1(x_1) + \lambda_1^{*T}(t)(h_1(x_1) - t)\right) + \lambda_1^{*T}(t)t - \lambda_1^{*T}(t)\tilde{t}$$

$$= \varphi_1(t) + (-\lambda_1^{*T}(t))(\tilde{t} - t)$$

$$\Rightarrow \varphi_1(\tilde{t}) \geq \varphi_1(t) + (-\lambda_1^{*T}(t))(\tilde{t} - t) \xrightarrow{\text{تعریف زیرگرادیان}} -\lambda_1^*(t) \in \partial\varphi_1(t) \xrightarrow{\text{به طور مشابه}} \lambda_2^*(t) \in \partial\varphi_2(t)$$

$$\Rightarrow d_k^1 = -\lambda_1^*(t_k), \quad d_k^2 = \lambda_2^*(t_k)$$

Dual Decomposition:

$$\begin{aligned}
\text{minimize} \quad & f_1(x_1) + f_2(x_2) \\
\text{s.t.} \quad & x_1 \in C_1, \ x_2 \in C_2 \\
& h_1(x_1) + h_2(x_2) \leq 0
\end{aligned}$$

$$\xrightarrow{\text{لاگرانژین}} L(x_1, x_2, \lambda) = f_1(x_1) + f_2(x_2) + \lambda^T(h_1(x_1) + h_2(x_2)) = (f_1(x_1) + \lambda^T h_1(x_1)) + (f_2(x_2) + \lambda^T h_2(x_2))$$

← مشابه حالت قبل شد

حل زیرمسائل توسط عامل ها :
$$\begin{cases} P_1: \text{minimize} \ f_1(x_1)+\lambda^T h_1(x_1) \rightarrow g_1(\lambda) \\ \qquad \text{s.t.} \ x_1 \in C_1 \\ P_2: \text{minimize} \ f_2(x_2)+\lambda^T h_2(x_2) \rightarrow g_2(\lambda) \\ \qquad \text{s.t.} \ x_2 \in C_2 \end{cases}$$

λ را تغییرناپذیر فرض می‌کنیم

$$\Rightarrow g(\lambda) = g_1(\lambda) + g_2(\lambda)$$

حل توسط CO :
$$\begin{cases} \text{maximize} \ g(\lambda) \\ \lambda \geq 0 \end{cases}$$

الگوریتم Dual Decomposition :

1. مقداردهی اولیه λ_0 ، $k=0$
2. حل زیرمسائل توسط عامل ها و به دست آوردن : $\begin{cases} d_k^1 \in \partial g_1(\lambda_k) \\ d_k^2 \in \partial g_2(\lambda_k) \end{cases}$
3. $\lambda_{k+1} = \left[\lambda_k - \alpha_k(d_k^1 + d_k^2)\right]_+$ ← Projected Subgradient
4. $k = k+1$
5. برو به مرحله 2 تا شرط توقف

λ در هر صورت همگرا می‌شود
(مسئله dual نسبت به λ محدب است)
اگر مسئله اصلی محدب باشد به مقدار بهینه همگرا می‌شود

$$g_i : \text{Concave} \Rightarrow -g_i : \text{Convex} \Rightarrow \underset{\text{اثبات}}{\cdots} \Rightarrow d_k^i = -h_i(x_i^*(\lambda_k)) \quad i=1,2$$

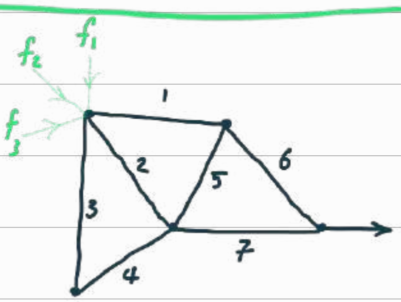
$$\Rightarrow -h_1(x_1^*(\lambda_k)) - h_2(x_2^*(\lambda_k)) \in \partial(-g(\lambda_k))$$

سایر مسائل گفته شده در Dual Decomposition با قید تفاضلی اینجا هم برقرار است.

Primal Decomposition در حالت قید تفاضلی نیاز به حل عملی بهینه سازی پیدا نکرد اما نسبت به Dual Decomposition در این حالت یک تغییر اضافه پیدا کرد و نیازمند حل مسئله بهینه سازی مقید شد (در صورتی که در Dual Decomposition زیرمسئله ها بدون قید هستند)

Example

Network تشکیل شده از تعدادی link :



خروجی

→ هر کدام از Agent ها می‌خواهند flow خود را از شبکه عبور دهند (شبکه مخابراتی / اینترنتی / ...)

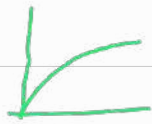
n Agents

n flows

$f_j \geq 0 \quad j = 1, \dots, n \rightarrow n = 3$

Network with m link $\rightarrow m = 7$

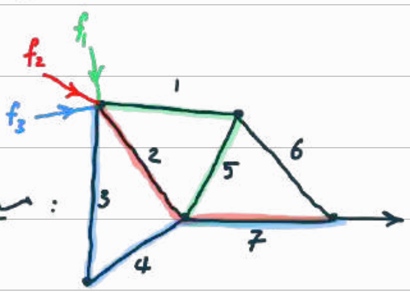
Utility of agent j : $u_j(f_j) \rightarrow$ Strictly Concave and Increasing (مقعر) :



t_i : traffic on link i $\rightarrow \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} = R_{mn} \times \underbrace{\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}}_{f}$

$$[R]_{ij} = \begin{cases} 1 & \text{flow } f_j \text{ Passes over link } i \\ 0 & \text{otherwise} \end{cases}$$

→ مسیرها مشخص هستند :



$$\Rightarrow R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}_{7 \times 3}$$

→ هیچ کس از 6 استفاده نمی‌کند

→ همه از 7 استفاده می‌کنند

$$\Rightarrow t_7 = R\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = f_1 + f_2 + f_3$$

$t \leqslant C$: قید ظرفیت لینک ها

$$\Rightarrow \text{مسئله بهینه سازی} : \begin{cases} \text{Maximize} \ \sum_{j=1}^{n} u_j(f_j) \\ \text{s.t.} \quad Rf \leqslant C \end{cases}$$

$Rf = [r_1\ r_2\ \cdots\ r_n]_{m \times n} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}_{n \times 1}$ ← ستون های ماتریس R

Dual Decomposition: $L(f,\lambda) = \sum_{j=1}^{n} -u_j(f_j) + \lambda^T(RF - C) = \sum_{j=1}^{n} -u_j(f_j) + \lambda^T\left(\sum_{j=1}^{n} r_j f_j\right)$

$$= -\lambda^T C + \sum_{j=1}^{n}\left(-u_j(f_j) + \lambda^T r_j f_j\right)$$

فقط به λ وابسته است ⇐ به CO سپرده می شود

می توان تعبیر قیمت داشته باشد

$$\lambda^T r_j f_j = [\lambda_1 \cdots \lambda_m]\begin{bmatrix} r_{1j} \\ \vdots \\ r_{mj} \end{bmatrix} f_j$$ ⟶ مجموع قیمت استفاده از link ها برای عامل j

$$\Rightarrow \begin{cases} \text{subproblem } j \ (j=1,\dots,n): \ g_j(\lambda) = \inf_{f_j}\left(\lambda^T r_j f_j - u_j(f_j)\right) \\ \text{CO}: \ \underset{\lambda \geqslant 0}{\text{Maximize}} \ g(\lambda), \quad g(\lambda) = \underbrace{-\lambda^T C}_{\text{affine}} + \underbrace{\sum_{j=1}^{n} \inf\left(\lambda^T r_j f_j - u_j(f_j)\right)}_{\text{Concave} \Leftarrow \text{affine تابع} \inf} \end{cases}$$ ⟶ Convex شد

Concave

$$d(\lambda) \in \partial(-g(\lambda)) \Rightarrow d = c - d^1 - d^2 - \cdots - d^n, \quad d^j(\lambda) = r_j f_j^*(\lambda)$$

زیرگرادیان زیرمسئله ها

بدین ترتیب الگوریتم dd به این صورت خواهد بود:

1. $k=0$, λ_0

2. محاسبه زیرگرادیان ها توسط عامل ها :

$$d_k^j = -r_j f_j^*(\lambda_k) \in \partial(-g_j(\lambda_k))$$

هر عامل f بهینه را پیدا و d_k^j را محاسبه و ارسال می‌کند

3. به‌روزرسانی λ_k توسط CO:

$$\lambda_{k+1} = [\lambda_k - \alpha_k(C - r_1 f_1^*(\lambda_k) - \dots - r_n f_n^*(\lambda_k))]_+$$

$$= [\lambda_k - \alpha_k(C - R f^*(\lambda_k))]_+$$

$k = k+1$

4. برو به مرحله 2 تا شرط توقف

end

* Augmented Lagrangian

تعمیمی است از Lagrangian برای حل محدودیت های آن

minimize $f(x)$

s.t. $Ax = b$ $\qquad x \in \mathbb{R}^n, \; A \in \mathbb{R}^{m \times n}, \; f: \mathbb{R}^n \to \mathbb{R}$

Convex

Lagrangian: $L_0(x, y) = f(x) + y^T(Ax - b)$

← نیاز به strictly Convex بودن تابع f جهت بازیابی متغیرهای Primal و همگرایی الگوریتم odd داریم

/ /

برای به دست آوردن g باید $\inf L_0$ حل پذیر باشد ← L_0 باید از پایین محدود باشد

← در حالت خاص‌تر $f(x)$ خطی (affine) باشد $\inf L_0$ نمی‌دهد

نقاط

← به محدود بودن تابع لاگرانژ از پایین برحسب x برای به دست آوردن تابع dual نیاز داریم

← برای رفع این نقاط ضعف Augmented Lagrangian تعریف می‌شود

Augmented Lagrangian : $L_\rho(x,y) = f(x) + y^T(Ax-b) + \underline{(\frac{\rho}{2})\|Ax-b\|_2^2}$ $\rho > 0$ → Penalty Parameter

عبارت اضافه شده همان قیدات ← در صورت وجود جواب Feasible ، پاسخ را تغییر نمی‌دهد

تابع لاگرانژ فوق را می‌توان مربوط به حل بهینه‌سازی زیر دانست :

$$\text{minimize} \quad f(x) + (\tfrac{\rho}{2})\underbrace{\|Ax-b\|^2}_{\text{Feasible } x \Rightarrow 0}$$
$$\text{s.t.} \quad Ax = b$$

جواب بهینه مسئله فوق با جواب بهینه مسئله اولیه یکسان است

عبارت اضافه شده strictly Convex است ← اگر فقط محدب یا خطی باشد اضافه شدن $(\frac{\rho}{2})\|Ax-b\|^2$ کل عبارت را strictly Convex می‌کند و باعث می‌شود inf موجود شود

برای این صورت f می‌تواند توابع پیچیده‌تری هم باشد : $f(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \end{cases}$ → C: Convex → indicator function

برای f به شکل indicator function ، $y^T(Ax-b)$ می‌تواند $-\infty$ شود و $f(x)$ هم $+\infty$ شود و inf لاگرانژین به دست نیاید. ← عبارت Quadratic مانع از این می‌شود که x بی‌نهایت شود

/ /

$\Rightarrow f(x)=0 \Rightarrow x\in C$ یک Quadratic Programming

← به طور کلی f را در شرایط آزادتری می‌توانیم انتخاب کنیم در حالی که از وجود $\inf L_p$ خیالمان راحت باشد

← relax کردن شرایط f

الگوریتم تکراری حل بهینه‌سازی به کمک Augmented Lagrangian

Method of Multipliers :

$$\begin{cases} x^{k+1} = \underset{x}{\text{Arg min}}\ L_p(x, y^k) \\ y^{k+1} = y^k + \rho(Ax^{k+1} - b) \end{cases}$$

← در الگوریتم‌های قبلی x^k از روی y^k به دست می‌آمد ولی در این حالت x^{k+1} از y^k به دست می‌آید و y هم Update می‌شود

step size ← (همان پارامتر پنالتی است)

می‌توان نشان داد } با تعبیر $x^{k+1} = x^k$ ، Lagrangian Optimality برقرار می‌ماند

با در نظر گرفتن step size ثابت الگوریتم به جواب بهینه همگرا می‌شود

← شرایط بهینگی : $Ax^* - b = 0$ → Feasibility

$0 \in \partial f(x^*) + A^T y^*$ → Lagrangian optimality

در هر مرحله با جایگذاری x^{k+1} در تابع لاگرانژ داریم :

$$0 \in \partial_x L_p(x^{k+1}, y^k) = \partial f(x^{k+1}) + A^T y^k + \rho A^T(Ax^{k+1} - b)$$

$$= \partial f(x^{k+1}) + A^T(\underbrace{y^k + \rho A^T(Ax^{k+1} - b)}_{y^{k+1}})$$

$$= \partial f(x^{k+1}) + A^T y^{k+1}$$

← optimality در همه iteration ها برای x^k و y^k شرط برقرار است

با همگرایی الگوریتم شرط Feasibility ($Ax^* - b = 0$) هم برقرار می‌شود ← می‌توان همگرایی الگوریتم را نشان داد

Augment کردن لاگرانژین قابلیت Decompose شدن مسئله را از بین می‌برد

Decomposable

$$\text{minimize} \quad \sum_{i=1}^{n} f_i(x_i) + \left(\frac{\rho}{2}\right) \|Ax-b\|^2 \qquad x=\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$\text{s.t.} \quad Ax=b$$

قابل جداسازی نیست
(چون خطی نیست)

← در الگوریتم MM (Method of Multipliers) متغیر (x^k, y^k) در طول تکرار الگوریتم شرط بهینگی تابع لاگرانژ را برآورده می‌کند و همچنین الگوریتم در نهایت همگرا شده و شرط Feasibility $(Ax^*-b=0)$ در نهایت برقرار خواهد شد. اما نقطه ضعف این روش این است که لاگرانژین Augment شده قابل decompose شدن برای به دست آوردن حل غیرمتمرکز نیست.

برای حل شکل فوق روشی با عنوان Alternating Direction Method of Multipliers (ADMM) به وجود آمد

مرجع مناسب: Survey برتر ← Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers

ADMM: $\quad \text{minimize} \quad f(x)+g(z)$

$$\text{s.t.} \quad Ax+Bz=c$$

$x \in \mathbb{R}^n, z \in \mathbb{R}^m, A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}, c \in \mathbb{R}^p$ ، f, g محدب می‌باشند

در اینجا نیز می‌خواهیم از Augmented Lagrangian استفاده کنیم:

$$L_\rho(x,z,y) = f(x) + g(z) + y^T(Ax+Bz-c) + \left(\frac{\rho}{2}\right)\|Ax+Bz-c\|^2$$

حال الگوریتم ADMM به صورت زیر عمل می‌کند:

$$\begin{cases} x^{k+1} = \underset{x}{\text{Arg min}}\; L_\rho(x, z^k, y^k) \\ z^{k+1} = \underset{z}{\text{Argmin}}\; L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{cases}$$

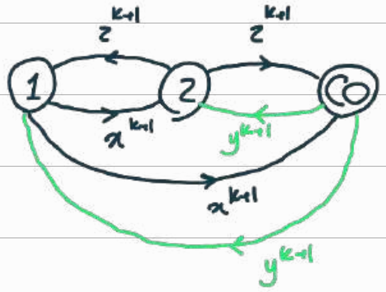
x^{k+1}

x^{k+1}, z^{k+1}

← برخلاف روش های قبلی، بهینه سازی به صورت Sequential انجام می شود و بین Agent ها هم ارتباطی وجود دارد.

Alternating

بدین ترتیب در الگوریتم فوق x و z به صورت ترتیبی یا Alternating به روز می شوند و این فرایند امکان Decompose شدن را به دست می دهد.



→ Agent ها تغییر تقسیم گرادیان را رد و بدل می کنند ولی توابع هدف را اعلام نمی کنند (هر Agent برای min کردن Lp نیاز به دانستن تابع هدف بقیه ندارد)

با تغییر متغیر می توان نحوه نمایش را ساده کرد:

scaled form: $\begin{cases} r = Ax + Bz - C & \rightarrow \text{residual} \\ u = (\frac{1}{\rho})y & \rightarrow \text{scaled dual veriable} \end{cases}$

$$y^T r + (\frac{\rho}{2})\|r\|^2 = (\frac{\rho}{2})\underbrace{\|r + (\frac{1}{\rho})y\|^2}_{(r+(\frac{1}{\rho})y)^T(r+(\frac{1}{\rho})y)} - (\frac{1}{2\rho})\|y\|^2$$

با جایگذاری رابطه فوق در الگوریتم ADMM خواهیم داشت:

$$\begin{cases} x^{k+1} = \underset{x}{\mathrm{Argmin}}\left(f(x) + \frac{\rho}{2}\|Ax + Bz^{k+1} - C + u^k\|^2 \right) \\ z^{k+1} = \underset{z}{\mathrm{Argmin}}\left(g(z) + \frac{\rho}{2}\|Ax^{k+1} + Bz - C + u^k\|^2 \right) \\ u^{k+1} = u^k + Ax^{k+1} + Bx^{k+1} - C \end{cases}$$

→ چون حل min به $g(z)$ و y ربطی ندارند حذف می شوند

→ $y^k = \rho u^k$

/ /

$$r^k = Ax^k + Bz^k - C \Rightarrow u^k = u^0 + \sum_{j=1}^{k} r^j$$

در جزوه بوید مفصل بحث شده

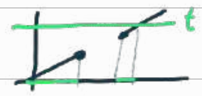
تابع همگرایی:

مفروضات: ① $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ $g: \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$

تابع f و g بسته (closed) و محدب سره (Proper Convex) می‌باشند

closed: سوراخ نداشته باشد!  → شرط ساده‌ایست

به بیان دقیق‌تر تابعی closed است که همه sublevel set های آن بسته باشند

sublevel set: $\{x : f(x) \leq t\}$  ⇒ نیست closed

Proper: $\begin{cases} \exists x \in X & f(x) < \infty \\ \forall x \in X & f(x) > -\infty \end{cases}$ → شرط ساده‌ایست

(منظور میل کردن به $-\infty$ نیست)

در مجموع این سه شرط معادل این است که $\text{epi}(f)$، closed، nonempty و Convex باشد.

$$\text{epi}(f) = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid f(x) \leq t\}$$

برای مثال indicator function این شرایط را دارد: $f(x) = \begin{cases} 0 & x \in C \\ +\infty & x \notin C \end{cases}$ → در روش ADMM حالت $+\infty$ دارد تابع هدف نمی‌شود و

بهینه‌سازی مقید به Quadratic Programming در $x \in C$ می‌شود (بدون Augment کردن تابع هدف affine می‌شود و min نداشت)

2 تابع لاگرانژین مسئله Augment نشده (L_0) دارای یک نقطه زینی است

← مسئله Unaugmented جواب داشته باشد

$$\text{وجود نقطه زینی} \equiv \begin{cases} (x^*, z^*) & \text{Primal optimal} \\ y^* & \text{Dual optimal} \\ \text{Strong Duality} & \end{cases}$$

تحت فرضیات فوق خواهیم داشت:

1 $k \to \infty \Rightarrow r^k \to 0$

2 $k \to \infty \Rightarrow y^k \to y^*$

3 $k \to \infty \Rightarrow f(x^*) + g(z^*) \to p^*$

پاسخ بهینه

لزوماً x^k و z^k به مقادیر بهینه همگرا نخواهند شد!

$$r = Ax^k + Bz^k - C \xrightarrow{k \to \infty} 0$$

← تنها یک زیرفضا مشخص می‌کند که z^k, x^k در آن قرار می‌گیرند

← ممکن است مثلاً به ازای z مشخص ← A و B می‌توانند رتبه کامل ستونی نباشند و فضای پوچ داشته باشند

بی‌نهایت x متناظر وجود داشته باشد

در مراحل اثبات: $s^{k+1} = PA^TB(z^{k+1} - z^k) \xrightarrow{k \to \infty} 0$ optimality (شرط 3) داریم

← ممکن است $z^{k+1} - z^k$ صفر نشود و در فضای پوچ A^TB قرار بگیرد

← شرط optimality همگرایی z^k را نتیجه می‌دهد

ناسازی سیلوستر

← اگر ماتریس A^TB دارای رتبه ستونی کامل باشد (A و B هر دو دارای رتبه ستونی کامل باشند) در این صورت $z^k \to z^{k+1}$ و بنابراین دنباله z^k همگرا خواهد شد. و بنابراین براساس معادله Feasibility ($r \to 0$) دنباله x^k نیز همگرا خواهد شد.

همگرایی روش ADMM در کاربرد: با تعداد تکرار نسبتاً مناسبی به جواب قابل قبول می‌رسد ولی برای همگرایی کامل نیاز به تکرارهای زیادی دارد

← مشابه روش گرادیان

*ADMM / Decomposition Applications - Extentions

- Constrained optimization

$$\begin{aligned} \text{minimize} \quad & f(x) \rightarrow f: \begin{cases} \text{Bounded Below} \\ \text{Bounded Subgradient} \end{cases} \\ \text{s.t.} \quad & x \in C \end{aligned}$$

$$\xrightarrow{\text{ADMM form}} \quad \begin{aligned} \text{minimize} \quad & f(x) + \underbrace{g(z)}_{\text{indicator function of } C \,\to\, z \notin C \Rightarrow g \to \infty} \\ \text{s.t.} \quad & x - z = 0 \quad \leftarrow \text{قید Consensus} \end{aligned}$$

$$\Rightarrow \begin{cases} x^{k+1} := \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{\rho}{2} \| x - z^k + u^k \|_2^2 \right) \\ z^{k+1} := \Pi_C(x^{k+1} + u^k) \\ u^{k+1} := u^k + x^{k+1} - z^{k+1} \end{cases}$$

→ برای range وسیع‌تری از توابع f می‌توان minimization را انجام داد (نسبت به Projected subgradient)

$$\underset{z}{\operatorname{argmin}} \left(g(z) + \frac{\rho}{2} \| x^{k+1} - z + u^k \|_2^2 \right) \overset{z \in C}{=} \underset{z}{\operatorname{argmin}} \left(\| x^{k+1} - z + u^k \|_2^2 \right) = \Pi_C(x^{k+1} + u^k)$$

← برای توابع Quadratic برقرار است

همچنین به عمل Projection روی C نیاز داریم که می‌تواند سخت باشد ← کارهایی برای ساده‌سازی این عمل انجام شده

/ /

- lasso (least absolute shrinkage and selection operator)

regularization : برای مسئله ill-posed (مشاهدات کم و پارامترهای زیاد / sparse) که پارامترها در آن بزرگ می‌شوند regularization انجام می‌شود

lasso مسئله : $$\text{minimize} \quad \underbrace{\tfrac{1}{2}\|Ax-b\|_2^2}_{\text{regression مسئله}} + \underbrace{\lambda \|x\|_1}_{\text{regularization}}$$

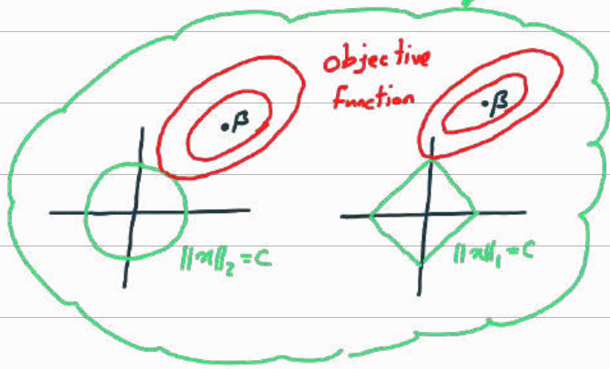
(generalization/shrinkage)

جلوگیری از بزرگ شدن پارامترها ← استفاده از $\|x\|_2$

حذف پارامترهای کم اهمیت ← استفاده از $\|x\|_0$

(Feature Selection)

→ با استفاده از $\|x\|_1$ هر دو انجام می‌شود



ADMM form →
$$\text{minimize} \quad \tfrac{1}{2}\|Ax-b\|_2^2 + \lambda \|z\|_1$$
$$\text{s.t.} \quad x - z = 0$$

$$\Rightarrow L_\rho(x,z,y) = \tfrac{1}{2}\|Ax-b\|_2^2 + \lambda\|z\|_1 + y^T(x-z) + \rho\|x-z\|_2^2$$

$$\Rightarrow \begin{cases} x^{k+1} := (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) & \leftarrow \text{Quadratic Programming} \\ z^{k+1} := S_{\lambda/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} := y^k + \rho(x^{k+1} - z^{k+1}) \end{cases}$$

$\|z\|_1$ مشتق → sign $\Rightarrow$ soft thresholding operator : $$S_k(a) = \begin{cases} a-k & a > k \rightarrow \text{shrinkage} \\ 0 & |a| < k \rightarrow \text{selection} \\ a+k & a < -k \rightarrow \text{shrinkage} \end{cases}$$

↑
optimization

- Consensus Optimization

$$\text{minimize} \sum_{i=1}^{N} f_i(x)$$
→ N agents
→ N blocks of training data
⋮

ADMM form $\longrightarrow$ minimize $\sum_{i=1}^{N} f_i(x_i)$ $\rightarrow g(z)=0 \rightarrow$ می‌توان g(z) را regularization فرض کرد

s.t. $x_i - z = 0$ $\longrightarrow$ برای رسیدن به فرم ADMM، z را اضافه می‌کنیم

x_i: local Veriables; z: global Variable

$$\Rightarrow L_\rho(x,y,z) = \sum_{i=1}^{N} \left(f_i(x_i) + y_i^T (x_i - z) + \frac{\rho}{2} \| x_i - z \|_2^2 \right)$$ $\longrightarrow$ Decompose شده است

$$\Rightarrow \begin{cases} x_i^{k+1} := \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{k^T} (x_i - z^k) + \frac{\rho}{2} \| x_i - z^k \|_2^2 \right) \\ z^{k+1} := \frac{1}{N} \sum_{i=1}^{N} \left(x_i^{k+1} + \frac{1}{\rho} y_i^k \right) \\ y_i^{k+1} := y_i^k + \rho (x_i^{k+1} - z^{k+1}) \end{cases}$$

(z^{k+1}) $\longleftarrow$ حل بسته

Example

Consensus SVM classification:

Data (examples): (a_i, b_i) $i=1,\dots,N$ $a_i \in \mathbb{R}^n$ $b_i \in \{-1, 1\}$

Linear Classifier: $\operatorname{Sign}(a^T w + v)$ w: weight v: offset

margin for ith example: $b_i (a_i^T w + v)$ $\longrightarrow$ می‌خواهیم مثبت باشد

loss for ith example: $l(b_i(a_i^T w + v))$ $\rightarrow$ hinge / logistic / Prohibit / exponential / ...

hinge: $\operatorname{Max}(0, 1 - b_i(a_i^T w + v))$

Choose w, v to minimize $\frac{1}{N} \sum_{i=1}^{N} l(b_i(a_i^T w + v)) + \underbrace{r(w)}_{\text{regularization term } (l_1, l_2, \dots)}$

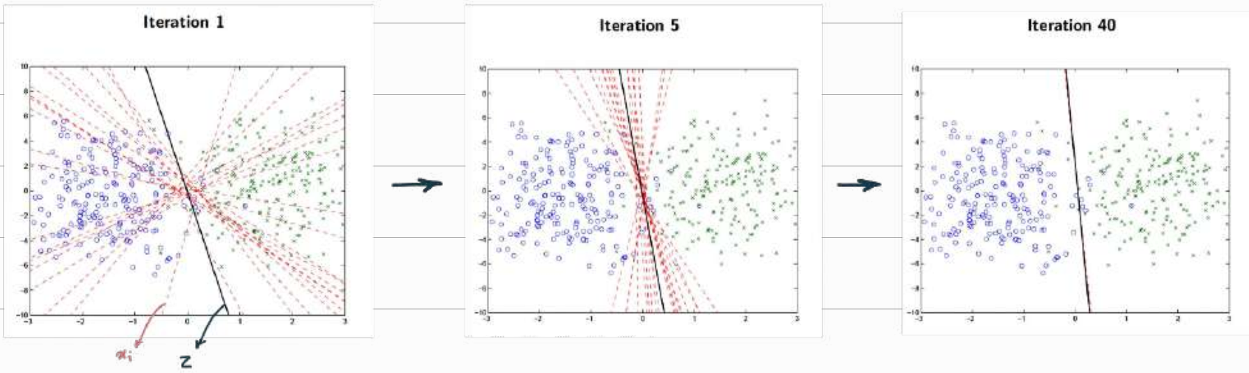
معمولاً از نرم 2 ($l_2 = \|w\|_2$) استفاده می‌شود

/ /

می‌خواهیم داده‌ها را split کنیم و با ADMM مسئله را حل کنیم

hinge loss $l(u) = (1-u)_+$ with l_2 reguarization

$n=2$, $N=400$, 20 groups $\longrightarrow$ در worst case گروه‌ها می‌توانستند فقط یک نوع داده داشته باشند

$\Rightarrow$ 

end

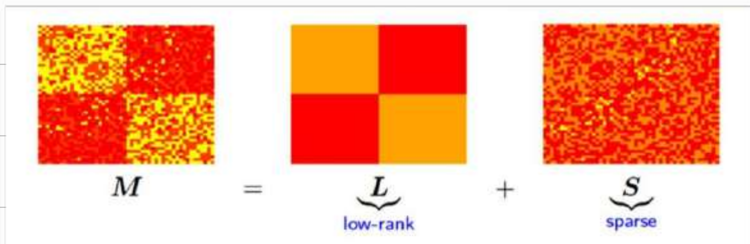
← جزوه / اسلاید Boyd

- other Applications

Network Application

Power system / smart Grid

Machine Learning

$\hookrightarrow$ Robust PCA : 

Nuclear Norm $:= \sum_{i=1}^{n} \sigma_i(L)$

$$\text{minimize } \|L\|_* + \lambda \|S\|_*$$
$$\text{s.t. } L+S=M$$

$\hookrightarrow$ Graphical lasso

- Extensions

Fast ADMM

Distributed ADMM → تعمیم به حالتی که بیش از دو متغیر داشته باشیم

Parallel multiblock ADMM

optimal Parameter Selection for ADMM → انتخاب مناسب

- Federated Learning

← Application ی از Decomposition است که به تازگی مورد استقبال واقع شده

← اولین بار در 2016 مطرح شد

← حل یک مسئله یادگیری ماشین (به طور خاص شبکه عصبی عمیق) برای چندین Dataset محلی

← به دلیل Privacy و Communication Cost نمی‌توانیم داده ها را Exchange کنیم و در یک جا جمع کنیم

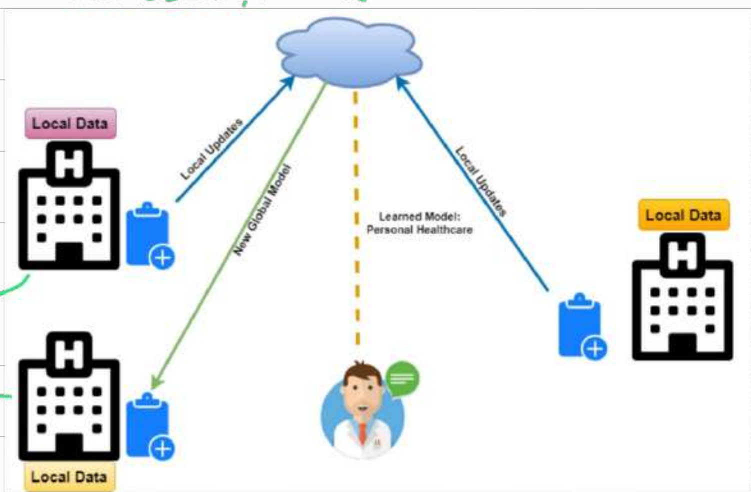
← ابعاد مسئله بسیار بزرگ است (Crowd Sourcing و Crowd Sensing)

← مدل ها به صورت محلی train می شوند

← پارامترهای مدل های محلی به سرور ارسال می شوند

← سرور با گرفتن پارامترها یک مدل جهانی (Global Model) می سازد و آن را با Node ها به اشتراک می گذارد

← مسئله ابعاد وسیعی دارد و Non Convex است



Privacy برای بیمارستان ها مهم است ←

ممکن است بیمارستانی در ساخت مدل شرکت نکند ولی مدل ساخته شده را دریافت کند ←

Core challenges:
- Expensive Communication → ابعاد Message ها تا حد امکان باید کم شوند
- System Heterogenety → User ها یکسان نیستند (توانایی محاسباتی متفاوت یا ...)
- Statistical Heterogenety → داده های local از توزیع های مختلف می آیند (non iid)
- Privacy Concerns

مثلاً مدل پیشنهاد دهنده کلمه بعدی برای هر فرد مختلف است

می تواند تابع پیچیده ی شبکه عصبی باشد

$$\Rightarrow \min_{w \in \mathbb{R}^d} f(w) \quad , \quad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w) \quad , \quad f_i(w) = \ell(\underbrace{x_i, y_i}_{\text{Data point}} ; \underbrace{w}_{\text{Parameter}})$$

$$\text{Client } k : \quad F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w) \quad \xrightarrow{\sum_k n_k(\cdot)} \quad f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

n_k: number of data points of client k

P_k: data set of client k

به دلیل ابعاد مسئله و Non Convex بودن Primal Decomposition مناسب تر است

→ Non Convex Problem

→ No Personal Variable → Update در سطح Agent را ساده می کند ← نیاز به بهینه سازی در هر Node از بین می رود و باعث می شود تنها محاسبه گرادیان نیاز باشد

→ Flexible local Updating

→ low client Participation

⇒ Primal Decomposition

$$g_k = \nabla F_k(w_k) \xrightarrow{f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)} \sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$$

$$\Rightarrow w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

$$\sum_{k=1}^{K} \frac{n_k}{n} w_t = w_t \sum_{k=1}^{K} \frac{n_k}{n} = w_t$$

Updating را می توان به Node ها سپرد: $\forall k \quad w_{t+1}^{k} \leftarrow w_t - \eta g_k$

معمولاً این کار انجام می شود

$$\xrightarrow{\sum_k \frac{n_k}{n}(\cdot)} w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^{k}$$

Algorithm 1 FederatedAveraging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

→ نمی‌توان همه client ها را شرکت داد

→ داده ها را به minibatch ها تقسیم می‌کند
(برای update هر داده صبر نمی‌کنیم!)

→ E بار از هر minibatch عبور می‌کنیم

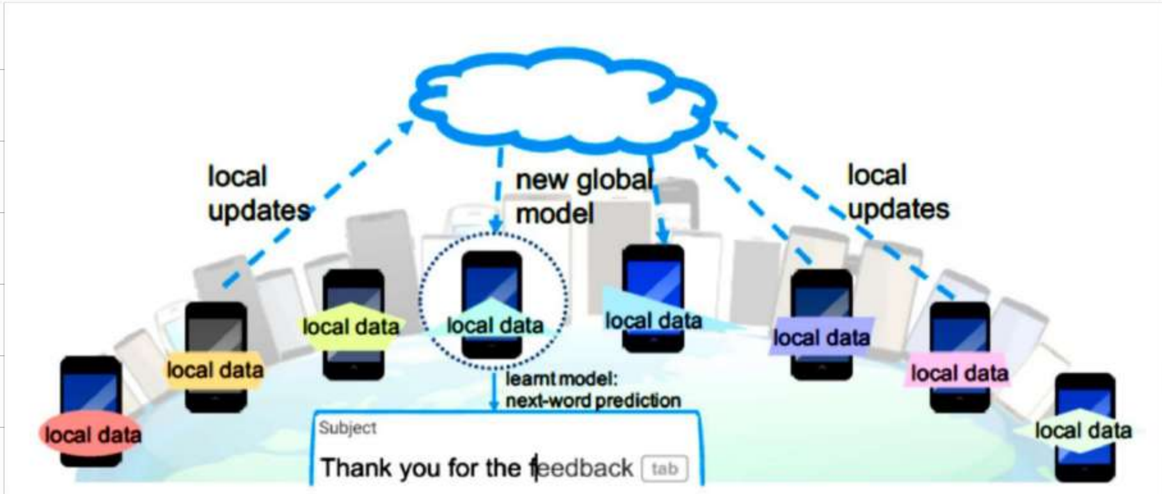
hyperparameter ها توسط ما تعیین می‌شوند

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$ → انتخاب درصدی (C) از client ها
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow$ ClientUpdate(k, w_t)
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$ → Update
  return w to server

→ تعداد local Update ها در هر دور : $u_k = E\frac{n_k}{B}$

next word Prediction (google):



→ Training dataset: 10 M public posts from social network

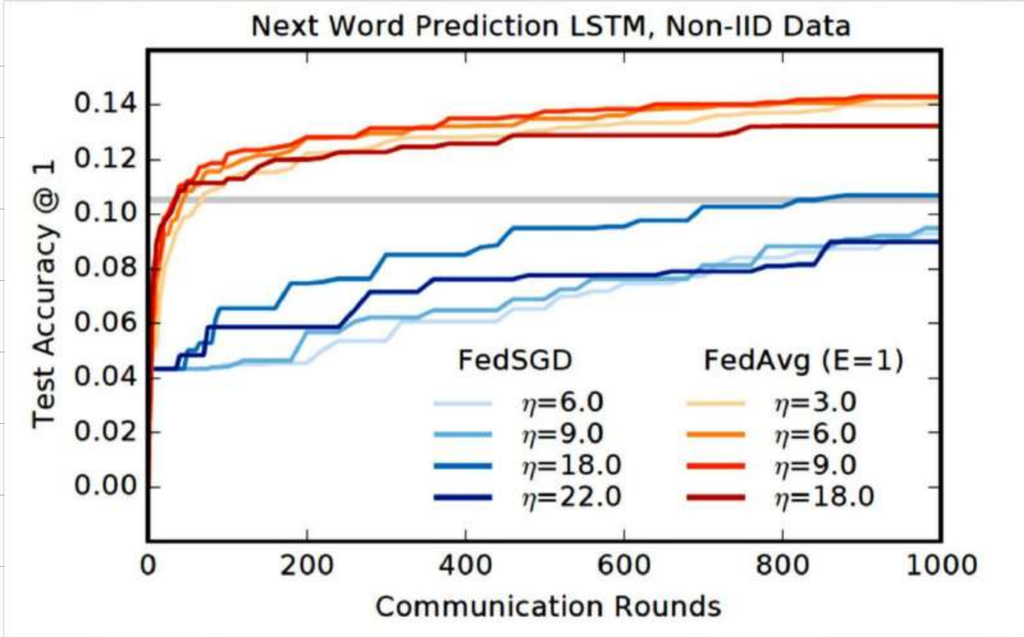
→ The posts grouped by authors, for a total of over 500 000 clients

→ Accuracy: The fraction of data where the highest predicted probability was on the correct next word, out of 10000 possibilities → highest predicted Prob. همان کلمه درست است؟

→ Test set: 1e5 posts from different (non-training) authors

→ Comparing FedAvg with $B = 8$ and $E = 1$ and FedSGD

→ تفاوت اصلی: FedAvg برای
استفاده از minibatch و چند بار عبور از داده ها در FedAvg است



→ Research.google.com

→ Survey های خوبی اخیراً در این زمینه منتشر شده

* بهینه‌سازی توزیع شده‌ٔ غیرهمکارانه

فرض تا اینجا همکاری عامل ها برای حل مسئله بهینه‌سازی بود

Example

$$\max \ \left(x_1 - \frac{1}{8}x_1^2\right) + \left(x_2 - \frac{1}{8}x_2^2\right) \longrightarrow \text{Decomposition قابل}$$

$$\text{s.t.} \quad x_1 + x_2 \leq 7$$

توابع در بازه [0,4] Concave مقعری هستند که قید فعال است (ایده‌آل هر دو Agent مقدار 4 است)

/ /

$\xrightarrow{\text{شرایط KKT}}$ $\begin{cases} -(1-\frac{1}{4}x_1)+\lambda=0 \\ -(1-\frac{1}{4}x_2)+\lambda=0 \\ x_1+x_2=7 \end{cases} \Rightarrow \begin{cases} x_1^*=3.5 \\ x_2^*=3.5 \\ \lambda^*=\frac{1}{8} \end{cases}$

λ برای برآورده شدن قید توسط CO به Agent ها Force می‌شد تا به سمت برآورده کردن قید حرکت کنند (نقش قیمت داشت)

فرض می‌کنیم λ قیمت واقعی باشد و پرداخت شود:

real Payment: λx_i $\Rightarrow$ payoff عامل i: $U_i(x_i,\lambda)=x_i-\frac{1}{8}x_i^2-\lambda x_i$

$\Rightarrow$ در پاسخ بهینه داریم: $U_i(x_i^*,\lambda^*)=x_i^*-\frac{1}{8}x_i^{*2}-\lambda^* x_i^*=1.53125$

فرض می‌کنیم عامل 1، x_1^* را از 3.5 به 3 تغییر دهد ($x_2^*=3.5$ ثابت می‌ماند). در این حالت قید فعال نیست و در نتیجه $\lambda^*=0$

$\Rightarrow U_1(\hat{x}_1,\lambda^*)=U_1(3,0)=\hat{x}_1-\frac{1}{8}\hat{x}_1^2-\lambda^*\hat{x}_1=1.875 > U_1(x_1^*,\lambda^*)=1.53125$

Agent ها انگیزه Deviate (تخطی) از مقدار بهینه را دارند

در حالتی که تابع مصرف محدب باشد و Coupling Constraint فعال باشد، یکی از Agent ها انگیزه دارد تقاضایش را مقداری کم کند تا با صفر شدن Payment سودش افزایش یابد

end

← یک عامل خودخواه (Selfish) می‌تواند انگیزه داشته باشد که از جواب بهینه تخطی کند ← با استفاده از Mechanism Design می‌توان Payment Function را طوری طراحی کرد که با وجود Agent های Selfish به optimal Solution برسیم و Agent ها انگیزه تخطی نداشته باشند

/ /

- Non - Cooperative optimization (Decision Making / Game)

← Agent ها Selfish هستند، هر کدام می‌خواهند سود خود را بیشینه کنند
← بهینگی global (Social optimality) ممکن است حاصل نشود
← برای حل از «نظریه بازی‌ها» استفاده می‌کنیم
← جان نش (ریاضیدان برنده نوبل) مفهوم «تعادل نش» را ابداع کرد

- اجزای بازی (Game Components)

Set of players (Agents): $\mathcal{N} = \{1, \dots, N\}$

strategy sets: $x_n \in \mathcal{X}_n \quad \forall n \in \mathcal{N}$

Objective Functions: $J_n(x_n, \underbrace{x_{-n}}_{\text{تغییرهای تصمیم‌گیری بازیگران بجز n}}) \quad \forall n \in \mathcal{N} \quad \rightarrow \quad x_{-n} = [x_1^T, \dots, x_{n-1}^T, x_{n+1}^T, \dots, x_N^T]$

→ هر Agent تغییر تصمیم‌گیری خودش را تنها برای بهینه‌کردن

تابع هدف خود تعیین می‌کند ولی تابع هدف هر Agent از تغییر تصمیم‌گیری سایر Agent‌ها نیز تأثیر می‌پذیرد (Coupling)

- Best Response Function

$$\Rightarrow BR_n(x_{-n}) = \arg\min_{x_n \in \mathcal{X}_n} J_n(x_n, x_{-n}) \quad \forall n \in \mathcal{N}$$ → تابع بهترین پاسخ

← باید با داشتن x_n مقدارش به دست بیاید (evaluate شود)
← اگر همه به این صورت عمل کنند در نهایت به کجا می‌رسند؟

• Information and Learning Game

در بازی غیرهمکارانه اطلاعات خصوصی وجود دارد (مثل تابع هدف و استراتژی فعلی) ← در بهینه‌سازی همکارانه هم تابع هدف اطلاعات خصوصی عامل‌ها بود

تاریخچه تصمیمات سایر بازیکنان (historical information) در اختیار هر بازیکن هست

← بازی تکرار شونده

← بازیکنان مستقیماً تصمیم دیگران را ببینند یا CO تاریخچه تصمیمات یا aggregation ی از آن را broadcast کند

برای حفظ اطلاعات شخصی

هر اطلاعاتی که Agent با آن بتواند BR را Evaluate کند

← برای گرفتن تصمیم هوشمندانه نیاز است Agent ها از سایرین خبر داشته باشند. در غیر این صورت باید تصادفی تصمیم بگیرند

حالت ساده‌تر را فرض می‌کنیم که بازیکنان تاریخچه تصمیم‌ها را می‌بینند

Agent ها با داشتن تاریخچه، تصمیم بازیکنان را تخمین می‌زنند ← انسان‌ها پیچیده فکر می‌کنند

← تخمین ساده‌ای مثل Naive expectation استفاده می‌شود

← استفاده از روش‌های پیچیده مثل شبکه عصبی مطلوب نیست

Naive Expectation: $\hat{x}_{-i}(t) = x_{-i}(t-1)$ → فرض ثابت ماندن تصمیم سایر Agent ها

← با محاسبه BR و تخمین در هر iteration، یادگیری اتفاق می‌افتد

• Nash Equilibrium

نتیجه بازی منجر به تعادل نش (معرفی شده توسط جان نش) می‌شود ← بازی به تعادل نش همگرا می‌شود

تعادل نش: نقطه‌ای که در آن هیچ کدام از Agent ها نتوانند با تغییر استراتژی‌شان با ثابت بودن استراتژی بقیه به نتیجه بهتر برسد

← هیچ Agent ی انگیزه تغییر ندارد

$(x_1^*, \dots, x_N^*)$ is an NE if $\forall n \in N \quad \forall x_n \in X_n : J_n(x_n^*, x_{-n}^*) \leq J_n(x_n, x_{-n}^*)$

تابع هزینه

/ /

Prisoner's Dillema:

		Prisoner2 Cooperate	Prisoner2 Defect
Prisoner1	Cooperate	3,3	0,5
	Defect	5,0	1,1

→ هر زندانی می‌تواند اعتراف کند یا نکند

بسته به حالت پیش آمده به مقدار نوشته شده

سال زندانی می‌شوند

Social Optimality

گرچه اگر هیچ‌کدام اعتراف نکنند Social welfare بیشینه می‌شود و به نفع هر دو است، ولی اگر خودخواهانه تصمیم‌گیری کنند در هر دو حالت برای هرکدام اعتراف کردن بهتر است. در نتیجه هر دو اعتراف می‌کنند و 3 سال زندان می‌روند!

← خودخواهی منجر به نتیجه بدتر می‌شود

تعادل نش

- Aggregative / Mean-Field game

← می‌خواهیم تعادل بازی غیرهمکارانه را به صورت توزیع شده به دست بیاوریم

objective Function: $J_n(x_n, \sigma(x_N)) \quad \forall n \in N$

$\underbrace{\sigma(x_N)}_{\text{Aggregative / Mean-Field Term}}$

← معمولاً اثرگذاری Agent های دیگر نرم یکسانی دارد و

می‌تواند به این صورت مدل شود

Aggregative Term: $\sigma(x_N) = \frac{1}{N} \sum_{n \in N} x_n$ → می‌تواند هر Aggregation دیگری هم باشد

Best Response: $x_n^{br} = \arg\min_{y \in X_n} J_n\left(y, \frac{1}{N} y + \frac{1}{N} \sum_{\substack{j \in N \\ j \neq n}} x_j\right) \quad \forall n \in N$

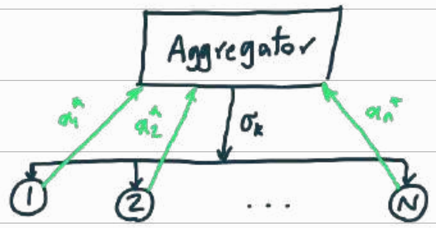
باید اثر تغییر خود Agent در Aggregative Term

در نظر گرفته شود

به این صورت Agent ها دسترسی به historical data ندارند و نیازی هم به آن ندارند. حتی نیاز به دانستن نوع Aggregation نیز ندارند.

یک Coordinator یا Aggregator اطلاعات عددی Aggregative Term را Broadcast می‌کند

Procedure $\Rightarrow$



$$\Rightarrow \begin{cases} \text{Agents:} \quad x_n^*(\sigma_k) = \arg\min_{y \in X_n} J_n(y, \sigma_k) \\ \text{Aggregator:} \begin{cases} \Lambda(\sigma_k) = \sum_{n \in \mathcal{N}} \frac{1}{N} x_n^*(\sigma_k) \\ \sigma_{k+1} = (1-\alpha_k)\sigma_k + \alpha_k \Lambda(\sigma_k) \end{cases} \end{cases}$$

$\rightarrow$ Agent می‌تواند اثر تصمیمش در σ_k را ببیند و بر اساس آن بهینه‌سازی انجام دهد (BR دقیق به دست می‌آید)

الگوریتم $\rightarrow$

initialization

randomly initialize $\sigma_0, k \leftarrow 0$

iteration

for $n \in \mathcal{N}$

$x_n^*(\sigma_k) \leftarrow \arg\min_{y \in \mathcal{X}_n} J_n(y, \sigma_k)$

end

$\Lambda(\sigma_k) \leftarrow \sum_{n \in \mathcal{N}} \frac{1}{N} x_n^*(\sigma_k)$

$\sigma_{k+1} \leftarrow (1-\alpha_k)\sigma_k + \alpha_k \Lambda(\sigma_k)$ $\rightarrow$ Naive تخمین: $\sigma_{k+1} = \Lambda(\sigma_k)$

$k \leftarrow k+1$

$\rightarrow$ این تخمین smooth تر است

چون از اثر تصمیم هر Agent در Aggregative Term در محاسبه BR صرف‌نظر شده دقیقاً به NE نمی‌رسیم

ε-Nash Equilibrium: $J_n\left(x_n^*, \frac{1}{N}\sum_{j \in \mathcal{N}} x_j^*\right) \leq \min_{x_n \in X_n} J_n\left(x_n, \frac{1}{N} x_n + \frac{1}{N}\sum_{\substack{j \in \mathcal{N} \\ j \neq n}} x_j^*\right) + \varepsilon \quad \forall n \in \mathcal{N}$

به نقطه‌ای در شعاعی اطراف NE می‌رسیم

نتیجه به اندازه حداکثر ε برای هر Agent می‌تواند بدتر باشد

هرچه N بزرگتر باشد اثر x_n در σ کمتر و در نتیجه ε کوچکتر می‌شود

فرض های همگرایی : • α_k غیرصعودی ، $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$

• $J_i(y,\sigma)$ ها Uniformly Lipscitz Continuous باشند :

$$J_i(y,\sigma) - J_i(\tilde{y},\tilde{\sigma}) \leqslant L\left(\|y-\tilde{y}\| + \|\sigma-\tilde{\sigma}\|\right)$$

شرط کافی : مشتق پذیر و دارای مشتق محدود باشند

• BR ها Non-Expansive باشند (لیپشیتز با ضریب 1) :

$$x_i^*(\sigma) - x_i^*(\tilde{\sigma}) \leqslant \|\sigma - \tilde{\sigma}\|$$

برای پارامترهای تابع هدف شرط impose می کند

نتایج همگرایی : • الگوریتم همگرا می شود :

$$\Lambda(\sigma) = \frac{1}{N}\sum_{i=1}^{N} x_i^*(\sigma) = \sigma$$

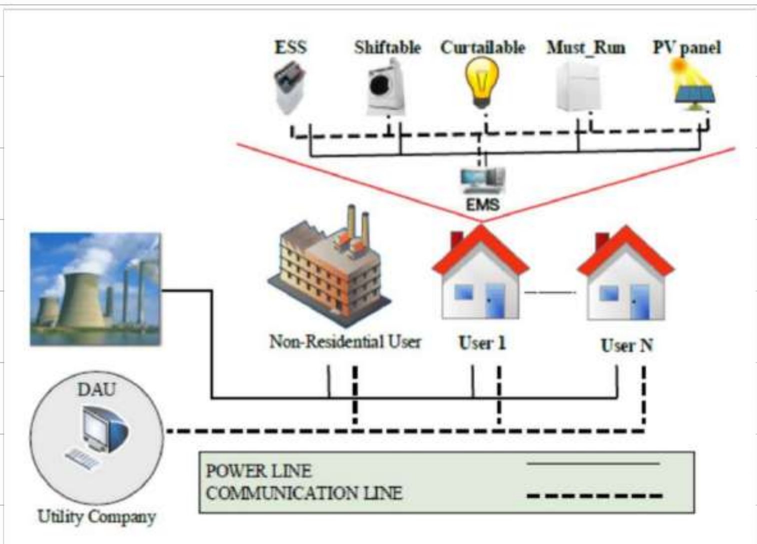
مقدار σ در مراحل بعد تغییر نخواهد کرد

$\left((1-\alpha_k)\sigma + \alpha_k\sigma = \sigma\right)$

• نتیجه Equilibrium ϵ_N-Nash است

• ϵ_N رابطه معکوس با N دارد ← برای مسائل با ابعاد بالا بهتر است !

• Application : Energy Management



→ Battery

→ Photovoltaic

→ Load

Battery soc (state of the charge) dynamic:

توان شارژ　　　توان دشارژ

$$SoC_n^t = SoC_n^{t-1} + \frac{\alpha_{1,n}}{\beta_n} p_{ch,n}^t - \frac{\alpha_{2,n}}{\beta_n} p_{dch,n}^t$$

$$SoC_n^{min} \le SoC_n^t \le SoC_n^{max} \qquad \forall t \in \mathcal{T} = \{1, ..., T\}$$

$$0 \le p_{ch,n}^t \le p_{ch,n}^{\max}, 0 \le p_{dch,n}^t \le p_{dch,n}^{\max}.$$

Battery Degradation Cost:

$$g(p_{b,n}^t) = a|p_{b,n}^t|^2 + b|p_{b,n}^t| + c$$

$$p_{b,n}^t = p_{ch,n}^t - p_{dch,n}^t \longrightarrow$$ در هر لحظه یکی از توان شارژ یا دشارژ صفر است

توان فتوولتاییک ارتباط مستقیم با توان انرژی خورشیدی دارد ← توان انرژی خورشیدی باید پیش بینی شود ← نیازمند پیش بینی درست وضعیت آب و هوا هستیم

$$\Rightarrow P_{PV,n}^t \quad \forall t \in T \longrightarrow$$ به دلیل خطای پیش بینی آب و هوا خطا خواهیم داشت

/ /

/ /

/ /

/ /

/ /

/ /

/ /

/ /

/ /

/ /

/ /

/ /

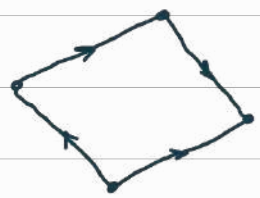
/ /

/ /

/ /

/ /

* Distributed Optimization



در غیاب Coordinator، اجماع کار آن را انجام می‌دهد

هدف کمینه کردن مجموع توابع هدف m عامل است که براساس یک توپولوژی متغیر با زمان (یا ثابت) باهم در ارتباط می‌باشند.

هدف عامل‌ها این است که به صورت همکارانه مسئله بهینه‌سازی مقید زیر را حل کنند:

$$\underset{x}{\text{Minimize}} \sum_{i=1}^{m} f_i(x)$$

$$x \in X = \bigcap_{i=1}^{m} X_i$$

که در آن $f_i : R^n \to R$ تابعی Convex است و اطلاعات شخصی عامل i محسوب می‌شود. $X_i \in R^n$ یک مجموعه بسته و Convex قید محلی عامل i ام می‌باشد و همچنین عامل‌ها به اطلاعات مجموعه مشترک $X = \bigcap_{i=1}^{m} X_i$ دسترسی دارند.

چون گراف strongly Connected است اگر هر Node اطلاعات مجموعه‌اش را با همسایه‌ها به اشتراک بگذارد این دسترسی حاصل می‌شود

مقدار بهینه مسئله فوق را با f^* نمایش می‌دهیم و مجموعه پاسخ‌های بهینه مسئله را با X^* نمایش می‌دهیم.

فرض می‌شود f_iها لزوماً در همه نقاط مشتق‌پذیر نیستند و بنابراین برای آن‌ها زیرگرادیان به صورت زیر تعریف می‌شود:

$$f_i(\bar{x}) - f_i(x) \leq S_{f_i}(\bar{x})\,(x - \bar{x}) \quad \forall x$$

که در آن $S_{f_i}(\bar{x}) \in R^n$ زیرگرادیان f_i در $\bar{x}$ می‌باشد.

$S_{f_i}(\bar{x}) \in \partial f_i(\bar{x}) \subseteq R^n$

→ مجموعه زیردیفرانسیل تابع f_i در $\bar{x}$

الگوریتم تصویرسازی زیرگرادیان توزیع شده:

روند الگوریتم به این صورت می‌باشد که هر عامل از یک تخمین اولیه از جواب مسئله بهینه‌سازی به صورت $x^i(0) \in X \subseteq R^n$ شروع می‌کند و در هر مرحله از الگوریتم این تخمین را به‌روز می‌کند.

وجود متغیر شخصی اثری روی روند اجرای الگوریتم ندارد، چون هر عامل می‌تواند تابع هدفش را نسبت به آن بهینه کند و سپس وارد روند الگوریتم شود:

$$f_i(x) = \min_{y_i} g_i(y_i, x)$$

⇐ همه عامل‌ها به دنبال پیدا کردن یک متغیر مشترک (x) هستند

x^i: تخمین عامل i از x^*

این به‌روزرسانی شامل چند مرحله است:

1. ترکیب تخمین هر عامل با عامل‌های همسایه ← اجماع
   ← با یک الگوریتم Consensus مثل weighted Average

2. حرکت در جهت عکس زیرگرادیان جهت کمینه کردن f_i ← بهینه‌سازی

3. تصویرسازی مقدار به دست آمده در مجموعه X ← قید

← در هر به‌روزرسانی یک قدم به سمت اجماع و یک قدم به سمت بهینه‌سازی (با در نظر گرفتن قید) برداشته می‌شود.

به این ترتیب هر عامل مراحل فوق را به شکل زیر انجام می‌دهد:

1. شروع از شرایط اولیه $x^i(0)$ $\quad i=1,\dots,m \quad , \quad k=0$

2. ترکیب اطلاعات:

فقط برای همسایه‌ها و خود i غیرصفر است

$$v^i(k)=\sum_{j=1}^{m} a_j^i(k)\,x^j(k)$$

weighted average →

3. $$x^i(k+1)=P_X\left[v^i(k)-\alpha_k d_i(k)\right]$$

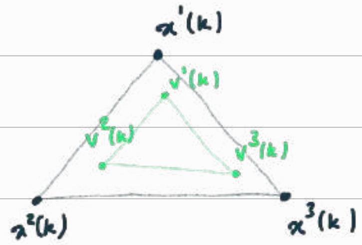
4. $k=k+1$

5. برو به مرحله 2 تا شرط توقف

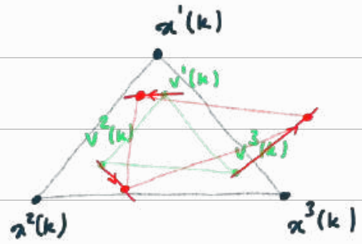
در روابط فوق $a_j^i(k)$ ضرایب ماتریس وزنی $A(k)$ و $\alpha_k>0$ گام به روزرسانی و $d_i(k)$ زیرگرادیان تابع $f_i(x)$ در $x=v^i(k)$ می‌باشد.

Example

$x\in\mathbb{R}^2 \quad , \quad m=3$



→ با ادامه روند به همین شکل مساحت مثلث کوچک و کوچک‌تر می‌شود تا به اجماع برسند



→ با حرکت در جهت عکس زیرگرادیان ممکن است در خلاف حرکت Consensus حرکت انجام شود

← باید بررسی شود با حرکت در جهت عکس زیرگرادیان ها عامل ها به اجماع می رسند ؟

← باید بررسی شود اگر عامل ها به اجماع برسند، مقدار به دست آمده بهینه کننده تابع هدف است ؟

اگرچه پیاده سازی الگوریتم ساده است، ولی از نظر معنوی و محاسباتی نیاز به بررسی دارد
(همگرایی به صورت شهودی مشخص نیست)

← سؤال اصلی: آیا بر اساس الگوریتم فوق مقدار تخمین همه عامل ها به مقدار بهینه کننده مسئله اصلی همگرا می شوند؟

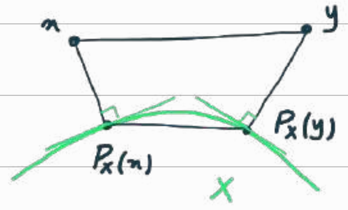
↳ (اجماع + بهینگی)

• همگرایی الگوریتم بهینه سازی توزیع شده

ایده به کار رفته در اثبات همگرایی الگوریتم بهینه سازی توزیع شده پرکاربرد است

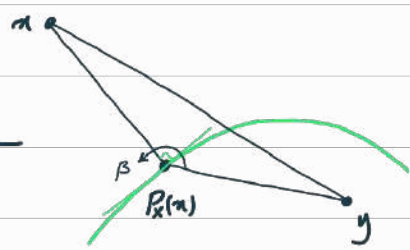
برخی خواص تصویرسازی ($P_X(\bar{x}) = \underset{x\in X}{\text{Argmin}} \|\bar{x} - x\|$):

1 non expansivity: $\|P_X(x) - P_X(y)\| \leq \|x-y\|$



← ضلع روبروی دو زاویه بزرگتر از 90° بزرگترین ضلع چهارضلعی است

2 $\|P_X(x) - y\|^2 \leq \|x-y\|^2 - \|P_X(x) - x\|^2 \quad \forall y \in X$



← چون $\beta \geq 90°$ است نامساوی بالا برقرار است (در حالت $\beta = 90°$ تساوی برقرار می شود)

مفروضات :

1 weight rule $a_i^i \geq \eta$, if $a_j^i > 0$ then $a_j^i \geq \eta$ → مشابه Consensus

وزن هر عامل به خودش مثبت است

وزن عامل هایی که به عامل i یال دارند مثبت است

← همه ضرایب مثبت ماتریس وزنی ، حد پایین η دارند

2 Doubly stochasticity of $A(k)$

$$A(k) = [a_j^i(k)]_{ij} \qquad 1^T A(k) = 1^T \qquad A(k)\,1 = 1$$

← جمع سطرها و ستون های $A(k)$ برابر 1 است

3 $G(V, E_\infty)$ Strongly Connected → گراف در long term ، strongly connected است

$$E_\infty = \{(j,i) \mid (j,i) \in E_k \text{ for infinitely many times } k\}$$

← infinitely often / infinitely many times : یال هایی که گراف را strongly connected

می کنند هر چند وقت یک بار وجود داشته باشند (اجتماع یال هایی که به صورت

infinitely often وصل می شوند strongly connected باشد) (مسیر بین هر دو Node

infinitely often برقرار باشد)

4 به ازای هر $(j,i) \in E_\infty$ عدد $B \geq 1$ وجود دارد که j حداکثر پس از B تکرار اطلاعات خود را برای i ارسال می کند

← هر دو یالی که ترافیک تبادل اطلاعات داشته باشند

← $B(m-1)$ تکرار طول می‌کشد تا دورترین Node ها اطلاعاتشان به هم برسد

3 , 4 ⇒ Periodicly Strongly Connected

5 مجموعه شدنی مسئله بهینه سازی محدب ، بسته و برای همه عامل‌ها معلوم است :

$$\bigcap_{i=1}^{m} X_i = X$$

closed and Convex

→ در بعضی الگوریتم‌ها هر عامل فقط مجموعه شدنی خودش را می‌داند

6 فرض می‌شود زیرگرادیان f_i در مجموعه X محدود است به طوری که :

$$\|d\| \leq L > 0 \quad \forall d \in \partial f_i(x) \quad \forall x \in X \quad \forall i$$

حد بالای زیرگرادیان

الگوریتم بهینه سازی توزیع شده :

$$v^i(k) = \sum_{j=1}^{m} a^i_j(k)\, x^j(k)$$

$$x^i(k+1) = v^i(k) - \alpha_k d_i(k) + \overbrace{\varphi^i(k)}^{\text{اثر Projection}}$$

$$\varphi^i(k) = \underbrace{P_X[v^i(k) - \alpha_k d_i(k)]}_{\text{Project شده}} - \underbrace{(v^i(k) - \alpha_k d_i(k))}_{\text{Project نشده}}$$

← الگوریتم همچنان به اجماع عامل‌ها منجر می‌شود؟

← اجماع در صورت وقوع در مقدار بهینه تابع هدف خواهد بود؟

/ /

بررسی اجماع :

هدف اولیه

از آنجاکه ماتریس وزنی به صورت تصادفی دوبل تعریف شده است ، انتظار داریم عامل ها به مقدار متوسطشان همگرا شوند .
بنابراین تعریف متوسط عامل ها را تعریف می کنیم :

$$y(k) = \frac{1}{m} \sum_{j=1}^{m} x^j(k)$$

$$y(k+1) = \frac{1}{m} \sum_{j=1}^{m} x^j(k+1) = \frac{1}{m} \sum_{i=1}^{m} v^i(k) - \frac{1}{m} \alpha_k \sum_{i=1}^{m} d_i(k) + \frac{1}{m} \sum_{i=1}^{m} \varphi^i(k)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{m} a_j^i x^j(k) = \frac{1}{m} \sum_{j=1}^{m} x^j(k) \underbrace{\sum_{i=1}^{m} a_j^i}_{1} = y(k)$$

A تصادفی دوبل ⇐ 1

$$\Rightarrow y(k+1) = y(k) - \frac{\alpha_k}{m} \sum_{i=1}^{m} d_i(k) + \frac{1}{m} \sum_{i=1}^{m} \varphi^i(k)$$

قضیه : تحت فرضیات ذکر شده (6 - 1) اگر دنباله $\{x^i(k)\}$ توسط الگوریتم تقریب سازی زیرگرادیان توزیع شده (PDSG) تولید شود و دنباله $\{y(k)\}$ به شکل فوق تعریف شود آنگاه :

خطای اجماع

1 $$\lim_{k \to \infty} \| x^i(k) - y(k) \| = 0 \quad \forall i \quad \text{if} \quad \lim_{k \to \infty} \alpha_k = 0$$

2 $$\sum_{k=1}^{\infty} \alpha_k \| x^i(k) - y(k) \| < \infty \quad \forall i \quad \text{if} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

→ قبلاً با تأثیر محدود کننده دیدیم (مثلاً $\alpha_k = \frac{1}{k}$)

→ محدود بودن مجموع وزن دار خطای اجماع در طول تکرارها

اثبات:

مشابه اجماع ، بردار زیرگرادیان

رابطه برداری (x^i اسکالر) : $x(k+1) = \underbrace{A(k)x(k)}_{\text{مشابه اجماع}} - \alpha_k \underbrace{d(k)}_{\text{بردار زیرگرادیان}} + \underbrace{\varphi(k)}_{\text{بردار Projection error}}$ ⟵ فرم Compact (برای نمایش ساده‌تر)

اگر x^i ها برداری باشند رابطه با $\sum$ یا با ضرب کرونکر نمایش داده می‌شود که جلوتر با $\sum$ اثبات را ادامه خواهیم داد

جملات را از مرحله k تا s بررسی می‌کنیم (بررسی فقط مرحله آخر کافی نیست و ممکن است تغییر اتفاق نیفتد)

$$\Phi(k,s) = A(k)A(k-1)\cdots A(s)$$

به دلیل اضافه شدن ($-\alpha_k d(k) + \varphi(k)$) این عبارات اضافه شدند

$$\Rightarrow x(k+1) = \Phi(k,s)x(s) + [-\alpha_k d(k) + \varphi(k)]$$
$$+ \Phi(k,k)[-\alpha_{k-1} d(k-1) + \varphi(k-1)] \qquad \Phi(k,k) = A(k)$$
$$+ \Phi(k,k-1)[-\alpha_{k-2} d(k-2) + \varphi(k-1)] \qquad \Phi(k,k-1) = A(k)A(k-1)$$
$$\vdots$$
$$+ \Phi(k,s+1)[-\alpha_s d(s) + \varphi(s)]$$

بازنویسی رابطه قبل در حالت برداری برای عامل i ام (درایه‌های i امین سطر ماتریس Φ در بردارهای x^j و d_j و φ^j ضرب می‌شود):

$$x^i(k+1) = \sum_{j=1}^{m} [\Phi(k,s)]^i_j \, x^j(s)$$
$$- \sum_{r=s}^{k-1} \sum_{j=1}^{m} [\Phi(k,r+1)]^i_j \, \alpha_r d_j(r) - \alpha_k d_i(k)$$
$$+ \sum_{r=s}^{k-1} \sum_{j=1}^{m} [\Phi(k,r+1)]^i_j \, \varphi^j(r) + \varphi^i(k)$$

برای داشتن مقدار متوسط از رابطه صفحه قبل متوسط می‌گیریم:

$$y(k+1) = y(s) - \frac{1}{m}\sum_{r=s}^{k-1}\sum_{j=1}^{m}\alpha_r d_j(r)$$

$$- \frac{\alpha_k}{m}\sum_{i=1}^{m} d_i(k) + \frac{1}{m}\sum_{r=s}^{k-1}\sum_{j=1}^{m}\varphi^j(r)$$

$$+ \frac{1}{m}\sum_{j=1}^{m}\varphi^j(k)$$

جمله قبل با متوسط‌گیری روی jهای $\sum$ ها ←

اثر Φ ها از بین می‌رود

حال برای $s \to 0$ و $k+1 \to k$ و اندازه تفاضل $x^i(k)$ و $y(k)$ داریم:

$y(0) = \frac{1}{m}\sum_{i=1}^{m} x^i(0)$

$$\| x^i(k) - y(k) \| \leq \sum_{j=1}^{m} \left| [\Phi(k-1,0)]_i^j - \frac{1}{m} \right| \| x^j(0) \|$$

استفاده از نامساوی مثلثی نرم در ضرب و جمع

$$+ \sum_{r=0}^{k-2}\sum_{j=1}^{m} \left| [\Phi(k-1, r+1)]_j^i - \frac{1}{m} \right| \alpha_r \overset{L}{\| d_j(r) \|}$$

از حد بالای d استفاده می‌کنیم

$$+ \alpha_{k-1} \overset{L}{\| d_i(k-1) \|} + \frac{\alpha_{k-1}}{m}\sum_{j=1}^{m} \overset{L}{\| d_j(k-1) \|}$$

$$+ \sum_{r=0}^{k-2}\sum_{j=1}^{m} \left| [\Phi(k-1, r+1)]_j^i - \frac{1}{m} \right| \| \varphi^j(r) \|$$

$$+ \| \varphi^i(k-1) \| + \frac{1}{m}\sum_{j=1}^{m} \| \varphi^j(k-1) \|$$

در بخش اجماع نشان داده شد: $\left| [\Phi(k,s)]_j^i - \frac{1}{m} \right| \leq C\beta^{k-s}$ ← از این حد بالا استفاده می‌کنیم

درایه‌های Φ با نرخ نمایی به سمت $\frac{1}{m}$ می‌روند

$$\Rightarrow \|x^i(k) - y(k)\| \leq C\beta^{k-1} \sum_{j=1}^{m} \|x^j(0)\| + mcL \sum_{r=0}^{k-2} \beta^{k-r-2} \alpha_r$$

$$+ 2\alpha_{k-1} L + C \sum_{r=0}^{k-2} \beta^{k-r-2} \sum_{j=1}^{m} \|\varphi^i(r)\|$$

$$+ \|\varphi^i(k-1)\| + \frac{1}{m} \sum_{j=1}^{m} \|\varphi^j(k-1)\|$$

نشان می‌دهیم خطای Projection ($\varphi^j(k)$) با ضریب α_k به سمت صفر می‌رود ($\|\varphi_i(k)\| \leq \alpha_k L$) :

② خاصیت تقریب‌سازی : $\|P_X(x) - y\|^2 \leq \|x - y\|^2 - \|P_X(x) - x\|^2 \quad \forall y \in X$

همچنین داریم : $\forall x^i \in \overbrace{X}^{\text{Convex}} \Rightarrow \underbrace{v^i(k)}_{\text{Convex Combination of } x^i\text{'s}} \in X$

$$\Rightarrow \|\overbrace{x^i(k+1)}^{v^i(k) - \alpha_k d_i(k) + \varphi^i(k)} - v^i(k)\|^2 \leq \|\cancel{v^i(k)} - \alpha_k \underbrace{d_i(k)}_{L} - \cancel{v^i(k)}\|^2 - \|x^i(k+1) - \overbrace{(v^i(k) - \alpha_k d_i(k))}^{x \text{ قبل از Projection}}\|^2$$

$$\leq \alpha_k^2 L^2 - \|\varphi^i(k)\|^2 \Rightarrow 0 \leq \alpha_k^2 L^2 - \|\varphi^i(k)\|^2$$

$$\Rightarrow \|\varphi^i(k)\| \leq \alpha_k L$$

با جایگذاری حد بالای $\varphi^i(k)$ در نامساوی داریم :

$$\|x^i(k) - y(k)\| \leq C\beta^{k-1} \sum_{j=1}^{m} \|x^j(0)\| + 2mcL \sum_{r=0}^{k-2} \beta^{k-r-2} \alpha_r + 4\alpha_{k-1} L \quad (*)$$

در حالت حدی $k \to \infty$: $\alpha \to 0$, $0 < \beta^k < 1 \Rightarrow \beta \to 0$

$$\Rightarrow \lim_{k\to\infty} \|x^i(k) - y(k)\| \leq 2mcL\beta^{-2} \lim_{k\to\infty} \sum_{r=0}^{k-2} \beta^{k-r} \alpha_r$$

$\sum$ را از جمله K در نظر می‌گیریم   تصاعد هندسی   جملات را تا ∞ جمع می‌کنیم

$$\sum_{r=0}^{k}\beta^{k-r}\alpha_r \leq \underbrace{\gamma_K}_{\substack{\max \alpha_r \\ 0\leq r\leq K}}\sum_{r=0}^{K}\beta^{k-r} + \underbrace{\varepsilon_K}_{\substack{\max \alpha_r \\ K+1\leq r\leq k}}\sum_{r=K+1}^{k}\beta^{k-r} \leq \gamma_K\beta^k\frac{1-\beta^{-(K+1)}}{1-\beta^{-1}} + \varepsilon_K\frac{1}{1-\beta}$$

در نظر می‌گیریم: $K = \frac{k}{2} \Rightarrow K, k \to \infty$ $\longrightarrow$ هم k هم K به ∞ میل می‌کند

$$\Rightarrow \underbrace{-\beta^k\beta^{-(K+1)}}_{\beta^{k-K-1} = \beta^{\frac{K}{2}-1} \to 0} \to 0 \;,\quad \underbrace{\varepsilon_K}_{\substack{\max \alpha_r \\ \infty\leq r}} \to 0 \Rightarrow \lim_{k\to\infty}\sum_{r=0}^{k}\beta^{k-r}\alpha_r = 0$$

$$\Rightarrow \lim_{k\to\infty}\|x^i(k)-y(k)\| = 0 \quad \forall i$$ $\longrightarrow$ اجماع صورت می‌گیرد

گرچه همگرایی را اثبات کردیم ولی نمی‌توانیم بیشتر از sub linear (حداقل نرخ همگرایی) را برای نرخ همگرایی آن ادعا کنیم

$$\circledast \times \alpha_k \Rightarrow \alpha_k\|x^i(k)-y(k)\| \leq C\alpha_k\beta^{k-1}\sum_{j=1}^{m}\|x^j(0)\| + 2mcL\sum_{r=0}^{k-2}\beta^{k-r-2}\alpha_k\alpha_r + 4\alpha_k\alpha_{k-1}L$$

از نامساوی‌های $2\alpha_k\alpha_r \leq \alpha_k^2+\alpha_r^2$ و $\alpha_k\beta^{k-1} \leq \alpha_k^2+\beta^{2(k-1)}$ برای مربع کامل سازی ضرایب استفاده می‌کنیم

$$\Rightarrow \alpha_k\|x^i(k)-y(k)\| \leq c\beta^{2(k-1)}\sum_{j=1}^{m}\|x^j(0)\| + c\alpha_k^2\sum_{j=1}^{m}\|x^j(0)\|$$
$$+ mc\alpha_k^2 L\underbrace{\sum_{r=0}^{k-2}\beta^{k-r-2}}_{\leq \frac{1}{1-\beta}} + mcL\sum_{r=0}^{k-2}\beta^{k-r-2}\alpha_r^2 + 2L(\alpha_k^2+\alpha_{k-1}^2)$$

$\longrightarrow$ جمع تا ∞

$$\Rightarrow \alpha_k\|x^i(k)-y(k)\| \leq c\beta^{2(k-1)}\sum_{j=1}^{m}\|x^j(0)\| + \underbrace{mcA\alpha_k^2}_{A = \frac{1}{m}\sum_{j=1}^{m}\|x^j(0)\| + \frac{L}{\beta-1}} + mcL\sum_{r=0}^{k-2}\beta^{k-r-2}\alpha_r^2 + 2L(\alpha_k^2+\alpha_{k-1}^2)$$

$$\xrightarrow{\sum_{k=1}^{\infty}} \sum_{k=1}^{\infty} \alpha_k \| x^i(k) - y(k) \| \le C \overbrace{\sum_{k=1}^{\infty} \beta^{2(k-1)} \sum_{j=1}^{m} \| x^j(0) \|}^{T1} + \overbrace{\sum_{k=1}^{\infty} \left(mCA\alpha_k^2 + 2L(\alpha_k^2 + \alpha_{k-1}^2) \right)}^{T2}$$

$$+ \underbrace{mCL \sum_{k=1}^{\infty} \sum_{r=0}^{k-2} \beta^{k-r-2} \alpha_r^2}_{T3}$$

$\sum_{k=1}^{\infty} \beta^{2(k-1)}$ محدود است $\Rightarrow$ $T1$ محدود است

$\sum_{k=1}^{\infty} \alpha_k^2$ محدود است $\Rightarrow$ $T2$ محدود است

برای $T3$:
$$\sum_{k=1}^{\infty} \sum_{r=0}^{k} \beta^{k-r} \alpha_r^2 = \sum_{r=0}^{k} \alpha_r^2 \sum_{k=r}^{\infty} \beta^{k-r} \le \frac{1}{1-\beta} \sum_{r=0}^{k} \alpha_r^2 \overset{k \to \infty}{<} \infty$$

چون r از صفر تا k تغییر می‌کرد k کوچک‌تر از r نمی‌شود $(k \ge r)$

$$\Rightarrow \underbrace{\sum_{k=1}^{\infty} \alpha_k \| x^i(k) - y(k) \| < \infty}$$ $\rightarrow$ اثبات بخشی از قضیه

(شرط وزن دهی فقط اجماع)

(وزن دار شده با α_k) نشان دهنده اثر

(Transition time و Projection

قضیه optimality: اگر فرضیات بیان شده 6-1 برقرار باشند و $\{x^i(k)\}$ توسط معادلات زیر تولید شود

$$v^i(k) = \sum_{j=1}^{m} a_j^i(k) x^j(k)$$

$$x^i(k+1) = P_X[v^i(k) - \alpha_k d_i(k)]$$

به طوری که $\underbrace{\sum \alpha_k = \infty}_{\text{non Summable}}$ و $\underbrace{\sum \alpha_k^2 < \infty}_{\text{squared Summable}}$ باشد و همچنین مجموعه X^* غیرتهی (جواب های بهینه) باشد، در این صورت پاسخ بهینه $x^* \in X^*$ وجود دارد به طوری که تخمین همه عامل ها به این پاسخ همگرا می‌شوند. به عبارت دیگر:

$$\lim_{k \to \infty} \| x^i(k) - x^* \| = 0 \quad \forall i$$ $\longrightarrow$ x^i ها به یک پاسخ از X^* همگرا می‌شوند

اثبات:

تابع پتانسیل: $\sum_{i=1}^{m} \| x^i(k+1) - x^* \|^2 = \sum_{i=1}^{m} \| P_X [v^i(k) - \alpha_k d_i(k)] - x^* \|^2$ → امید داریم نزدیک باشد و به سمت صفر برود

خاصیت ① تعدیرسازی: $\| P_X(x) - P_X(y) \| \leqslant \| x - y \|$

(non expansibility)

$x^* \in X \Rightarrow P_X(x^*) = x^*$

$$\Rightarrow \dots \leqslant \sum_{i=1}^{m} \| v^i(k) - \alpha_k d_i(k) - x^* \|^2$$ → $P_X[\cdots]$ را از بین بردیم

$$= \sum_{i=1}^{m} \| v^i(k) - x^* \|^2 + \sum_{i=1}^{m} \alpha_k^2 \| d_i(k) \|^2 - 2\alpha_k \sum_{i=1}^{m} \underline{d_i^T(k)(v_i(k) - x^*)}$$

طبق تعریف زیرگرادیان → $\geqslant f_i(v^i(k)) - f_i(x^*)$

$$\leqslant \sum_{i=1}^{m} \| v^i(k) - x^* \|^2 + m\alpha_k^2 L^2 - 2\alpha_k \sum_{i=1}^{m} [f_i(v^i(k)) - f_i(x^*)]$$

$L \sum_{j=1}^{m} a_j^i x^*(k)$

$\sum_{j=1}^{m} a_j^i x^j(k)$

$$\sum_{i=1}^{m} \| \sum_{i=1}^{m} a_j^i(k) (x^j(k) - x^*) \|^2 \leqslant \sum_{i=1}^{m} \sum_{j=1}^{m} a_j^i \| x^j(k) - x^* \|^2 = \sum_{i=1}^{m} \| x^j(k) - x^* \|^2$$

عبارتی که به دنبال آن بودیم به دست آمد

$$\Rightarrow \sum_{j=1}^{m} \| x^j(k+1) - x^* \|^2 \leqslant \sum_{j=1}^{m} \| x^j(k) - x^* \| + m\alpha_k^2 L^2 - 2\alpha_k \sum_{i=1}^{m} [f_i(v^i(k)) \underline{- f_i(x^*) + f_i(y(k))} - f_i(y(k))]$$

⊕

$y(k)$ متشکل از x^i هاست ⇐ $y(k)$ است Convex Combination ⇐ $f_i(y(k)) \geqslant f_i(x^*)$

هم بالای زیرگرادیان

$$\sum_{i=1}^{m} | f_i(v^i(k)) - f_i(y(k)) | \leqslant \sum_{i=1}^{m} L \| v^i(k) - y(k) \| \leqslant L \sum_{j=1}^{m} \sum_{i=1}^{m} a_j^i \| x^j(k) - y(k) \|$$

تعریف زیرگرادیان / شرط لیپشیتز

$\sum_{j=1}^{m} a_j^i x^j$ $\quad \sum_{j=1}^{m} a_j^i y$

$$= \sum_{j=1}^{m} L \| x^j(k) - y(k) \|$$

$$\Rightarrow \sum_{i=1}^{m} \| x^j(k+1) - x^* \|^2 \leq \sum_{j=1}^{m} \| x^j(k) - x^* \|^2 + m\alpha_k^2 L^2 - 2\alpha_k \sum_{i=1}^{m} [f_i(y(k)) - f_i(x^*))]$$

$$+ 2\alpha_k L \sum_{i=1}^{m} \| x^i(k) - y(k) \|$$

عبارت به دست آمده را محافظه‌کارانه اضافه می‌کنیم چون علامت آن را نمی‌دانیم

عبارات ظاهر شده نشان دهنده نزولی بودن تابع پتانسیل نیستند! در نتیجه در حالت حدی و به صورت long term به آن‌ها نگاه می‌کنیم

← طرفین را از K تا N جمع می‌کنیم ($f = \sum_{i=1}^{m} f_i$)

$$\Rightarrow \sum_{i=1}^{m} \| x^i(N+1) - x^* \|^2 \leq \sum_{i=1}^{m} \| x^i(K) - x^* \|^2 + mL^2 \sum_{k=K}^{N} \alpha_k^2$$

عبارت‌های بالایی با هم ساده می‌شوند و تنها عبارت اول در آخر باقی می‌ماند

$$- 2\sum_{k=K}^{N} \alpha_k (f(y(k)) - f(x^*)) + 2L \sum_{k=K}^{N} \alpha_k \sum_{i=1}^{m} \| x^i(k) - y(k) \|$$

حالت حدی: $K=0$, $N \to \infty$

$$\Rightarrow 0 \leq \sum_{i=1}^{m} \| x^i(N+1) - x^* \|^2 \leq \overbrace{\sum_{i=1}^{m} \| x^i(0) - x^* \|^2}^{<\infty} + \overbrace{mL^2 \sum_{k=0}^{\infty} \alpha_k^2}^{<\infty\ (\text{فرض})}$$

$$\underbrace{- 2\sum_{k=0}^{\infty} \alpha_k (f(y(k)) - f(x^*))} + \underbrace{2L \sum_{k=0}^{\infty} \alpha_k \sum_{i=1}^{m} \| x^i(k) - y(k) \|}_{<\infty\ (\text{نشان دادیم})}$$

باید $<\infty$ باشد چون در غیر این صورت کل عبارت را منفی می‌کند در صورتی که کل عبارت مثبت است

$$\Rightarrow \sum_{k=0}^{\infty} \alpha_k (f(y(k)) - f(x^*)) < \infty$$

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad , \quad y(k) \in X \Rightarrow f(y(k)) - f^* \geq 0$$

$$\Rightarrow \liminf_{k\to\infty} \left[f(y(k)) - f^* \right] = 0$$

← اگر از یک K بزرگ به بعد حد پایین مثبت داشته باشد:

$$\sum_{k=K}^{\infty} \alpha_k \left(f(y(k)) - f^* \right) \geq \sum_{k=K}^{\infty} \alpha_k \varepsilon = \varepsilon \sum_{k=0}^{\infty} \alpha_k \to \infty \Rightarrow \sum_{k=0}^{\infty} \alpha_k \left(f(y(k)) - f^* \right) \to \infty$$

∞ بزرگتر

$N \to \infty$, $K \to \infty$ (مثلاً : $N = 2K \to \infty$)

$$\Rightarrow \sum_{i=1}^{m} \| x^i(N+1) - x^* \|^2 \leq \sum_{i=1}^{m} \| x^i(K) - x^* \|^2 + mL^2 \sum_{k=K}^{N} \alpha_k^2$$

$$- 2 \sum_{k=K}^{N} \alpha_k \left(f(y(k)) - f(x^*) \right) + 2L \sum_{k=K}^{N} \alpha_k \sum_{i=1}^{m} \| x^i(k) - y(k) \|$$

$$\Rightarrow \lim_{N\to\infty} \sum_{i=1}^{m} \| x^i(N+1) - x^* \|^2 \leq \lim_{K\to\infty} \sum_{i=1}^{m} \| x^i(K) - x^* \|^2$$

→ در ∞ رشته تابع نامساوی غیرصعودی است!

به دلیل محافظه‌کاری‌ها و نامساوی‌های استفاده شده در اثبات در عمل همگرایی خیلی زودتر رخ می‌دهد

← چون دنباله گفته شده مثبت (Bounded Below) و غیرصعودی است می‌توان گفت همگراست

⇐ به ازای $x^* \in X^*$ دنباله $\sum_{i=1}^{m} \| x^i(k) - x^* \|$ همگراست و داشتیم: $\lim_{k\to\infty} \| x^i(k) - y(k) \| = 0$

بنابراین دنباله $\{ \| y(k) - x^* \| \}$ به ازای هر $x^* \in X^*$ همگراست و در نتیجه $y(k)$ به مقداری همگرا خواهد شد

و این به علاوه پیوستگی تابع f و $\liminf_{k\to\infty} f(y(k)) = f^*$ اثبات می‌کند $y(k)$ به یک $x^* \in X^*$ همگرا می‌شود

← چون $y(k)$ در بی‌نهایت ثابت است inf حذف می‌شود

/ /

اثبات اجماع

$$\Rightarrow \lim_{k\to\infty} y(k) = x^* \Rightarrow \lim_{k\to\infty} x^i(k) = x^* \quad \forall i$$ → متغیرهای عامل ها به مقدار بهینه همگرا می شود.

* مسئله بهینه سازی توزیع شده با قیود تساوی و نامساوی

تصویرسازی روی قیود ممکن است مشکل باشد → حل با قیود به صورت تساوی و نامساوی می تواند مزیت داشته باشد

تعریف مسئله:
$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{N} f_i(x)$$

ممکن است بتوان هر قیود را به صورت تساوی و نامساوی بیان کرد

$$\text{s.t.} \quad g(x) \leq 0 \quad h(x) = 0 \quad x \in X$$

Convex ← $f_i : \mathbb{R}^n \to \mathbb{R}$ (Private information of agent-i)

X : Compact and Convex

x : global decision variable

Convex ← $g : \mathbb{R}^n \to \mathbb{R}^m$ $\quad g(x) \leq 0 \to g_l \leq 0 \quad l \in \{1, \dots, m\}$

$h : \mathbb{R}^n \to \mathbb{R}^v$ $\quad h(x) = Ax - b \quad A \in \mathbb{R}^{v\times n}$

- فرض می کنیم قیود و مجموعه X برای عامل ها معلوم است

- ارتباط عامل ها از طریق یک گراف (می تواند تغییر با زمان باشد) $G_t(V, E_t)$ برقرار می شود

- P^* مقدار بهینه مسئله بهینه سازی و محدود می باشد

• x^* مجموعه پاسخ های بهینه مسئله ، غیرتهی می باشد

مسئله فوق را به صورت زیر بازنویسی می کنیم :

$$\min_{x} \sum_{i=1}^{N} f_i(x)$$

$$\text{s.t.} \quad N g(x) \leq 0$$ → ضرب N در جواب مسئله تاثیری ندارد

$$N h(x) = 0$$ ولی در حل توزیع شده به ما کمک می کند

$$x \in X$$

برای مسئله فوق تابع Penalty dual (یک نوع لاگرانژین modify شده) را تعریف و بهینه می کنیم .

ابتدا تابع پنالتی را به صورت زیر تعریف می کنیم :

$$H : R^n \times R^{m \geq 0} \times R^{v \geq 0} \to R$$

$$H(x, \mu, \lambda) = f(x) + N\mu^T \underbrace{[g(x)]^+}_{\text{Max}(0,\, g(x))} + N\lambda^T \overbrace{|h(x)|}^{\geq 0}$$ → 3 تفاوت با تابع لاگرانژین

بر این اساس تابع Penalty dual به صورت زیر تعریف می شود :

$$q_p : R^{m \geq 0} \times R^{v \geq 0} \to R$$

$$q_p = \inf_{x \in X} H(x, \mu, \lambda)$$ → شبیه ارتباط لاگرانژ و مسئله دوگان

/ /

حل مسئله Penalty dual به صورت زیر تعریف می‌شود:

$$\max_{\substack{\mu \in R^m \geq 0 \\ \lambda \in R^v \geq 0}} q_p(\mu, \lambda)$$
$$\text{s.t.} \quad \mu \geq 0$$
$$\lambda \geq 0$$

Dual معمولی نه Penalty dual

فرض کنید (μ^*, λ^*) یک پاسخ بهینه مسئله Dual باشند.

$$d_a^* = q_a(\mu^*, \lambda^*) = \inf_{x \in X} \{ f(x) + N\mu^{*T} g(x) + N\lambda^{*T} h(x) \}$$

$$\leq \inf_{x \in X} \{ f(x) + N\mu^{*T} [g(x)]^+ + N|\lambda^{*T}| \, |h(x)| \}$$

$$= q_p(\mu^*, |\lambda^*|) \leq \max_{\substack{\mu \geq 0 \\ \lambda \geq 0}} q_p(\mu, \lambda) = d_p^*$$

حل مسئله Penalty Dual

مقدار بهینه تابع dual

$$\Rightarrow d_a^* \leq d_p^*$$

مقدار بهینه تابع Penalty dual

می‌دانیم که برای هر $x^* \in X^*$ که شدنی می‌باشد داریم:

$$x^* \in X^* \,, \quad [g(x^*)]^+ = 0 \,, \quad |h(x^*)| = 0$$

$g(x^*) \leq 0$

بنابراین خواهیم داشت:

$$q_p(\mu, \lambda) = \inf_x H(x, \mu, \lambda) \leq \underset{x^* \in X^*}{H(x^*, \mu, \lambda)} = f(x^*) = p^*$$

$\forall \mu, \lambda$

پاسخ بهینه مسئله

$$\xrightarrow[\mu \geq 0 \ \lambda \geq 0]{\max} \max_{\mu \geq 0 \ \lambda \geq 0} q_p(\mu, \lambda) = d_p^* \leq p^*$$

$$\Rightarrow \boxed{d_a^* \le d_p^* \le P^*}$$

از نامساوی فوق روشن است اگر شرایطی برقرار باشد که $d_a^* = P^*$ باشد (مثلاً اگر مسئله بهینه سازی محدب و شرایط slater برقرار باشد) در این صورت خواهیم داشت :

$$d_p^* = P^*$$

← اگر Duality gap صفر باشد Duality gap تابع Penalty dual هم صفر است

← duality gap برای مسئله Penalty dual کوچکتر یا مساوی duality gap در مسئله dual است :

$P^* - d_p^* \le P^* - d_a^*$ ⟶ مزیت تابع Penalty dual

مزیت دیگر تابع پنالتی نسبت به تابع لاگرانژ، محدب بودن تابع پنالتی نسبت به متغیر x می باشد که در تابع لاگرانژ به دلیل امکان منفی بودن ضرایب λ این امر لزوماً محقق نمی شود

$f(x)$: محدب

$Max(0,x)$: محدب

$g(x)$: محدب

← $[g(x)]^+$ محدب با ضریب مثبت

$|x|$: محدب

$h(x)$: affine

← $|h(x)|$ محدب با ضرایب مثبت

$\Leftarrow$ H : محدب

در تابع لاگرانژ λ می تواند منفی باشد $\Leftarrow$ تابع لاگرانژ لزوماً محدب نشود

تعریف نقطه زینی برای تابع پنالتی : چندتایی مرتب (x^*, μ^*, λ^*) یک نقطه زینی تابع پنالتی بر $X \times R^{m}_{\ge 0} \times R^{l}_{\ge 0}$ می باشد اگر و فقط اگر این چندتایی مرتب مقادیر بهینه مسائل Primal و Penalty dual باشند و تساوی زیر برقرار باشد:

حل مسئله Penalty dual ← $\sup\limits_{\mu\ge0,\ \lambda\ge0}\ \inf\limits_{x\in X} H(x,\mu,\lambda) = \inf\limits_{x\in X}\ \sup\limits_{\mu\ge0,\ \lambda\ge0} H(x,\mu,\lambda)$ → حل مسئله Primal

($\sup(\dots)$ یا ∞ می شود یا f که با انتخاب x شدنی f حاصل می شود)

/ /

subgradiant, supergradiant — می‌تواند توزیع شود

به سادگی می‌توان نشان داد تابع بالایی ($H(x,\mu,\lambda)$ به صورت مجموع توابع Convex - Concave می‌باشد:

$$H_i(x,\mu,\lambda) = f_i(x) + \mu^T[g(x)]^+ + \lambda^T|h(x)| \Rightarrow H(x,\mu,\lambda) = \underbrace{\sum_{i=1}^{N}}_{\text{مجموع}} H_i(x,\mu,\lambda)$$

$H_i(x,w)$: $w=(\mu,\lambda)$ fixed $\rightarrow$ Convex in x

$H_i(x,\mu,\lambda)$: x fixed $\rightarrow$ Concave in (μ,λ) $\rightarrow$ H نسبت به (μ,λ) خطی است ولی چون هدف Maximization است آن را Concave درنظر می‌گیریم

الگوریتم توزیع شده زیرگرادیان برای حل مسئله Penalty-Primal-dual:

1 Initialization: $\lambda^i(0) \in R^v_{\geq 0}$ $\mu^i(0) \in R^m_{\geq 0}$ $x^i(0) \in X$ $y^i(1) = Nf_i(x^i(0))$ $\forall i \in V$

چون از $x(0)$ استفاده می‌کند

تخمین اولیه از $\sum f_i$ (تابع هدف اصلی)
(تغییر کمکی است و تاثیری در خود بهینه‌سازی ندارد)

2 Information fusion: $V_x^i(k) = \sum_{j=1}^{N} a_j^i(k) x^j(k)$

نظر برای همسایگان عامل i و خود عامل i غیرصفر است

$$V_\mu^i(k) = \sum_{j=1}^{N} a_j^i(k)\mu^j(k)$$

$$V_\lambda^i(k) = \sum_{j=1}^{N} a_j^i(k)\lambda^j(k)$$

$$V_y^i(k) = \sum_{j=1}^{N} a_j^i(k)y^j(k)$$

3 Optimization: $x^i(k+1) = P_x[V_x^i(k) - \underbrace{\alpha(k)}_{\text{step size}} D_x^i(k)]$

زیرگرادیان تابع $H_i(x, V_\mu^i(k), V_\lambda^i(k))$ نسبت به x در $x = V_x^i(k)$

اگر همه قیود به صورت تساوی‌ها و نامساوی‌ها بیان شوند Projection نخواهیم داشت

$$\Rightarrow D_x^i(k) = Df_i(V_x^i(k)) + \sum_{l=1}^{m}[V_\mu^i(k)]_l\, D[g_l(V_x^i(k))]^+ + \sum_{l=1}^{v}[V_\lambda^i(k)]_l\, D|h_l(V_x^i(k))|$$

$$\mu^i(k+1) = \underbrace{V_\mu^i(k)}_{+} + \underbrace{\alpha(k)}_{+}\,\underbrace{[g(V_x^i(k))]^+}_{+} \geq 0$$

supergradient of H_i at $(V_x^i(k), V_\mu^i(k), V_\lambda^i(k))$

$$\lambda^i(k+1) = \underbrace{V_\lambda^i(k)}_{+} + \underbrace{\alpha(k)}_{+}\,\underbrace{[|h(V_x^i(k))|]}_{+} \geq 0$$

supergradient of H_i at $(V_x^i(k), V_\mu^i(k), V_\lambda^i(k))$

supergradient درمکان subgradient است برای توابع Concave (چون H_i نسبت به μ^i و λ^i خطی است از هر دو عبارت می‌توانیم استفاده کنیم)

$$y^i(k+1) = V_y^i(k) + N(\underbrace{f_i(x^i(k)) - f_i(x^i(k-1))}_{\text{معیاری از همگرایی است}})$$

→ Agent ها تابع هدف را یاد می‌گیرند (اثری در بهینه‌سازی ندارد)

(در نهایت: $Nf_i(0) \to \sum f_i \to p^*$)

شرایط به‌روزرسانی $\alpha(k)$:

چون الگوریتم پیچیده‌تر شده شرایط هم پیچیده‌تر می‌شود

1. $\lim_{k\to\infty} \alpha(k) = 0$
2. $\lim_{k\to\infty} \alpha(k+1)\,S(k) = 0 \qquad S(k) = \sum_{l=0}^{k} \alpha(l)$
3. non summable: $\sum_{k=0}^{\infty} \alpha(k) = \infty$
4. squared summable: $\sum_{k=0}^{\infty} \alpha^2(k) < \infty$
5. $\sum_{k=0}^{\infty} \alpha^2(k+1)\,S(k) < \infty$
6. $\sum_{k=0}^{\infty} \alpha(k+1)\,S^2(k) < \infty$

گرچه شرایط پیچیده‌تر است ولی $\alpha(k) = \frac{1}{k+1}$ همه شروط را برقرار می‌کند

قضیه (همگرایی الگوریتم توزیع شده Penalty-Primal-dual): مسئله بهینه سازی چندعامله مقید مذکور را در نظر بگیرید و فرض کنید فرض های 6 - 1 (که در الگوریتم زیرگرادیان تصویر شده و توزیع شده بیان شد) برقرار باشد. در این صورت اگر $\{x^i(k)\}$ و $\{y^i(k)\}$ به کمک الگوریتم DPPDS (Distributed Penalty Primal Dual subgradient) تولید شده باشند و step size ($\alpha(k)$) شرایط فوق را برآورده کند، آنگاه پاسخ $\tilde{x} \in X^*$ وجود دارد به طوری که:

برای این مسئله X به جای closed باید Compact (closed + bounded) باشد

$$\lim_{k \to \infty} \| x^i(k) - \tilde{x} \| = 0 \qquad \forall i \in V$$

$\rightarrow$ x^i ها به پاسخ بهینه می‌رسند $\Rightarrow$ اجماع صورت گرفته

$$\lim_{k \to \infty} \| y^i(k) - P^* \| = 0 \qquad \forall i \in V$$

$\rightarrow$ y^i ها به $\underbrace{\text{بهینه } \Sigma f_i}_{P^*}$ می‌رسند $\Rightarrow$ همگرایی به مقدار بهینه صورت گرفته

← توقف الگوریتم را با بررسی تغییر y^i ها می‌توان انجام داد

* Extensions / Applications

همه مواردی که در بخش Decentralized گفته شد، در صورتی که در آنها عامل ها بتوانند ارتباط گرافی داشته باشند در اینجا هم کاربرد خواهد داشت

- stochastic consensus-based subgradient methods

$$x_i(t+1) = P_X[w_i(t+1) - \alpha(t)(\tilde{\nabla} f_i(w_i(t+1)) + \underbrace{e_i(t+1)}))]$$

خطای محاسبه زیرگرادیان ∇f_i (نویز اندازه‌گیری، خطای محاسبه تابع هدف (برای مثال از روش Monte Carlo evolution، ...)

فرض می‌شود خطا از نظر Bound به مفهوم Mean Square محدود است (واریانس خطا محدود است)

همگرایی الگوریتم به طور متوسط بررسی می‌شود

← α هم متغیر تصادفی می‌شود

- Consensus-based Subgradient method for noisy links → وجود نویز در راه‌های ارتباطی

$$w_i(t+1) = x_i(t) - \eta(t) \sum_{j=1}^{m} r_{ij} \left(x_j(t) + \underbrace{\xi_{ij}(t)}_{\text{random zero mean noise on the link from } j \text{ to } i}\right)$$

$$r_{ij} = 0 \quad \text{when } j \notin N_i$$

$\eta(t) > 0$ is a noise-damping step size → به همگرایی کمک می‌کند

$$x_i(t+1) = w_i(t+1) - \alpha(t) g_i(w_i(t+1))$$ → نویز در تست زیرگرادیان هم اثر می‌گذارد

شرایط همگرایی برای ترکیب $\eta(t)$ و $\alpha(t)$ وجود خواهد داشت:

$$\sum_t \alpha(t) = \infty \qquad \sum_t \alpha^2(t) < \infty$$

$$\sum_t \eta(t) = \infty \qquad \sum_t \eta^2(t) < \infty$$

$$\sum_t \alpha(t)\eta(t) < \infty \qquad \sum_t \frac{\alpha^2(t)}{\eta(t)} < \infty$$

- Consensus-based Subgradient methods for distributed sets

تا به حال فرض می‌کردیم هر عامل ها از $X = \bigcap_{i=1}^{m} X_i$ آگاه هستند ولی می‌توان X_i را اطلاعات شخصی عامل i فرض کرد

در این مدل هر الگوریتم Projection را برای Set خودش انجام می‌دهد:

$$x_i(t+1) = P_{X_i}\left[w_i(t+1) - \alpha(t) \tilde{\nabla} f_i(w_i(t+1))\right]$$

/ /

ابتدا شرایط محدودکننده‌ای برای همگرایی تعیین شد

سپس شرایط relax شد و به حالت stochastic (Random Graph + Communication Noise) تعمیم داده شد

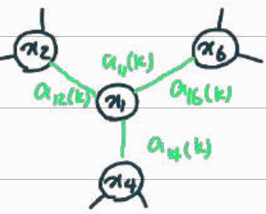
- Distributed Random Projection (DRP) Algorithm

local Set هر عامل (X_i) می‌تواند اشتراک مجموعه‌ها یا قیدهای کوچک‌تری باشد ← Projection روی هرکدام می‌تواند ساده‌تر باشد

← هردور به صورت تصادفی یکی از این set ها انتخاب می‌شود و Projection روی آن اعمال می‌شود
(شبیه ایدهٔ Stochastic Gradient Descent که داده تصادفی انتخاب می‌شود)

$\Rightarrow$ Initialize $x_i(0)$ $\forall i \in V$

for $k \geq 0$, $\forall i$: 1. Mixing $\quad v_i(k) = \sum_{j=1}^{m} a_{ij}(k)\, x_j(k)$



2. Gradient Update $\quad \tilde{v}_i(k) = v_i(k) - \alpha_k \nabla f_i(v_i(k))$

جزء set های X_i

3. Projection $\quad$ A random varable $\Omega_i(k) \in I_i$ is drawn

A Component $X_i^{\Omega_i(k)}$ of $X_i = \bigcap_{j \in I_i} X_i^j$ is used

for Projection : $x_i(k+1) = \Pi_{X_i^{\Omega_i(k)}} [\tilde{v}_i(k)]$

- Distributed Mini-Batch Random Projection (DMRP)

تعداد قیدها می‌تواند خیلی زیاد شود (مثلاً X_i حاصل اشتراک 10^4 ابرصفحه باشد) ← مثلاً در SVM هر داده یک قید دارد

← هر نمونه تخمین ضعیفی از X_i می‌شود ← 100 نمونه منجر به تخمین بهتری می‌شود

/ /

$\Rightarrow$ Initialize $x_i(0)$ $\forall i \in V$

for $k \geq 0$, $\forall i$: **1. Mixing** $\quad v_i(k) = \sum_{j=1}^{m} a_{ij}(k)\, x_j(k)$

تعداد Projection

2. Gradient Update $\quad \psi_i^0(k) = v_i(k) - \alpha_k \nabla f_i(v_i(k))$

3. Projections A batch of independent random variables $\Omega_i^r(k) \in J_i$

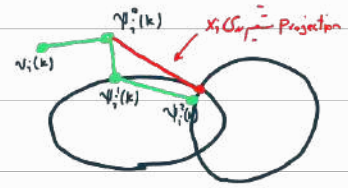
$r = 1, \dots, b$ is drawn.

The Components $X_i^{\Omega_i^1(k)}, \dots, X_i^{\Omega_i^b(k)}$ of $X_i = \bigcap_{j \in J_i} X_i^j$

are used for sequential Projections:

$$\psi_i^r(k) = \Pi_{X_i^{\Omega_i^r(k)}}\left[\psi_i^{r-1}(k)\right] \quad \text{for } r = 1, \dots, b$$

$$x_i(k+1) = \psi_i^b(k)$$

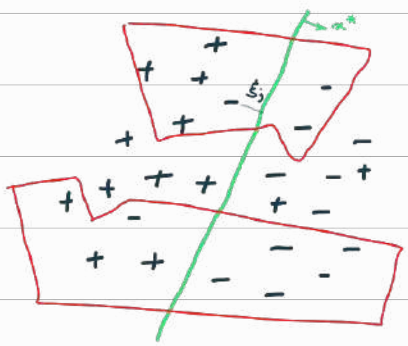


← به نقطه دیگری می‌رسیم ولی هزینه محاسباتی Projection کم می‌شود

← باعث sub optimality می‌شود ولی Performance نسبت به random Projection بهتر می‌شود

- Support Vector Machine (SVM)

m مرکز که هر مرکز i مجموعه داده $\{(z_j, y_j), j \in J_i\}$ را دارد



$$\Rightarrow \min_{\substack{x \in R^d \\ \xi \in R^p}} \sum_{i=1}^{m} \left(\frac{1}{2m} \|x\|^2 + C \sum_{j \in J_i} \xi_j \right)$$

$$\text{s.t.} \quad (x, \xi) \in \bigcap_{i=1}^{m} X_i$$

$$X_i \triangleq \{(x, \xi) \mid y_j \langle x, z_j \rangle \geq 1 - \xi_j,\ \xi_j \geq 0,\ j \in J_i\} \quad i = 1, \dots, m$$

هر local Constraint شامل تعداد زیادی Constraint (به تعداد داده‌ها) می‌شود

هر Constraint ساده است ولی Projection روی اشتراکشان پیچیده می‌شود ← DMRP

- Asynchronous Consensus-based Subgradient methods

→ Gossip → لازم نیست همه Agent ها همزمان Update شوند

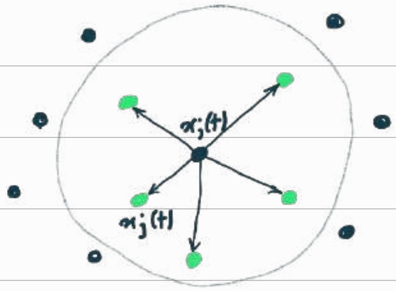
$$v_i(k) = \left(x_{I_k}(k-1) + x_{J_k}(k-1)\right)/2 \qquad i \in \{I_k, J_k\}$$

Random Projection با ترکیب ←
$$x_i(k) = \Pi_{X_i^{\Omega_i(k)}}\left[v_i(k) - \alpha_i(k)\nabla f_i(v_i(k))\right]$$

$$x_i(k) = x_i(k-1) \qquad \forall i \notin \{I_k, J_k\}$$ → بقیه عامل‌ها تغییری نکنند

→ Broad Cast

در هر Iteration یک عامل فعال می‌شود (مشابه gossip) ولی اطلاعاتش
را برای همه همسایه‌هایش ارسال می‌کند



⇒ At time t, agent I_t wakes up and broadcasts its estimate $x_i(t)$

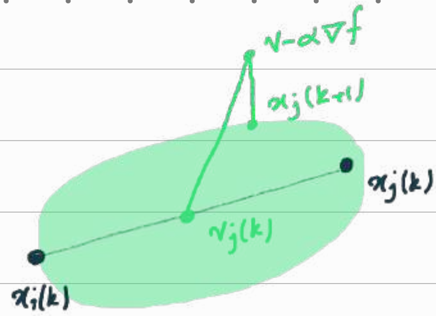
Agent I_t and $j \notin N_{I_t}$ do not update:

$$x_j(t+1) = x_j(t) \qquad \text{for } j \notin \{I_t\} \cup N_{I_t}$$

Neighbors $j \in N_{I_t}$ of agent I_t update:

$$v_j(t) = \beta x_{I_t}(t) + (1-\beta)\, x_j(t) \quad \longrightarrow \quad \beta: \text{ضریب}$$

$$x_j(t+1) = \Pi_X\left[v_j(t) - \alpha_j(t)\left(\nabla f_j(v_j(t)) + \underbrace{\epsilon_j(t)}_{\text{Error}}\right)\right]$$



- Distributed ADMM

در نسخه‌ای که داشتیم Agent ها به صورت Sequential به روزرسانی می‌کنند و متغیر Dual را برای Coordinator ارسال می‌کردند

در Extension توزیع شده همه همزمان به روزرسانی می‌کنند و هرکدام تخمینی از متغیر Dual دارند

← بدون نیاز به Coordinator و به روزرسانی Sequential روی گراف عمل می‌کند

← با وجود قطع هم برقرار شده

- A Fast Second order Distributed Method

روش های گسترش شده (sub)gradient-based بودند که First order هستند
تعمیم به روش های Second-order (مثل نیوتن) هم انجام شده که محاسبات بیشتر دارد اما سریع‌تر است

- Applications

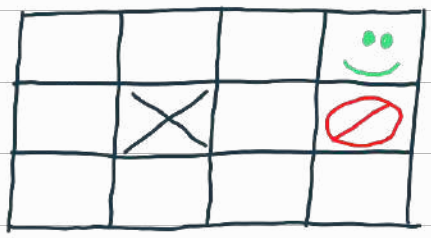
→ electric Power Systems
→ Machine Learning
→ Deep Learning
→ Computer Networks → به طور کلی هرجا که عامل های مختلف داشته باشیم قابل استفاده است
→ Communication Networks

یادگیری تقویتی چند عامله (Multi Agent Reinforcement Learning)

* Dynamic Optimization

Example

بهینه‌سازی در یک Grid World :



→ از هر خانه به چهار جهت می‌توان حرکت کرد

→ هر حرکت Cost دارد

→ رسیدن به ⊘ هزینه بالا دارد

→ رسیدن به ☺ پاداش بالا دارد

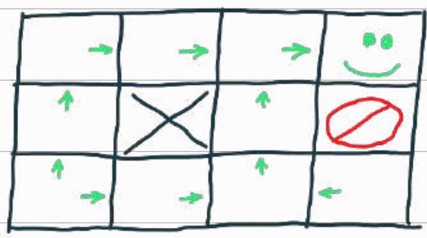
برای حل به روش های بهینه‌سازی که تا به حال دیدیم: از تابع هدف مدل explicit و پیوسته نداریم

اگر تغییر بهینه‌سازی مسیر باشد، تعداد حالات خیلی زیاد می‌شود (curse of dimensionality)

← اگر جهت ها پیوسته بود تعداد حالات بی‌نهایت می‌شد

⇐ در محیط state (وضعیت) تعریف می‌کنیم و در هر وضعیت بهترین جهت را پیدا می‌کنیم (Multi Stage Decision Making)

هر خانه را یک وضعیت در نظر می‌گیریم ⇐



حتی ممکن است مدل قرارگیری هدف را ندانیم! (به نوعی تابع هدف را حتی به صورت Implicit هم نداریم)

⇐ یک Multi Stage Decision Making داریم

⇐ یک متغیر جدید به نام State (وضعیت) داریم که مشخص می‌کند چه تصمیمی بگیریم

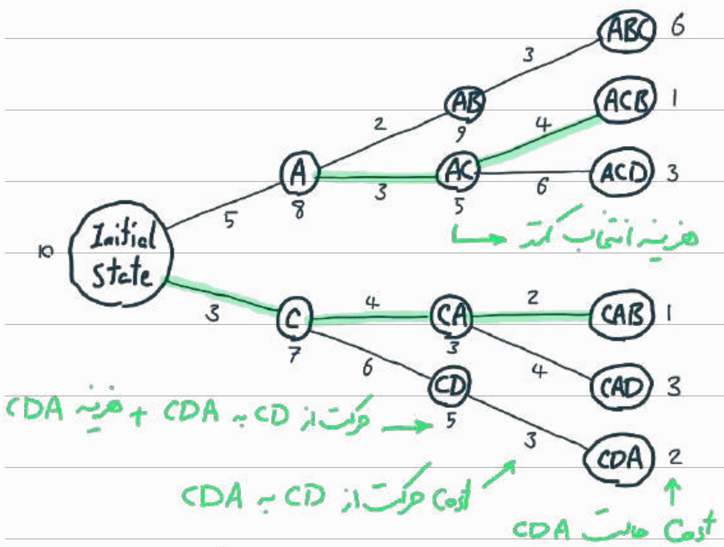
⇐ تصمیم‌گیری در هر state می‌تواند منجر به مجموعه‌ای از حالت‌ها شود (در یک مسئله Deterministic مجموعه تک عضوی است)

⇐ هر حالت (با توجه به اینکه چطور در رسیدن به هدف کمک می‌کند) یک ارزش دارد و در تابع هدف اثرگذار است

⇐ رفتن از یک حالت به حالت دیگر، به حالتی که در آن بودیم و تصمیمی که گرفتیم بستگی دارد

state یک متغیر دینامیکی است

Example



مثال از scheduling (Dynamic Programming):

می‌خواهیم بهترین ترتیب بین A, B, C, D را پیدا کنیم که کمترین هزینه را داشته باشد

با رویکرد Dynamic Programming، از انتها Cost ها را به دست می‌آوریم و بهترین انتخاب در هر state را پیدا می‌کنیم تا به Initial State برسیم (حل Bottom-UP)

← اگر تعداد state ها زیاد باشد، حل تصمیم و جستجوی همه مسیرها بسیار زمان‌بر است

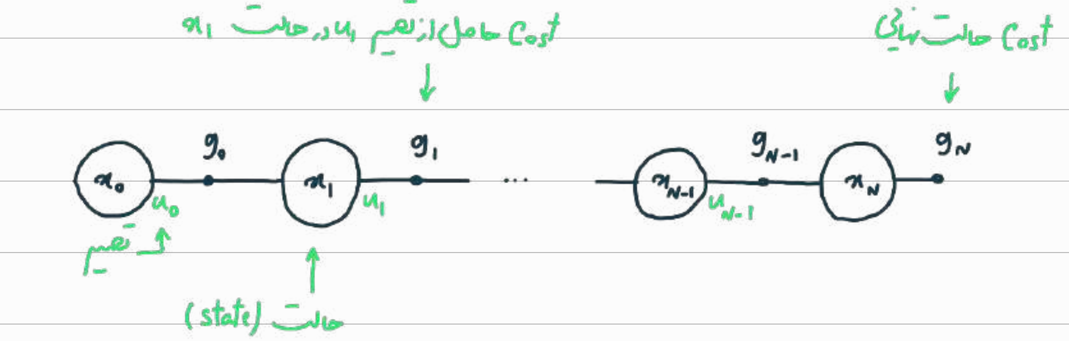
⇐ مسیر بهینه: C → CA → CAB ، هزینه: 10

← از انتها به حل subproblem ها می‌پردازیم و به عقب حرکت می‌کنیم

← در هر state-time، مقدار Cost to go و تصمیم بهینه را ذخیره می‌کنیم

هزینه از state به بعد

end



$$\Rightarrow \min_{u_1, \dots, u_{N-1}} E\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right]$$

عامل تصادفی (مسئله می‌تواند حالت تصادفی داشته باشد)

می‌تواند Explicit باشد یا نباشد

$\Rightarrow$ System: $x_{k+1} = f_k(x_k, u_k, w_k) \quad k = 0, \dots, N-1$ → Dynamic

Control Constraints: $u_k \in U_k(x_k)$ → تصمیم‌گیری

Probability Distribution: $w_k \sim P(\cdot \mid x_k, u_k)$

Policies: $\pi = \{\mu_0, \dots, \mu_{N-1}\}$, $u_k = \mu_k(x_k)$ → به دنبال دنباله π بهینه هستیم

(استراتژی) (μ) بهینه در هر state

state را به تصمیم Map می‌کند

(تصمیم به state وابسته است)

Expected Cost of π starting from x_0: $J_\pi(x_0) = E\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right]$

Optimal Cost Function: $J^*(x_0) = \min_\pi J_\pi(x_0)$

Dynamic Programming (DP) Algorithm:

- start with $J_N(x_N) = g_N(x_N)$

Cost رسیدن به حالت k

- go backwards using: $J_k(x_k) = \min_{u_k \in U_k(x_k)} \underset{w_k}{E} \left[g_k(x_k, u_k, w_k) + J_{k+1}(\underbrace{f_k(x_k, u_k, w_k)}_{x_{k+1}}) \right]$

(for $k = N-1, N-2, \dots, 0$)

ابتدا بهترین تصمیم بین x_{N-1} و x_N به دست آید و ...

Cost بعد از حالت k

بر حسب x_k و u_k نوشته می‌شود تا بتوان بهینه‌سازی انجام داد

/ /

Example

System: $x_{k+1} = (1-a)x_k + a u_k \qquad k=0,1 \qquad 0<a<1$

Cost: $\underbrace{r(x_2-T)^2}_{} + \underbrace{u_0^2 + u_1^2}_{} \qquad r>0 \qquad \Rightarrow u_0, u_1 = ?$

هدف رسیدن دما به T است (Cost نهایی)   Cost حرارت دادن (Cost بین state ها)

$\Rightarrow$ DP Algorithm:

$$J_2(x_2) = r(x_2-T)^2$$

$$J_1(x_1) = \min_{u_1}\left[u_1^2 + r(\overbrace{(1-a)x_1 + a u_1}^{x_2} - T)^2\right]$$

$$J_0(x_0) = \min_{u_0}\left[u_0^2 + J_1(\underbrace{(1-a)x_0 + a u_0}_{x_1})\right]$$

end

* یادگیری تقویتی

← یک مسئله بهینه‌سازی دینامیکی داریم

← رابطه‌ی بسته و لزوماً قطعی از تابع هدف در دسترس نیست

← رابطه بسته و لزوماً قطعی از مدل دینامیک (تغییر حالت) نداریم

همچنان می‌توان از برنامه ریزی پویا استفاده کرد

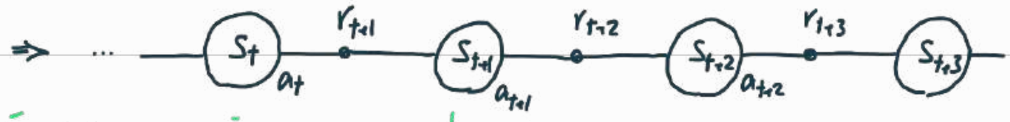
← حالت نهایی مسئله لزوماً مشخص نیست / لزوماً حالت نهایی وجود ندارد

ممکن است در محیطی با state های زیاد باشیم و قرار باشد صرفاً در آن حرکت کنیم (مقصد نداشته باشیم)

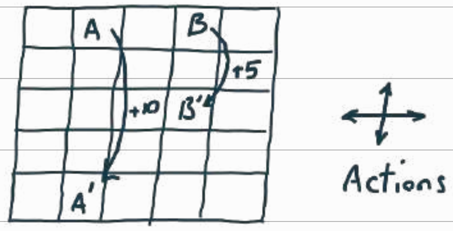
← تعداد مراحل تصمیم‌گیری می‌تواند نامعلوم یا نامحدود باشد

← فضای حالت یا فضای تصمیم می‌تواند به قدری بزرگ باشد که استفاده از DP را بسیار مشکل کند (Curse of dimensionality)

reward (می‌تواند توزیع داشته باشد و تصادفی باشد)



state حاصل از تصمیم a_t در S_t هم می‌تواند تصادفی باشد



→ Agent با رسیدن به A به A' و با رسیدن به B به B' پرش می‌کند و reward متناسب دریافت می‌کند

→ Agent در محیط حرکت می‌کند و انتظار می‌رود سود بلندمدت خود را max کند (حالت نهایی نداریم)

فرایند تصمیم مارکفی (MDP):

فرض می‌شود محیط خاصیت مارکفی دارد: حالت و پاداشی که عامل در مرحله $t+1$ به دست می‌آورد تنها به حالت و کنش (action) عامل در زمان t بستگی دارد

$$Pr\{S_{t+1} = s', r_{t+1} = r \mid S_t, a_t, r_t, S_{t-1}, a_{t-1}, \dots, r_1, S_0, a_0\}$$
$$= Pr\{S_{t+1} = s', r_{t+1} = r \mid S_t, a_t\}$$

اطلاعات موجود از مسئله:

احتمال transition از s به s' تحت عمل a : $P^a_{ss'} = Pr\{S_{t+1} = s' \mid S_t = s, a_t = a\} \quad \forall s,s' \in S,\ a \in A(s)$

مدل دینامیک مسئله / اطلاعات Transition حالت‌ها

(لزوماً Explicit نیست)

متوسط پاداش تحت عمل a در s رفتن به s': $R^{a}_{ss'} = E[r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'] \quad \forall s, s' \in S, a \in A(s)$

مثل تابع هدف

در هر مرحله عامل باید نناظری بین حالتی که در آن قرار گرفته و احتمال انتخاب هر عمل ممکن برقرار کند. این تناظر استراتژی یا سیاست نام دارد که همان تعبیر بهینه سازی است.

$\pi_t(s, a) = Prob(a_t = a \text{ when } s_t = s)$

عامل باید سیاست خود را طوری تعیین کند تا به هدف خود که بیشینه کردن مجموع پاداش هایی است که در طول زمان به دست می آورد برسد.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

$0 < \gamma < 1$: Discount Factor → reward های آینده ارزش کمتری پیدا می کنند

$\gamma = 0$ → عامل miopic است و تنها پاداش لحظه ای را می بیند

$\gamma = 1$ → حل کامل مسئله بهینه سازی

γ می تواند ویژگی Cognitive عامل باشد، یا معنی واقعی داشته باشد (مثل افت ارزش پول ناشی از تورم)

در این حالت تابع هدف با وجود γ دقیق است

میزان ارزش یک حالت به پاداش هایی که با شروع از آن حالت انتظار می رود گفته می شود.

ارزش حالت : $V^{\pi}(s) = E_{\pi}[R_t \mid s_t = s] = E_{\pi}\left[\sum_{k=0} \gamma^k r_{t+k+1} \mid s_t = s\right]$

متوسط Reward

متوسط گیری روی π
(π هم تصادفی است)

ارزش حالت S تحت سیاست π

/ /

میزان ارزش یک عمل در حالت به پاداش هایی که با شروع از آن حالت با اتخاذ آن عمل انتظار می رود گفته می شود

ارزش state-action : $Q^\pi(s,a) = E_\pi[R_t \mid s_t = s, a_t = a] = E_\pi[\sum_{k=0} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a]$

ارزش انتخاب a در حالت S تحت سیاست π

(action های مراحل بعد طبق π انتخاب می شوند)

← هدف Max کردن این توابع با تنظیم درست π است

$V^\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \dots \mid s_t = s] = E_\pi[r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s]$

$= \sum_a \pi(s,a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]$ ← نحوه محاسبه $V^\pi(s)$ با پاداش مدل سیستم

$Q^\pi(s,a) = E_\pi[r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a] = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')]$

رابطه های به دست آمده بسته نیستند و بازگشتی هستند ← نمی توان به سادگی بهینه سازی کرد

سیاست حریصانه (greedy) : انتخاب عملی که متوسط پاداش های مورد انتظار را بیشینه می کند با احتمال 1 و بقیه با احتمال صفر انجام می شود

$\pi^*(s,a) = \underset{a}{\mathrm{Argmax}}\, Q(s,a)$

اگر ارزش $V^\pi(s')$ معلوم باشد آنگاه استراتژی فوق بهینه است چراکه متوسط پاداش های مورد انتظار را بیشینه می کند

سیاست greedy بهینه

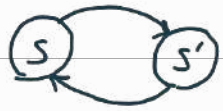
رابطه بهینگی بلمن (اصل برنامه ریزی پویا) :

$V^*(s) = \underset{a}{\mathrm{Max}}\, E[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a]$

$= \underset{a}{\mathrm{Max}} \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^*(s')]$

$= \underset{a}{\mathrm{Max}}\, Q^*(s,a)$

/ /

وابستگی ها ممکن است به شکل دور برقرار باشد : 

← معادله بلمن در صورتی منجر به پاسخ بهینه می شود که ارزش state ها تغییر نکند (V(S) ها را باید داشته باشیم)

← هم π و هم V(S) در معادله بالا مجهول است ← نمی توان به سادگی بهینه سازی کرد

- برنامه ریزی پویای تقریبی

ارزش بازگشتی براساس سیاست (Policy Iteration/PI) :

ارزش یابی سیاست :

$$V^{\pi}_{k+1}(S) = E_{\pi}\left[r_{t+1} + \gamma V^{\pi}_{k}(S_{t+1}) \mid S_t = S\right]$$

$$= \sum_{a} \pi(S,a) \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}_{k}(s')\right]$$

یک مقداردهی اولیه می کنیم و به صورت iterative به روزرسانی می کنیم

وقتی ارزش ها همگرا شد سیاست را بهبود می دهیم

بهبود سیاست :

$$\pi_{k+1}(S) = \arg\max_{a} Q^{\pi_k}(S,a) = \arg\max_{a} \sum_{s'} P^{a}_{ss'}\left[R^{a}_{ss'} + \gamma V(s')\right]$$

زمانی که سیاست هم تغییر نکرد رابطه بهینگی بلمن برقرار است و به سیاست بهینه همگرا شده ایم

الگوریتم : 1 Initialization: $V(s) \in \mathbb{R}$ and $\pi(s) \in A(s)$ $\forall s \in S$, $V(terminal) = 0$

2 Policy Evaluation

Loop

$\Delta \leftarrow 0$

Loop for each $s \in S$

$v \leftarrow V(s)$

$$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V(s)]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

Until $\Delta < \theta$ (θ: small positive number determining the accuracy of estimation)

③ Policy Improvement

Policy-stable $\leftarrow$ true

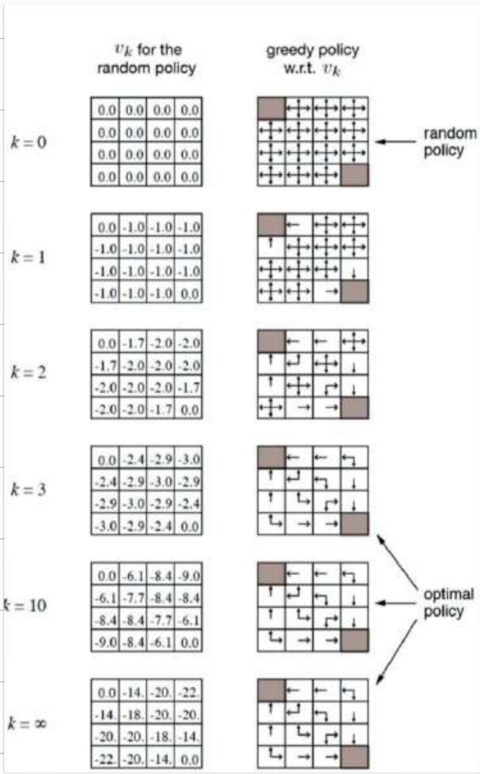
For each $s \in S$

$\quad$ old-action $\leftarrow \pi(s)$

$\quad \pi(s) \leftarrow \arg\max_a \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V(s')]$

$\quad$ if old-action $\neq \pi(s)$ then Policy-stable $\leftarrow$ false

if Policy-stable then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to ②



→ هر حرکت پاداش ۱- دارد غیر از حرکتی که به خانه‌های مشخص شده برسد

→ ستون سمت چپ نشان دهنده Policy Evaluation در هر Iteration است

→ ستون سمت راست greedy Policy متناظر با مقادیر به دست آمده در سمت چپ است

→ مشاهده می‌شود با سه تکرار به پاسخ optimal می‌رسیم

صبر کردن تا همگرایی کامل حلقه Policy Evaluation لازم نیست

روش بازگشتی براساس ارزش (Value Iteration/VI):

ترکیبی است از دو مرحله روش PI (خود تابع ارزش هر بار براساس سیاست greedy به روزرسانی می‌شود)

$$V_{k+1}(s) = \max_a E_\pi \{ r_{t+1} + \gamma V_k^\pi(S_{t+1}) \mid S_t = s, a_t = a \}$$

$$= \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V_k^\pi(s')] = \max_a Q_{k+1}(s,a)$$

← هر وقت همگرا شود به معادله بلمن رسیده‌ایم

در مسائل مختلف PI یا VI هر کدام می‌توانند عملکرد بهتر در سرعت همگرایی بهینه داشته باشند.

الگوریتم: Initialize $V(s)$ $\forall s \in S^+$, $V(\text{terminal}) = 0$

Loop

  $\Delta \leftarrow 0$

  Loop for each $s \in S$

    $v \leftarrow V(s)$

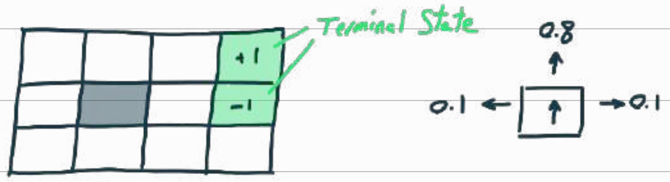
    $V(s) \leftarrow \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')]$

    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

Until $\Delta < \theta$

Output a deterministic Policy $\pi \approx \pi_*$ such that $\pi(s) = \arg\max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')]$

Example



← در 20% موارد حرکت به سمتی که انتخاب می‌کنیم نیست

/ /

Value Iteration :

0	0	0	+1
0		0	-1
0	0	0	0

→

0	0	0.8	+1
0		0	-1
0	0	0	0

→

0	0.64	0.8	+1
0		0.54	-1
0	0	0	0

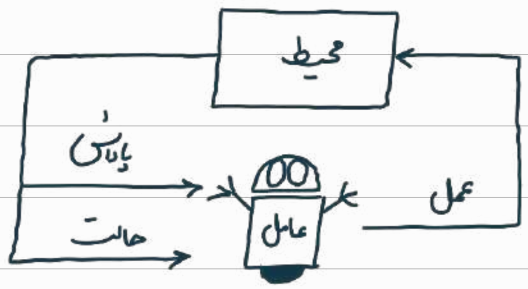
→ ... → تا همگرایی مقادیر به روز رسانی را ادامه می‌دهیم

end

• یادگیری تقویتی : بدون مدل و با تعامل با محیط

در خیلی مسائل مدل سیستم در دست نیست ($P_{ss'}^a$ و $R_{ss'}^a$ را نداریم)

← عامل با زندگی و تعامل با محیط و دریافت اطلاعات عددی از حالت و پاداش سعی در بیشینه کردن سود خود خواهد داشت.



در خیلی موارد انسان‌ها هم به همین صورت تصمیم‌گیری می‌کند و با گرفتن Feed back تصمیم خود را اصلاح می‌کند

روش مونت کارلو :

در این روش با گذر از حالت‌ها و دریافت پاداش‌ها سعی در بهبود تخمین خود از $Q(s,a)$ داریم

$$\cdots - S_t \xrightarrow[a_t]{r_{t+1}} S_{t+1} \xrightarrow[a_{t+1}]{r_{t+2}} S_{t+2} \xrightarrow[a_{t+2}]{r_{t+3}} S_{t+3} \xrightarrow[a_{t+3}]{} \cdots$$

در هر اپیزود i با شروع از یک حالت و دنبال کردن یک سیاست مجموع پاداش‌های دریافتی محاسبه می‌شود

$$Return_i = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots + \gamma^{k-1} r_k$$

میانگین پاداش‌های دریافت شده در اپیزودهای مختلف و با شروع از جفت عمل-حالت (s,a) و در k حرکت به عنوان تخمینی از $Q(s,a)$ در نظر گرفته می‌شود

$$Q(s,a) \leftarrow \underset{i}{Average}(Return\ i)$$

پس اگر این کار به تعداد زیاد انجام شود مقادیر Q به مقادیر واقعی نزدیک می‌شوند. به این صورت مسئله حل می‌شود

Exploration-Exploitation Balance: ← هم می‌خواهیم در محیط تجربه کسب کنیم و هم می‌خواهیم بهینه‌سازی انجام دهیم

← اگر به اندازه کافی تجربه نداشته باشیم مقادیر Q به درستی به دست نمی‌آیند

← سیاست عامل بین کاوش و بهره‌برداری تعادل ایجاد می‌کند

← اوایل تصمیم‌گیری باید سیاست به نحوی باشد که بیشتر کاوش و کسب تجربه را انجام دهد

و در نهایت سیاست باید به سمتی برود که عمل بهینه را انتخاب کند

← سیاست هم تعیین بهینه‌سازی است و هم عامل مشاهده محیط

ε-greedy:

$$\pi(s,a) = \begin{cases} 1-\varepsilon + \dfrac{\varepsilon}{|A(s)|} & \text{if } a = a^* \\[2ex] \dfrac{\varepsilon}{|A(s)|} & \text{if } a \neq a^* \end{cases}$$

ε در ابتدا بزرگ در نظر گرفته می‌شود (تصمیم‌گیری تصادفی / $\varepsilon \to 1$) و با همگرا شدن مقادیر Q آن را به سمت صفر می‌بریم (تصمیم‌گیری greedy و بهینه)

Softmax:

$$\pi(s,a) = \frac{e^{\frac{Q(a)}{\tau}}}{\sum_{b \in A(s)} e^{\frac{Q(b)}{\tau}}}$$

T: Temperature

T بزرگ کسرها را به سمت $\frac{1}{n}$ می‌برد ($\lim_{T \to \infty} \frac{Q(a)}{T} = 0$) و منجر به تصمیم‌گیری تصادفی می‌شود

/ /

τ وقتی به سمت صفر برود Q ها تعیین کننده گره ها می شوند (با تقسیم گره ها به $e^{\frac{Q(a)}{\tau}}$ می توان دید در این حالت احتمال گره مربوط به عمل بهینه 1 و احتمال سایر گره ها صفر می شود)

به سیاست های مثل ε-greedy و softmax ، soft policy گفته می شود

الگوریتم مونت کارلو : Initialize : $\pi \leftarrow$ an arbitrary ε-greedy policy

$Q(s,a) \in \mathbb{R}$ (arbitrary) $\forall s \in S, a \in A(s)$

$Returns(s,a) \leftarrow$ empty list $\forall s \in S, a \in A(s)$

Repeat forever (for each episode):

T بار تغییرپذیری در محیط تحت π

Generate an episode following π : $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode $t = T-1, T-2, \dots, 0$:

$G \leftarrow \gamma G + R_{t+1}$ ← محاسبه Return

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow average(Returns(S_t, A_t))$ ← به روزرسانی Q

$A^* \leftarrow \arg\max_a Q(S_t, a)$ ← بهترین action های برابر انتخاب به همراه انتخاب می شود

For all $a \in A(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1-\varepsilon+\varepsilon/|A(S_t)| & \text{if } a = A^* \\ \varepsilon/|A(S_t)| & \text{if } a \neq A^* \end{cases}$$

به روزرسانی Policy

مقایسه با approximate DP : - MC با تعامل مستقیم با محیط عمل می کند (در حالتی که مدلی از محیط وجود ندارد)

- ساده تر شدن مدلسازی و شبیه سازی

- امکان توجه بیشتر به مجموعه مشخصی از حالت ها در مقابل حالت های کم ارزش تر

← می توان از state هایی که می دانیم مهم ترند ، episode های بیشتری شروع کرد

- تخمین های هر حالت مستقل از حالت های دیگر است (برخلاف DP که تخمین هر حالت براساس حالت دیگر است) ← دینامیک مسئله وارد محاسبات نمی شود

• روش های تفاضل زمانی (TD)

ایدهٔ اصلی براساس ترکیب روش برنامه ریزی پویا و روش مونت کارلو است

← مانند MC می توان بدون داشتن اطلاعات از محیط و با تعامل با آن عمل یادگیری را انجام داد

← براساس ایدهٔ DP در هر مرحله تخمین خود را براساس تخمین قبلی و مشاهدات جدید بهبود می بخشد و مانند MC منتظر رسیدن به حالت نهایی برای بهبود تخمین نمی شود (= بوت استرپ / Bootstrap)

← از خطای تخمین مدل عامل در حالت کنونی و مدل عامل پس از انجام عمل و قرارگرفتن در حالت جدید، جهت بهبود توابع ارزش استفاده می شود:

$$V(S_t) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid S_t = S\right] = E\left[r_{t+1} + \gamma V(S_{t+1}) \mid S_t = S\right]$$

learning rate: $0<\alpha<1$

مشاهده

$$\xrightarrow{\text{Update تخمین}} V_{t+1}(S_t) = V_t(S_t) + \alpha \cdot \mathcal{E}_t = V_t(S_t) + \alpha \cdot \underbrace{\big[\underbrace{r_{t+1} + \gamma V_t(S_{t+1})}_{\hat{V}(S_t)} - V_t(S_t)\big]}_{\mathcal{E}_t}$$

خطای تفاضلی

با گرفتن پاداش از محیط

تخمین قبلی اصلاح می شود

$$\begin{cases} \hat{V}(S_t) > V_t(S_t) \Rightarrow \mathcal{E}_t > 0 \Rightarrow \text{افزایش تخمین} \\ \hat{V}(S_t) < V_t(S_t) \Rightarrow \mathcal{E}_t < 0 \Rightarrow \text{کاهش تخمین} \end{cases}$$

همین روابط را می توان برای $Q(s,a)$ هم نوشت.

← به این صورت در هر مرحله تخمین Update می شود و نیاز نیست مانند MC تا انتهای episode پیش برویم

انواع روش های TD (Temporal Difference) $\left\{ \begin{array}{l} \text{SARSA} \\ \text{Q-Learning} \\ \text{Actor-Critic} \end{array} \right.$

روش SARSA :

SARSA :

عامل عمل a را براساس سیاست خود انتخاب می‌کند (Soft Policy مثل ε-greedy یا Softmax) پاداش r را دریافت می‌کند و به حالت s′ می‌رود و دوباره براساس سیاست خود عمل a′ را انجام می‌دهد. پس از این مراحل تابع ارزش-عمل (state-action) به شکل زیر به‌روز می‌شود:

$$Q(s,a) \leftarrow Q(s,a) + \alpha.[\underbrace{r + \gamma Q(s',a')}_{\hat{Q}_t(s_t)} - Q(s,a)]$$

→ سیاست تابعی از ارزش Q است (Softmax/ ε-greedy)

الگوریتم :

Initialize $Q(s,a)$ arbitrary $\forall s \in S^+$, $a \in A(s)$ except that $Q(\text{terminal}, \cdot) = 0$ ← چون از terminal State به‌بعد actionی اعمال نمی‌شود

Loop for each episode

  Initialize S

  Choose A from S using Policy derived from Q (e.g. ε-greedy)

  Loop for each step of episode:

    Take action A, observe R, S' → نیاز به رفتن تا انتهای episode نیست (برخلاف MC)

    Choose A' from S' using policy derived from Q (e.g. ε-greedy)

    $Q(S,A) \leftarrow Q(S,A) + \alpha\, [R + \gamma Q(S',A') - Q(S,A)]$

    $S \leftarrow S'$, $A \leftarrow A'$

  Until S is terminal

این روش on Policy است (از action ی که انجام می‌دهیم (A') برای Update استفاده می‌کنیم)

روش Q (Q-Learning): شبیه SARSA است با این تفاوت که مقدار تابع ارزش-عمل به ازای عمل حریصانه Update می‌شود و به‌روزرسانی قبل از اتخاذ عمل بعدی (a') انجام خواهد شد

off Policy

بعد از s'، action ی که بیشترین ارزش را ایجاد می‌کند در نظر گرفته می‌شود

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \underbrace{\max_{a} Q(s',a')}_{} - Q(s,a)]$$

برخلاف SARSA که a' براساس Policy انتخاب می‌شد (واقع گرایانه)
در Q-Learning به صورت greedy درنظر گرفته می‌شود (خوش بینانه)

← سیاست به صورت تابعی از تابع ارزش است (Softmax , ε-greedy)

الگوریتم:

Initialize $Q(s,a)$ arbitrary $\forall s \in S^+, a \in A(s)$ except that $Q(terminal, \cdot) = 0$

Loop for each episode

    Initialize S

    Loop for each step of episode:

        Choose A from S using Policy derived from Q (e.g. ε-greedy)

        Take action A, observe R, S'

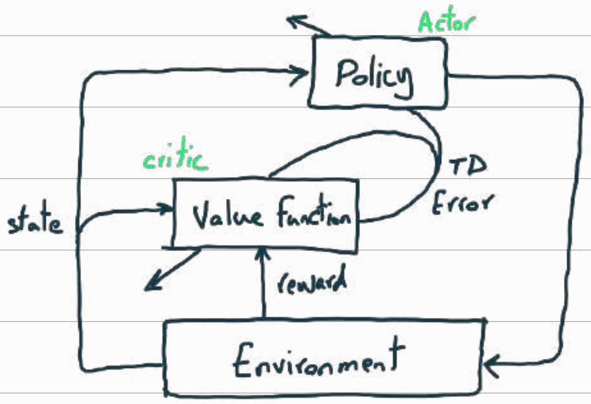
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \underbrace{\max_{a} Q(S',a)}_{\text{SARSA تفاوت با}} - Q(S,A)]$

        $S \leftarrow S'$

    Until S is terminal

روش عملگر - نقاد (Actor-Critic):

به جای ارزش state-action ، ارزش state ها تخمین زده می‌شوند



$$\delta_t = r_{t+1} + \gamma V(S_{t+1}) - V(S_t) \leftarrow \text{TD Error}$$

توجه دارد به ارزش state بعدی (خارج از action بعد از آن) برای همین action فعلی ارزیابی می‌شود.

از TD Error برای تصحیح Policy و تعیین ارزش استفاده می‌شود:

$$\pi_t(s,a) = Pr\{a_t = a \mid s_t = s\} = \frac{e^{P(s,a)}}{\sum_b e^{P(s,a)}} \quad , \quad P(s_t,a_t) \leftarrow P(s_t,a_t) + \beta\delta_t$$

پارامتری است که مطلوبیت انتخاب a در s را نشان می‌دهد

$$V_{t+1}(S_t) = V_t(S_t) + \alpha\delta_t = V_t(S_t) + \alpha\left[r_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)\right]$$

← وجود مدل برای سیستم: روش برنامه‌ریزی پویای تقریبی

سیاست بهینه (greedy Policy): $\pi^*(s) = \underset{a}{\text{Argmax}}\, Q(s,a)$

عدم وجود مدل برای محیط: روش‌های MC و TD

سیاست جهت یادگیری مدل و بهینه‌سازی: ε-greedy و softmax

مدل محیط (در قالب Q) هم در کنار پیدا کردن سیاست بهینه یادگرفته می‌شود

- یادگیری تقویتی پیوسته (Continues RL)

تا به حال مدل محیط را به صورت Table مدل می‌کردیم:

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
s_0	490	490	490	700	490	490	490	490
s_1	490	490	700	490	490	490	490	490
s_2	490	490	490	700	700	490	490	490
s_3	490	490	700	1000	700	700	490	490
s_4	490	490	490	700	700	700	490	490
s_5	490	490	700	700	1000	700	700	490
s_6	490	490	490	490	490	700	490	490
s_7	490	490	490	490	700	1000	700	490
s_8	700	490	490	490	700	700	1000	700
s_9	700	490	490	490	490	700	700	1000
s_{10}	700	490	490	490	490	490	490	490
s_{11}	490	490	490	490	490	700	490	490
s_{12}	490	490	490	490	490	700	700	490
s_{13}	490	490	490	490	490	700	700	700
s_{14}	490	490	490	490	490	490	700	700
s_{15}	490	490	490	490	490	490	490	700

← مقادیر خانه‌ها تعیین می‌شد

با زیاد شدن تعداد state ها و action ها جدول بسیار بزرگ می‌شود و چون برای تخمین درست Value باید هر کدام را چند بار دیده باشیم یادگیری مشکل می‌شود

← جدول را تخمین می‌زنیم (ورودی action و state و خروجی Value)

← استفاده از یک Function Approximator

پارامتر تخمین (مثل وزن شبکه عصبی)

→ $Q(s,a;w)$, $\pi(s,a;\theta)$ ⟶ Policy را Parametrize کردن

مسئله می‌تواند گسسته باشد که آن را به صورت پیوسته تخمین می‌زنیم، یا فضای state و action خودش پیوسته باشد

⇐ احتمال انتخاب action ها به شکل smooth تغییر می‌کند (در حالت گسسته با تغییر ارزش ها ممکن بود شاهد تغییر زیاد Policy در یک iteration شویم)

⇐ در صورت داشتن دانش اولیه (Prior Knowledge) می‌توانیم از آن استفاده کنیم (در شکل یا پارامتر تابع)

⇐ Policy به انواع روش ها می‌تواند Parametrize شود (با یک general function approximator) ⟵ محدود به ε-greedy و Softmax و ... نیستیم

⇐ برای فضای action و فضای state پیوسته می‌تواند مورد استفاده قرار گیرد

← روش های گسسته با زیاد شدن state و action دچار Curse of dimensionality می‌شد و برای فضاهای پیوسته قابل استفاده نبود

⇐ در حالت گسسته تبدیل حالت stochastic به deterministic باید ε یا Temperature را خودمان تغییر می‌دادیم ولی در این حالت پارامتر θ خود به خود learn می‌شود

Policy Gradient:

$\tau \in \varepsilon$

در روش‌های گذشته Policy ثابت بود و اگر پارامتری داشت مقدار و نحوهٔ تغییر آن مشخص بود. در حالت پیوسته خود Policy را یاد می‌گیریم.

← می‌خواهیم بهترین θ را برای $\pi_\theta(s,a)$ پیدا کنیم ← باید تابع هدف از π_θ را داشته باشیم

برای محیط‌های episodic می‌توانیم start Value را در نظر بگیریم:

ارزش state اولیه

$$J_1(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1]$$

در محیط‌های Continual می‌توانیم average Value را در نظر بگیریم:

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s)\, V^{\pi_\theta}(s)$$

توزیع Markov Chain برای π_θ ← احتمال مشاهده s تحت سیاست π_θ

فرض می‌شود این توزیع برای یک سیاست ثابت است

در به‌روزرسانی θ از $d^{\pi_\theta}(s)$ به صورت مستقیم استفاده نمی‌شود چون مدل محیط را نداریم

← چون می‌خواهیم از توابع هدف مشتق بگیریم آنها را به صورت غیر بازگشتی می‌نویسیم

همچنین می‌توانیم Average Reward Per time-step را در نظر بگیریم:

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) R_s^a$$

← استفاده از متوسط Reward یک step

با مشتق‌گیری از هرکدام از توابع هدف می‌توانیم θ را Update کنیم

$$\nabla_\theta \pi_\theta(s,a) = \pi_\theta(s,a)\,\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} = \pi_\theta(s,a)\,\underbrace{\nabla_\theta \log \pi_\theta(s,a)}_{\text{Score Function}}$$

→ این رابطه در مشتق‌گیری از J که برحسب π است نسبت به θ به ما کمک می‌کند

function approximator: $\overbrace{\phi(s,a)^T\theta}^{\text{Linear}} \xrightarrow{\text{Softmax}} \pi_\theta(s,a) = \dfrac{e^{\phi(s,a)^T\theta}}{\sum_b e^{\phi(s,b)^T\theta}} \propto e^{\phi(s,a)^T\theta}$

Kernel / Feature نمایش

$$\Rightarrow \text{Score Function: } \nabla_\theta \log \pi_\theta(s,a) = \underbrace{\phi(s,a) - E_{\pi_\theta}[\phi(s,\cdot)]}$$

مقدار متوسط ϕ از $\phi(s,a)$ کم می‌شود ← حالت score دارد

با یک one-step MDP شروع می‌کنیم:

می‌خواهیم پاداش را بیشینه کنیم

با $s \sim d(s)$ شروع می‌کنیم، با یک مرحله گرفتن $r = R_{s,a}$ حرکت به پایان می‌رسد

نیاز به مشتق‌گیری از d نیست

برای سادگی فرض می‌شود d تابعی از π نیست

$$J(\theta) = E_{\pi_\theta}[r] = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a) R_{s,a}$$

$$\Rightarrow \nabla_\theta J(\theta) = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(s,a)\, \underbrace{\nabla_\theta \log \pi_\theta(s,a)\, R_{s,a}} = E_{\pi_\theta}[\underbrace{\nabla_\theta \log \pi_\theta(s,a)}_{\text{score function}}\, r]$$

امید ریاضی نسبت به این عبارت گرفته شده

(شرطی روی s و a)

Expectation را می‌توان به صورت تجربی حساب کرد (مثل Stochastic Gradient Descent)

Policy Gradient Theorem: برای هر سیاست Differentiable به شکل $\pi_\theta(s,a)$ و هر تابع هدف $J = J_1$ یا $J = J_{avR}$ یا $\frac{1}{1-\gamma} J_{avV}$، گرادیان به شکل زیر به دست می‌آید:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$$

→ Multi step MDP تعمیم به

→ جایگزینی پاداش لحظه‌ای (r) با long-term Value (Q)

/ /

هر state-action یک Sample است

: Monte Carlo Policy Gradient ← استفاده از Stochastic gradient ascent

← استفاده از Policy gradient theorem

← از v_t به عنوان یک Unbiased Sample از $Q^{\pi_\theta}(s_t, a_t)$ استفاده می‌کنیم

فعلاً فقط π را Parametrize کرده‌ایم

$$\rightarrow \Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

الگوریتم:

```
function REINFORCE
    initialise θ arbitrary
    for each episode {s1, a1, r2, ..., sT-1, aT-1, rT} ~ πθ do
        for t=1 to T-1 do
            θ ← θ + α ∇θ log πθ(st, at) vt
        end for
    end for
    return θ
end function
```

→ تا همگرا شدن θ تکرار می‌کنیم

این روش همچنان از جدول برای Q استفاده می‌کند ← واریانس بالا دارد

برای Q هم از Function Approximator استفاده می‌کنیم: $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$ ← Q هم Parametrize می‌شود

← Critic: پارامترهای $Q_w(s,a)$ را یاد می‌گیرد

← Actor: پارامترهای $\pi_\theta(s,a)$ را یاد می‌گیرد

Q_w تخمینی از Q است

الگوریتم‌های Actor-Critic از approximate Policy gradient استفاده می‌کنند:

$$\nabla_\theta J(\theta) \approx E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)] \Rightarrow \Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a) \underbrace{Q_w(s,a)}_{\text{این تخمین را به ما می‌دهد Critic}}$$

Action-Value Function Approximation:

تقریب تابع Action-Value (به جای Q از q استفاده می‌کنیم): $\hat{q}(S,A,w) \approx q_\pi(S,A)$

تابع هدف (متوسط مربعات خطا): $J(w) = E_\pi[(q_\pi(S,A) - \hat{q}(S,A,w))^2]$

چون مدل را نداریم از stochastic gradient descent استفاده می‌کنیم:

$$-\frac{1}{2}\nabla_w J(w) = (q_\pi(S,A) - \hat{q}(S,A,w))\nabla_w \hat{q}(S,A,w)$$

$$\Rightarrow \Delta w = \alpha(q_\pi(S,A) - \hat{q}(S,A,w))\nabla_w \hat{q}(S,A,w)$$

π را هم به صورت کامل نداریم ← Target

چون $q_\pi(S,A)$ را نداریم از تخمین آن استفاده می‌کنیم:

در MC از G_t استفاده می‌کنیم:

$$\Delta w = \alpha(G_t - \hat{q}(S_t,A_t,w))\nabla_w \hat{q}(S_t,A_t,w)$$

در TD(0) از $R_{t+1} + \gamma Q(S_{t+1},A_{t+1})$ استفاده می‌کنیم:

$$\Delta w = \alpha(R_{t+1} + \gamma \hat{q}(S_{t+1},A_{t+1},w) - \hat{q}(S_t,A_t,w))\nabla_w \hat{q}(S_t,A_t,w)$$

Actor-Critic Policy Gradient:

از linear value function approximator استفاده می‌کنیم: $Q_w(s,a) = \phi(s,a)^T w$

Critic: w را با TD(0) به‌روزرسانی می‌کند ← خطای TD کمینه می‌شود

Actor: θ را با Policy gradient به‌روزرسانی می‌کند

/ /

الگوریتم : function QAC

initialize s, θ → w را هم تصادفی اولیه می‌کنیم

Sample $a \sim \pi_\theta$

for each step do

Sample reward $r = R_s^a$, Sample transition $s' \sim P_s^a$ → مدل محیط را نداریم (با اجرای a
به s' می‌رسیم و r را دریافت می‌کنیم)

Sample action $a' \sim \pi_\theta(s', a')$

$\delta = \overbrace{r + \gamma Q_w(s', a')}^{\text{Target}} - Q_w(s, a)$ → TD error

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$

$w \leftarrow w + \beta \delta \underline{\phi(s, a)}$ → مشتق q نسبت به w

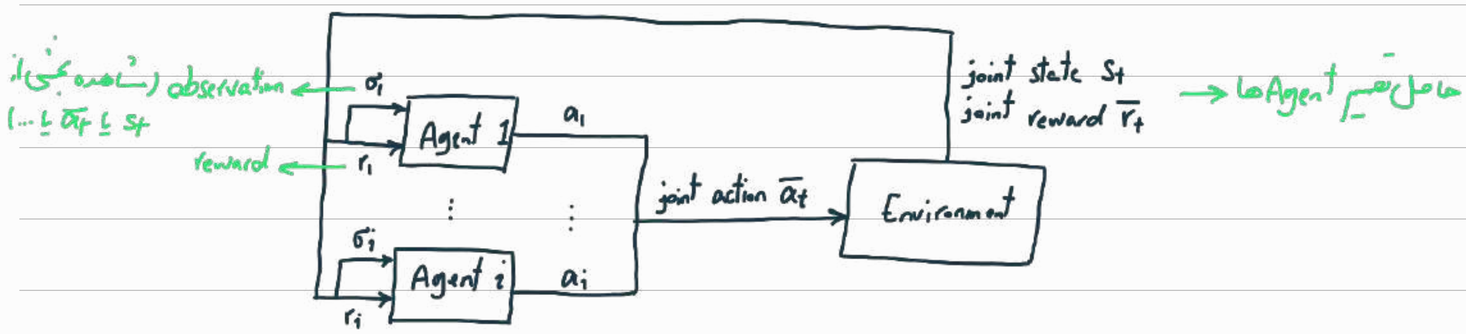
$a \leftarrow a'$, $s \leftarrow s'$

end for

end function

مطالعه بیشتر : اسلایدها درس David Silver

* یادگیری تقویتی چندعامله (Multi Agent Reinforcement Learning)



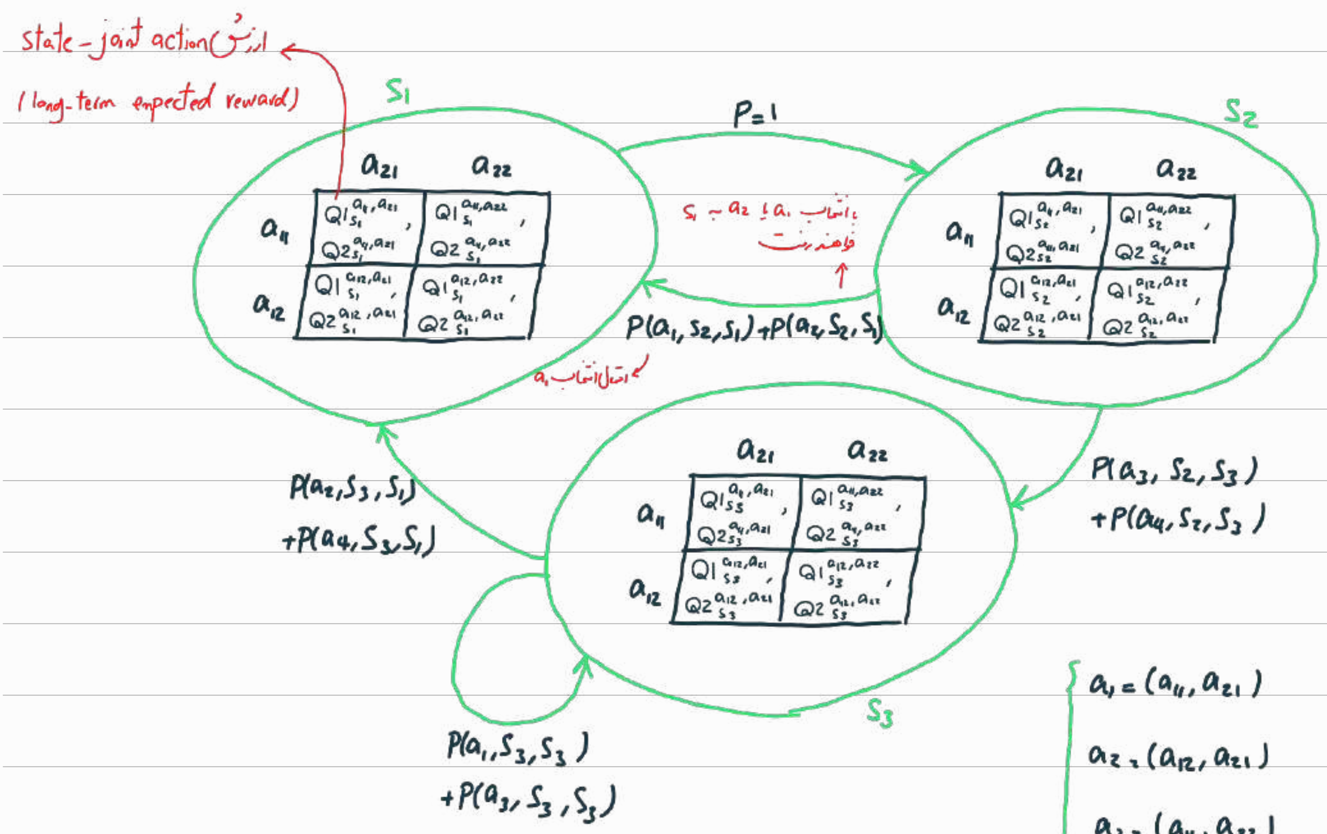
- هر عامل به دنبال Long term expected reward بیشینه است
- عامل‌ها می‌توانند selfish یا Cooperative باشند
- محیط می‌تواند دینامیکی باشد

Multi Agent MDP : $(N, S, \vec{A}, \vec{R}, T)$

- N: مجموعه عامل ها $(1,\dots,n)$
- S: مجموعه state ها (برای محیط غیر دینامیک یعنو دارد)
- $\vec{A} = A_1 \times \cdots \times A_n$
  - A_i: مجموعه action های عامل i
- $\vec{R} = R_1 \times \cdots \times R_n$
  - $R_i : S \times \vec{A} \to R$ → تابع پاداش برای فضای عامل i که به state و انتخاب هر عامل ها وابسته است.
  - $\Leftarrow$ Coupling
- $T : S \times \vec{A} \to \Pi(S)$ (فضای احتمالاتی برای state ها)
  - تابع transition (تصادفی) → transition با انتخاب هر عامل ها وابسته است
  - $\Leftarrow$ Coupling

به دلیل وجود Coupling به Multi Agent MDP ، Markov Game هم گفته می شود

برای مثال، اگر سه state و دو Agent داشته باشیم و هر Agent دو action داشته باشد:



$$
\begin{cases}
a_1 = (a_{11}, a_{21}) \\
a_2 = (a_{12}, a_{21}) \\
a_3 = (a_{11}, a_{22}) \\
a_4 = (a_{12}, a_{22})
\end{cases}
$$

/ /

← در هر state یک table (یا Matrix game) داریم ← به نوعی در هر state یک game بین Agent ها برقرار شده

joint Q-table

• Learning in Multi Agent MDP

ساده‌ترین راه نگاه به مسئله به صورت Single Agent MDP برای هر Agent است.

$$Q_i(s,a_i) \leftarrow (1-\alpha)Q_i(s,a_i) + \alpha\,[R_i(s,\vec{a}) + \gamma \underbrace{V_i(s')}_{\max_{a_i\in A_i} Q_i(s,a_i):\ \text{Q-Learning}}]$$

← تنها یک اندیس اضافه می‌شود

به این صورت هر عامل محیط را Passive نگاه می‌کند و این حقیقت که Q value تابع action عامل های دیگری نیز هستند را ignore می‌کند.

← محیط از دید عامل i، non-stationary به نظر می‌رسد و در نتیجه فرآیند یادگیری پیچیده‌تر می‌شود

برای در نظر گرفتن action بقیه، باید یک observation از action سایر عامل ها هم داشته باشد. به این صورت Q-table هر عامل extend می‌شود:

$$Q_i(s,\vec{a}) \leftarrow (1-\alpha)Q_i(s,\vec{a}) + \alpha\,[R_i(s,\vec{a}) + \gamma V_i(s')]$$

← Curse of dimensionality ؟

← تخمین $V_i(s')$ ؟

با استفاده از روش های on-Policy مثل SARSA می‌توان $V_i(s')$ را داشت: $V_i(s') \leftarrow Q_i(s',a')$

در غیر این صورت برای استفاده از روش های off-Policy مثل Q-Learning، چون action بقیه را می‌بینند باید یک تخمین (expectation/estimation) از action سایر عامل ها داشته باشد.

← قبل از دیدن action بعدی باید Update کند

← در حالت کلی هر عامل برای action selection نیاز به تخمین (expectation/estimation) از action دیگران دارد

(1994) minimax Q-Learning: برای عامل های Selfish (هر عامل به دنبال Max کردن سود خود است)
فرض می‌کنیم دو عامل داریم (به تعداد عامل های دیگر قابل تعمیم است)

$$V_1(s) \leftarrow \max_{P_1 \in \Pi(A_1)} \underbrace{\min_{a_2 \in A_2}} \sum_{a_1 \in A_1} P_1(a_1) Q_1(s,(a_1,a_2))$$

Probabilistic Policy

بدترین حالتی که عامل 2 می‌تواند به وجود بیاورد در نظر گرفته می‌شود (worst case)

Policy نیز می‌تواند با همین ایده Update شود

Belief Based Algorithm:

$$V_i(s) \leftarrow \max_{a_i} \underbrace{\sum_{a_{-i} \in A_{-i}}} P_i(s,a_{-i}) Q_i(s,(a_i,a_{-i}))$$

با فرض داشتن توزیع انتخاب action سایر عامل ها
میزان expected را در نظر می‌گیرد

محاسبه توزیع: هیستوگرام (فرکانسی)
KNN
هر Pdf estimator دیگر

برای انتخاب action نیز می‌توان به همین شکل عمل کرد

Friend-or-Foe (FoF) Algorithm: هر عامل، بعضی عامل های دیگر را دوست و بعضی را دشمن در نظر می‌گیرد

Friend: $V_1(s) \leftarrow \max_{a_1 \in A_1, a_2 \in A_2} Q_1(s,(a_1,a_2))$ → max max

Foe: $V_1(s) \leftarrow \max_{P_1 \in \Pi(A_1)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} P_1(a_1) Q_1(s,(a_1,a_2))$ → max min

دست های عامل در راستای افزایش Value با عامل همکاری می کند در صورتی که دشمنان در راستای کاهش Value باهم همکاری می کنند.

$$\Rightarrow V_i(s) \leftarrow \max_{\pi \in \Pi(X_1 \times \cdots \times X_k)} \; \min_{y_1,\dots,y_l \in Y_1 \times \cdots \times Y_l} \; \sum_{x_1,\dots,x_k \in X_1 \times \cdots \times X_k} \pi(x_1)\cdots\pi(x_k)\, Q_i[s, x_1, \dots, x_k, y_1, \dots, y_l]$$

← Action های دشمنان نیز می توانند Probabilistic باشند، یا این که Action های دوستان و دشمنان Deterministic باشند یا ...

ترکیب Cooperation و Competition

برای Action Selection هم به همین صورت می توان عمل کرد

- Fully Cooperative Tasks

عامل ها هدف مشترکی دارند (reward function مشترک دارند و می خواهند Common discount return را طی learning افزایش دهند)

Q-Learning → $$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha \left[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k) \right]$$

← مشکل Curse of dimensionality وجود دارد

greedy Policy → $$\bar{h}_i^*(x) = \arg\max_{u_i} \; \max_{u_1,\dots,u_{i-1},u_{i+1},\dots,u_n} Q^*(x,u)$$

همه friend هستند

چون تابع Value برای همه یکی است Q افزایش یابد

← در حالت های تساوی عامل i به صورت تصادفی انتخاب می کند، چون هر عامل به صورت تصادفی tie را break می کند ممکن است به پاسخ sub optimal برسیم

/ /

برای مثال:

Agent2

Agent1	2,2	3,3
	3,3	2,2

$\text{Agent1} \begin{cases} a_{11} \rightarrow a_{22} \rightarrow 3 \\ a_{12} \rightarrow a_{21} \rightarrow 3 \end{cases} \xrightarrow{\text{انتخاب تصادفی}} a_{11}$

$\text{Agent2} \begin{cases} a_{21} \rightarrow a_{12} \rightarrow 3 \\ a_{22} \rightarrow a_{11} \rightarrow 3 \end{cases} \xrightarrow{\text{انتخاب تصادفی}} a_{21}$

نتیجه 2 می شود! (بدترین نتیجه حاصل می شود!) ← این مشکل حتی با همگرایی کامل مقادیر Q نیز وجود دارد

Distributed Q-Learning: مشکل Curse of dimensionality و tie breaking را حل می کند
تنها در محیط های deterministic اثبات همگرایی دارد

فرض می کنیم هر Agent وجود Agent های دیگر را ignore می کند:

$$Q_{i,k+1}(x_k, u_{i,k}) = \max\{Q_{i,k}(x_k, u_{i,k}),\ r_{k+1} + \gamma \max_{u_i} Q_{i,k}(x_{k+1}, u_i)\}$$

← چون reward مشترک است و در محیط deterministic هستیم Learning ساده تر می شود
دشکلاتی که با ignore کردن سایر Agent ها به وجود می آمد را نداریم

← Q-Value تنها زمانی Update می شود که منجر به افزایش Q-Value شود (تا به حال پایین
Q-Value تلی وجدید را در نظر می گیریم)

← Policy (greedy) نیز در صورتی Update می شود که منجر به improvement در Q-Value ها شود

$$\bar{h}_{i,k+1}(x_k) = \begin{cases} u_{i,k} & \text{if } \max_{u_i} Q_{i,k+1}(x_k, u_i) > \max_{u_i} Q_{i,k}(x_k, u_i) \\ \bar{h}_{i,k}(x_k) & \text{otherwise} \end{cases}$$

← اگر انتخاب $u_{i,k}$ با دید greedy منجر به افزایش Q در iteration بعدی شود
خود u_i به عنوان action بهینه انتخاب می شود

← $u_{i,k}$ می تواند به هر شکلی انتخاب شده باشد (می تواند تصادفی باشد برای exploration)
وانتخاب یک action می تواند به معنی بهینه بودن احتمال انتخابش باشد

→ با تعریفی که تا به حال از
سیاست greedy
داشتیم متفاوت است

با این تعریف از سیاست greedy ، اگر انتخاب action تصادفی نباشد اولین action انتخاب شده تا انتها باقی می‌ماند! بنابراین باید در انتخاب randomness داشته باشیم.

در شرایطی که تابع reward ثبت باشد و $Q_{i,0} = 0 \ \forall i$ ، سیاست local عامل‌ها به سیاست global بهینه برای global Q همگرا می‌شود.

← Coupling تنها از طریق r_{k+1} در سیاست عامل‌ها اثر می‌گذارد ولی تحت شرایط گفته شده به global optimum همگرا می‌شود.

Indirect Coordination Methods: → JAL → FMQ

← action Selection عامل‌ها به سمتی که reward جمعی بهینه‌ای داشته باشد بایاس می‌شود ← حرکت به سمت Coordinated action Selection

← همچنان Agent ها به صورت local یاد می‌گیرند

← بایاس به سمت افزایش احتمال ارزش‌ها و reward های بهینه ، به کمک statistical data عمل‌های دیگر انجام می‌شود.

← action ها یا reward های گذشته

← حرکت به سمت global optimum انجام می‌شود

Joint Action Learners (JAL): برای task های static کاربرد دارد ← Single State

$$Q_i(u_i, u_{-i}) = (1-\gamma) Q_i(u_i, u_{i-1}) + \gamma r(u_i, u_{-i})$$ discount factor = 0

↑ Q-Value Update

براساس emperical model از سایر Agent ها عمل می‌کند (شبیه روش Bleaf Based)

$$\hat{\sigma}_j^i(u_j) = \frac{C_j^i(u_j)}{\sum_{\tilde{u}_j \in U_j} C_j^i(\tilde{u}_j)}$$

→ تعداد دفعاتی که عامل i عمل u_j را از عامل j دیده

→ احتمال اجرای عمل u_j توسط عامل j از دید عامل i

$$\Rightarrow \tilde{Q}_i(u_i) = \sum_{u_{-i}} Q_i(u_i, u_{-i}) \prod_{j\neq i} \hat{\sigma}^i_j(u_j)$$

local Q (فقط به u_i وابسته است)

$$\Rightarrow \tilde{Q}_i(u_i) = (1-\gamma)\tilde{Q}_i(u_i) + \gamma r(u_i)$$

نشان داده می‌شود که با این روش اگرچه عامل‌ها به صورت local یاد می‌گیرند اما پاسخ نهایی به پاسخ بهینهٔ global می‌رسد

Frequency Maximum Q-Value (FMQ): ← Deterministic Tasks / Static Tasks

از ابتدا Q-Value تک عامله در نظر گرفته می‌شود (هر عامل فرض می‌کند فقط خودش در محیط حضور دارد)

$$Q_i(u_i) = (1-\gamma)Q_i(u_i) + \gamma r(u_i, u_{-i})$$

$$\tilde{Q}_i(u_i) = Q_i(u_i) + \underbrace{\nu \underbrace{\frac{C^i_{max}(u_i)}{C^i(u_i)}}_{\text{احتمال مشاهدهٔ } r_{max}} r_{max}(u_i)}_{Bias}$$

$C^i_{max}(u_i)$ → تعداد دفعات مشاهده r_{max} با اعمال عمل u_i

$r_{max}(u_i)$ → بیشترین reward مشاهده شده بعد از عمل u_i

برای عامل i، انتخاب u_i منجر به نتایج مختلف می‌شود (به دلیل انتخاب سایر عامل ها). اثر انتخاب عامل های دیگر در احتمال مشاهدهٔ r_{max} کد شده است.

← Q-Value با توجه به احتمال r_{max} و مقدار آن Bias می‌شود

← به جای Q، learning براساس $\tilde{Q}$ اعمال می‌شود

اثبات همگرایی این روش به global optimum نیز وجود دارد

/ /

→ A Comprehensive survey of multiagent reinforcement learning ; Busonio, Lucian, Robert Babuska, 2008

- Multi-Agent Reinforcement Learning via Double Averaging Primal-Dual Optimization

  32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canada

  از کنفرانس های معتبر ML

Multi Agent RL

→ Policy evaluation via MARL → بخش جالبی RL ارزیابی سیاست بود

→ Using Neural Networks to avoid curse of dimentionality → استخراج ویژگی برای کوچک کردن فضای state

→ Proposing an optimization Problem to find the parameters of neural network

→ Reformulating the problem to become Convenient with model-free approaches

→ Reformulating the problem to become Convenient with distributed optimization approaches

→ Decomposition of the problem to N sub-Problems to enable agents to Cooperatively solve the optimization problem by preserving the privacy of the agents

Multi-Agent MDP: $(S, \{A_i\}_{i=1}^{N}, P^a, \{R_i\}_{i=1}^{N}, \gamma)$

state

action — $a := (a_1, \dots, a_N) \in A_1 \times \dots \times A_N$

state transition matrix — $P^a \in \mathbb{R}^{|S| \times |S|}$

local reward recieved by agent i — $R_i(s,a)$

discount factor — $\gamma \in (0,1)$

global reward function: $R_c(s,a) = N^{-1} \sum_{i=1}^{N} R_i(s,a)$ → می‌خواهیم Max شود

Collective Reward

احتمال انتخاب joint action a در state s

فرض می‌کنیم سیاست $\pi(a|s)$ به ما داده شده ، می‌خواهیم آن را ارزیابی کنیم

شرط reward در state s

تحت سیاست π

$$R_c^{\pi} := \frac{1}{N} \sum_{i=1}^{N} R_i^{\pi}(s) \quad \text{where} \quad R_i^{\pi}(s) := E_{a \sim \pi(\cdot|s)}[R_i(s,a)]$$

local reward

هر سیاست ثابت یک Markov Chain القا می‌کند که transition matrix آن به شکل زیر است:

$$[P^\pi]_{s,s'} = \sum_{a\in A} \pi(a|s)\cdot[P^a]_{s,s'} \longrightarrow$$ احتمال رفتن از s به s'

aperiodic: state ی وجود نداشته باشد که تنها در Period های خاص قابل دسترسی باشد (مثلاً هر سه time step)

irreducible: از هر state به همه state ها بتوان رسید (گراف Markov Chain، strongly Connected باشد)

اگر Markov Chain، aperiodic و irreducible باشد: یک توزیع احتمال stationary (μ^π) روی S القا می‌کند

$\mu^\pi : S \to [0,1] \implies \mu^\pi(s)$: احتمال بودن در $s\in S$ با گذشتن به اندازه کافی تحت π

← فرکانس visit کردن در بی‌نهایت / steady state

Value Function: $$V^\pi(s) := E\left[\sum_{p=1}^{\infty} \gamma^p R_c^\pi(s_p) \mid s_1 = s, \pi\right]$$

$\longrightarrow$ Bellman Equation: $$V^\pi = R_c^\pi + \gamma \overbrace{P^\pi V^\pi}^{E[V(s')]}$$

← بردار حاصل از تجمیع $V^\pi(s)$ ها ($V^\pi \in \mathbb{R}^{|S|}$)

← بردار حاصل از تجمیع $R_c^\pi(s)$ ها

expected transition matrix $\rightarrow [P^\pi]_{ss'} := E_\pi[P^a_{ss'}]$

$$\Rightarrow \begin{bmatrix} V^\pi(s_1) \\ \vdots \\ V^\pi(s_n) \end{bmatrix} = \begin{bmatrix} R_c^\pi(s_1) \\ \vdots \\ R_c^\pi(s_n) \end{bmatrix} + \gamma \begin{bmatrix} P^\pi_{s_1 s_1} & \cdots & P^\pi_{s_1 s_n} \\ \vdots & \ddots & \vdots \\ P^\pi_{s_n s_1} & \cdots & P^\pi_{s_n s_n} \end{bmatrix} \begin{bmatrix} V^\pi(s_1) \\ \vdots \\ V^\pi(s_n) \end{bmatrix}$$

← می‌خواهیم V^π ی به دست بیاوریم که در این معادله صدق کند

برای Value Function Approximation از استفاده می‌کنیم ← از Curse of dimensionality جلوگیری می‌شود (کاهش بعد)

← از تخمین V استفاده می‌کنیم

← فضا را پیوسته می‌کند

/ /

$\{ V_\theta(s) := \phi^T(s)\theta : \theta \in \mathbb{R}^d \}$ $\phi(s) : S \to \mathbb{R}^d$ → d ویژگی

Basis Function → می‌تواند شبکه عصبی باشد

به این صورت پارامتر θ کاهش پیدا می‌کند (کاهش بعد)

بردار ویژگی و تخمین Value Function را برای هم به صورت Compact می‌نویسیم:

$$\Phi := (\cdots ; \phi^T(s) ; \cdots) \in \mathbb{R}^{|S| \times d} \qquad V_\theta \in \mathbb{R}^{|S|} \quad \Rightarrow \quad V_\theta = \Phi\theta$$

← باید θ را طوری پیدا کنیم که $V_\theta \approx V^\pi$

↑ V که در معادله بلمن صدق می‌کند

Mean Square Projected Bellman Error

$$\Rightarrow MSPBE(\theta) := \frac{1}{2} \| \Pi_\Phi (\underbrace{V_\theta - \gamma P^\pi V_\theta - R_c^\pi}_{\text{bellman error : } e}) \|_D^2 + \rho \|\theta\|^2$$

$\tilde{e}$: تصویرسازی e برای زیرفضایی که بردار ویژگی ها در آن تعریف شده

$\{\Phi\theta : \theta \in \mathbb{R}^d\}$

در این فضا می‌توانیم بهینه سازی را انجام دهیم

$$\Rightarrow \frac{1}{2} E[\tilde{e}^T \tilde{e}] = \frac{1}{2} \sum_{s \in S} \tilde{e}^2(s) \underbrace{\mu(s)}_{\text{احتمال مشاهده } S} = \frac{1}{2} \tilde{e}^T \underbrace{D}_{\operatorname{diag}(\{\mu^\pi(s)\}_{s\in S})} \tilde{e} = \frac{1}{2} \|\tilde{e}\|_D^2$$

→ norm وزن‌دار

متوسط وزن‌دار

↑ ماتریس قطری

/ /