روش‌های بهینه‌سازی غیرخطی

دکتر بابک نجار اعرابی　　کاظم فولادی

دانشکده‌ی مهندسی برق و کامپیوتر
دانشگاه تهران
اردیبهشت ۱۳۸۶

تکثیر یا نقل قول کلی یا جزیی تنها با ذکر ماخذ مجاز است.

فهرست مطالب

۱ **مقدمه** ۳

۲ **بهینه‌سازی محلی** ۴

۲-۱ روش‌های بی‌نیاز از مشتق: جستجوی مستقیم ۴

۲-۱-۱ روش سیمپلکس ۴

۲-۱-۲ روش هوک و هِوِز ۶

۲-۲ روش‌های مبتنی بر گرادیان ۶

۲-۲-۱ روش تندترین شیب ۶

۲-۲-۲ روش نیوتون ۸

۲-۲-۳ روش شبه نیوتون ۹

۲-۲-۴ روش BFGS ۹

۲-۲-۵ روش گرادیان مزدوج ۱۰
۲-۲-۶ روش حداقل مربعات غیرخطی: روش گاوس ـ نیوتون . . . ۱۱
۲-۲-۷ روش حداقل مربعات غیرخطی: روش لِونبرگ ـ مارکوارد . . ۱۳

۳ بهینه‌سازی سراسری ۱۴
۳-۱ تکرار جستجوی محلی ۱۴
۳-۲ جستجو با جهت تصادفی ۱۴
۳-۳ تافت شبیه‌سازی‌شده ۱۵
۳-۴ الگوریتم ژنتیک ۱۶

فصل ۱

مقدمه

در بهینه‌سازی، هدف یافتن نقطه‌ی بهینه‌ی تابعی به شکل

$$y = f(\boldsymbol{x})$$

است که در آن $\boldsymbol{x}$ یک بردار nتایی به فرم

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$$

می‌باشد. معمولاً توابع f مورد علاقه‌ی ما خواص زیر را دارند:

- تعداد زیادی بهینه‌ی محلی دارند،
- نقاط بهینه‌ی آنها جواب تحلیلی ساده‌ای ندارد،
- باتغییر شرایط اولیه، بهینه‌های مختلفی پیدا می‌شود.

فصل ۲

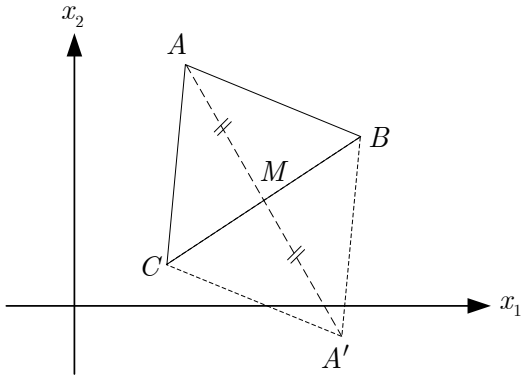
بهینه‌سازی محلی

۲-۱ روش‌های بی‌نیاز از مشتق: جستجوی مستقیم

در این روش‌ها، تابع f در تعدادی نقطه که به صورت هوشمند انتخاب می‌گردد. ارزیابی می‌شود.

۲-۱-۱ روش سیمپلکس

برای توضیح این روش، تابع دو متغیره‌ی $y = f(x_1, x_2)$ را در نظر می‌گیریم که می‌خواهیم آن را می‌نیمم کنیم.



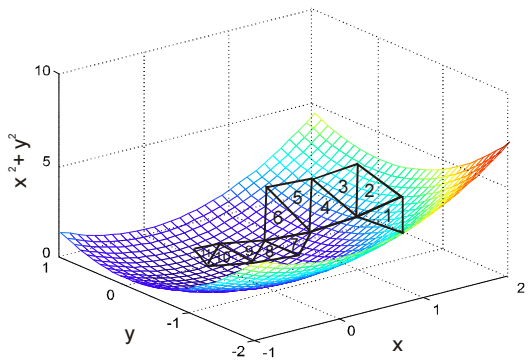
$$f(A) > f(B), f(C)$$

سه نقطه‌ی تصادفی را در فضای جستجو انتخاب می‌کنیم: A، B و C (مثلث حاصل از این سه نقطه سیمپلکس نام دارد). نقطه‌ای که بزرگترین مقدار تابع را دارد، A در نظر می‌گیریم و قرینه‌ی این نقطه را نسبت به خط گذرنده از دو نقطه‌ی دیگر B و C به دست می‌آوریم (A'). حال این عملیات را با سیمپلکس جدید A'، B و C ادامه می‌دهیم. با این عملیات، سیمپلکس رفته رفته به نقطه‌ی می‌نیمم نزدیک می‌شود. وقتی این فرایند به نوسان افتاد، می‌توانیم مثلاً طول اضلاع را نصف کنیم.

در حالت n متغیره، یک چندوجهی با $n+1$ راس متحدالفاصله در نظر گرفته می‌شود.

مزیت این روش در آن است که در هرگام تنها یک مرتبه $f(.)$ در یک نقطه‌ی جدید محاسبه می‌شود.

این روش به تعداد تکرار زیادی نیاز دارد و بنابراین سرعت بالایی ندارد، بعلاوه هوشمندی خاصی در انتخاب جهت جستجو ندارد.



تذکر. این روش ارتباطی با روش سیمپلکس در برنامه‌ریزی خطی ندارد.

۲-۱-۲ روش هوک و هِوِز

در این روش، در فضای n بعدی، n بردار مستقل عمود بر هم (پایه) انتخاب می‌کنیم:

$$\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$$

حال جستجو را در هر مرحله در راستای یکی از بردارهای فوق جلو می‌بریم:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i \boldsymbol{v}_{i \bmod n}$$

می‌توان این الگوریتم را قدری بهبود داد، به این ترتیب که بعد از یافتن $\boldsymbol{x}_{i+1}$، یک بار هم در جهت $\boldsymbol{x}_{i+1} - \boldsymbol{x}_i$ که تخمینی از گرادیان است، جستجو کنیم و بعد به تکرار جستجو در جهات ثابت nگانه‌ی $\boldsymbol{v}_1$ تا $\boldsymbol{v}_n$ بپردازیم.

۲-۲ روش‌های مبتنی بر گرادیان

روش‌های مبتنی برگرادیان از اطلاعات مشتق مرتبه اول $f(.)$ استفاده می‌کنند. این روش‌ها در دسته‌های عمده‌ی زیر طبقه‌بندی می‌شوند.

۲-۲-۱ روش تندترین شیب

ابتدا به بسط تیلور تابع f حول $\mathbf{x}_0$ توجه می‌کنیم:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + (\nabla f)^T(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + O(||\boldsymbol{x} - \boldsymbol{x}_0||^2)$$

برای $\mathbf{x}$های متعلق به همسایگی $\mathbf{x}_0$ با تقریب می‌توان نوشت:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + (\nabla f)^T(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \quad , \boldsymbol{x} \in N(\boldsymbol{x}_0)$$

با توجه به این بسط می‌توان گفت کوچکترین مقدار f در این همسایگی، وقتی رخ می‌دهد که $(\boldsymbol{x} - \boldsymbol{x}_0)^T(\nabla f)(\boldsymbol{x}_0)$ حتی‌الامکان کوچک باشد و این زمانی است که بردارهای $\nabla f(\boldsymbol{x}_0)$ و $\boldsymbol{x} - \boldsymbol{x}_0$ هم‌راستا و در خلاف جهت یکدیگر باشند. به عبارت دیگر می‌توان نوشت:

$$\boldsymbol{x} - \boldsymbol{x}_0 = -\eta \nabla f(\boldsymbol{x}_0) \quad , \eta > 0$$

و از آنجا داریم:

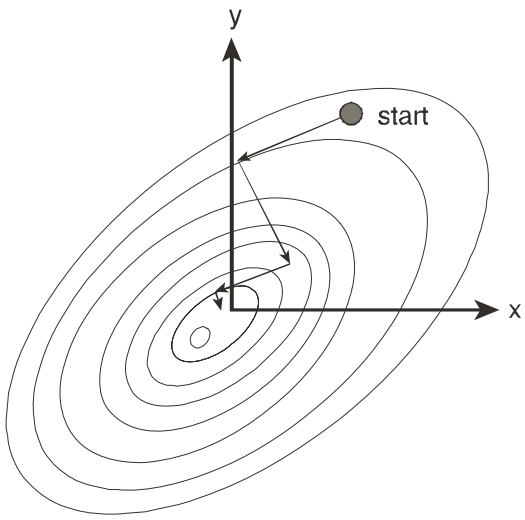
$$\boldsymbol{x} = \boldsymbol{x}_\circ - \eta \nabla f(\boldsymbol{x}_\circ)$$

به پارامتر η اصطلاحاً اندازه‌ی گام (step size) می‌گوییم.

برای یافتن ηی مناسب، از جستجوی خط (line search) استفاده می‌کنیم. η را چنان می‌یابیم که $f(\boldsymbol{x}_\circ - \eta \nabla f(\boldsymbol{x}_\circ))$ می‌نیمم شود:

$$\begin{aligned}
\frac{\partial f(\boldsymbol{x}_\circ - \eta \nabla f(\boldsymbol{x}_\circ))}{\partial \eta} = \circ \quad &\Rightarrow \quad -\{\nabla f(\underbrace{\boldsymbol{x}_\circ - \eta \nabla f(\boldsymbol{x}_\circ)}_{\boldsymbol{x}})\}^T \nabla f(\boldsymbol{x}_\circ) = \circ \\
&\Rightarrow \quad \nabla f(\boldsymbol{x})^T \nabla f(\boldsymbol{x}_\circ) = \circ \\
&\Rightarrow \quad \nabla f(\boldsymbol{x}) \perp \nabla f(\boldsymbol{x}_\circ)
\end{aligned}$$

به عبارت دیگر جهت‌های حرکت متوالی بر هم عمود هستند:



داریم

$$\boldsymbol{x} = \boldsymbol{x}_\circ - \eta \nabla f(\boldsymbol{x}_\circ)$$

و

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_\circ) + \nabla f(\boldsymbol{x}_\circ)^T (\boldsymbol{x} - \boldsymbol{x}_\circ) + \frac{۱}{۲} (\boldsymbol{x} - \boldsymbol{x}_\circ)^T H_f(\boldsymbol{x}_\circ)(\boldsymbol{x} - \boldsymbol{x}_\circ)$$

در نتیجه

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_\circ) + (-\eta\|\nabla f(\boldsymbol{x}_\circ)\|^{۲}) + \frac{۱}{۲}\eta^{۲}\nabla f(\boldsymbol{x}_\circ)^T H_f(\boldsymbol{x}_\circ)\nabla f(\boldsymbol{x}_\circ)$$

و از آنجا

$$\frac{\partial f(\boldsymbol{x}_\circ)}{\partial \eta} = \circ \Rightarrow \eta^* = \frac{\nabla f(\boldsymbol{x}_\circ)^T \nabla f(\boldsymbol{x}_\circ)}{\nabla f(\boldsymbol{x}_\circ)^T H_f(\boldsymbol{x}_\circ)\nabla f(\boldsymbol{x}_\circ)}$$

مزیت این روش در پیاده‌سازی ساده و استفاده از مشتق مرتبه‌ی اول و نیز مقیاس‌پذیری الگوریتم است. بعلاوه، نتیجه در هر تکرار بدتر از تکرار قبلی نمی‌شود.

عیب این روش در همگرایی کند نزدیک نقطه‌ی بهینه است. بخصوص اگر حول می‌نیمم تقریب درجه دوم بزنیم، همگرایی در یک گام حاصل نمی‌شود.

بنابراین این روش برای بهینه‌سازی زمانی مناسب است که در آغاز بهنیه‌سازی قرار داریم و از نقطه‌ی می‌نیمم دور هستیم.

۲-۲-۲ روش نیوتون

در یک تابع درجه‌ی دوم داریم:

$$f(\boldsymbol{x}) = \frac{۱}{۲}\boldsymbol{x}^T A\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x} + c \Rightarrow \boldsymbol{x}^* = -A^T\boldsymbol{x}$$

اندازه‌ی گام در روش نیوتون به گونه‌ای انتخاب شده است که در یک گام نقطه‌ی بهنیه‌ی تابع درجه‌ی دوم را به دست دهد.

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_\circ) + \nabla f(\boldsymbol{x}_\circ)^T(\boldsymbol{x}-\boldsymbol{x}_\circ) + \frac{۱}{۲}(\boldsymbol{x}-\boldsymbol{x}_\circ)^T H_f(\boldsymbol{x}_\circ)(\boldsymbol{x}-\boldsymbol{x}_\circ) + O(\|\boldsymbol{x}-\boldsymbol{x}_\circ\|^{۳})$$

با مشتق‌گیری داریم:

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \circ \Rightarrow \nabla f(\boldsymbol{x}_\circ)^T + H_f(\boldsymbol{x}_\circ)(\boldsymbol{x}-\boldsymbol{x}_\circ) = \circ$$

و از آنجا

$$\boldsymbol{x} = \boldsymbol{x}_\circ - H_f^{-۱}(\boldsymbol{x}_\circ)\nabla f(\boldsymbol{x}_\circ)$$

مزیت این روش در همگرایی بسیار سریع‌تر از روش تندترین شیب در نزدیکی نقطه‌ی کمینه است.

اما معایبی نیز برای این روش وجود دارد، از جمله:

- نیاز به محاسبه‌ی مشتقات مرتبه دوم دارد.
- ممکن است $H_f(\boldsymbol{x})$ وارون‌پذیر نباشد.
- تضمینی وجود ندارد که مقدار تابع در هر گام حتماً در حال کوچک‌تر شدن باشد. در واقع این موضوع بستگی به $H_f(\boldsymbol{x})$ در نقطه‌ی $\boldsymbol{x} = \boldsymbol{x}_i$ دارد. بخصوص زمانی که از می‌نیمم دور هستیم، این وضعیت احتمال وقوع بیشتری دارد.

نتیجه این است که به شرط معین مثبت بودن ماتریس هسین $H_f(\boldsymbol{x})$، برای یافتن می‌نیمم، همگرایی تضمین می‌شود. پس حول نقطه‌ی می‌نیمم که تقریب درجه دوم با تقریب بسیار خوبی معتبر است، این روش کارساز خواهد بود.

۲-۲-۳ روش شبه نیوتون

ایده‌ی اساسی روش شبه نیوتون، در تقریب زدن وارون ماتریس هسین است که محاسبه‌ی مستقیم آن به دلیل وجود مشتقات مرتبه دوم و نیز امکان وارون‌ناپذیری دشوار است.

در این روش داریم

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i S_i \boldsymbol{g}_i$$

که در آن S_i ماتریس $n \times n$ تخمین وارون هسین $([\nabla^2 f(\boldsymbol{x}_i)]^{-1})$ و $\boldsymbol{g}_i = \nabla f(\boldsymbol{x}_i)$ است.

S_i در هر تکرار توسط رابطه‌ی زیر بهنگام می‌شود:

$$S_{i+1} = S_i + Q_i, \qquad S_0 = I$$

در واقع به ازای $S_0 = I$ از جهت عکس گرادیان و به عبارت دیگر تندترین شیب شروع می‌کنیم. برای محاسبه‌ی Q روش‌های مختلفی وجود دارد که از آن جمله می‌توان به بهترین آنها یعنی روش BFGS اشاره نمود.

۲-۲-۴ روش BFGS

رابطه‌ی تکرار در این روش به صورت زیر است:

$$S_{i+1} = \left(I - \frac{\Delta \boldsymbol{x}_i \Delta \boldsymbol{g}_i^T}{\Delta \boldsymbol{x}_i^T \Delta \boldsymbol{g}_i}\right) S_i \left(I - \frac{\Delta \boldsymbol{x}_i \Delta \boldsymbol{g}_i^T}{\Delta \boldsymbol{x}_i^T \Delta \boldsymbol{g}_i}\right)^T + \frac{\Delta \boldsymbol{x}_i \Delta \boldsymbol{g}_i^T}{\Delta \boldsymbol{x}_i^T \Delta \boldsymbol{g}_i}$$

که در آن

$$\Delta \boldsymbol{x}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i, \qquad \Delta \boldsymbol{g}_i = \boldsymbol{g}_{i+1} - \boldsymbol{g}_i$$

می‌توان نشان داد که:

اگر S_i معین مثبت باشد، S_{i+1} هم معین مثبت است.

اگر f درجه‌ی دوم و $\boldsymbol{x}$ برداری nبعدی باشد، $S_n = H_f^{-1}(\boldsymbol{x}^*)$. یعنی پس از طی حداکثر n گام به نقطه‌ی بهینه می‌رسیم.

برای جستجوی خط، واقعاً لازم نیست بهینه را بیابیم، یک روش جستجو که به جواب تقریباً خوبی منجر شود، هم به اندازه‌ی کافی مناسب است و در مجموع منجر به الگوریتم مؤثرتری می‌شود.

در مجموع این روش بسیار خوب است و برای مسایل با اندازه‌ی متوسط (مثلاً تا ۱۰۰ متغیر) با موفقیت استفاده می‌شود.

۲-۲-۵ روش گرادیان مزدوج

با وجود اینکه روش شبه نیوتون بسیار خوب و مؤثر است، اما پیچیدگی محاسباتی آن $O(n^2)$ می‌باشد. روش گرادیان مزدوج با این هدف بنا شده است که این پیچیدگی را به $O(n)$ کاهش دهد. به این منظور به جای تخمین زدن وارون هسین، در یک گام جهت حرکت یعنی حاصلضرب وارون هسین در گرادیان را تخمین می‌زند.

رابطه‌ی تکرار در این روش به صورت

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i \boldsymbol{p}_i$$

است که در آن

$$\boldsymbol{p}_i = \boldsymbol{g}_i - \beta_i \boldsymbol{p}_{i-1}, \qquad \boldsymbol{p}_0 = \nabla f(\boldsymbol{x}_0)$$

و

$$\boldsymbol{g}_i = \nabla f(\boldsymbol{x}_i), \qquad \beta_i = \frac{\boldsymbol{g}_i^T \boldsymbol{g}_i}{\boldsymbol{g}_{i-1}^T \boldsymbol{g}_{i-1}} = \frac{\|\boldsymbol{g}_i\|}{\|\boldsymbol{g}_{i-1}\|}$$

در این روش داریم

$$\boldsymbol{g}_i^T H_{i-1} \boldsymbol{g}_{i-1} = 0$$

می‌توان دید که این روش در حقیقت تقریبی از روش شبه نیوتون است، وقتی که S_i در هر بهنگام‌سازی مجدداً به ماتریس همانی مقداردهی شود. بنابراین انتظار داریم که روش گرادیان مزدوج تعداد تکرار بیشتری نسبت به روش شبه نیوتون نیاز داشته باشد تا همگرا شود. اما هر تکرار سریع‌تر است. معمولاً با بزرگ‌تر شدن مساله به صورت تدریجی، روش گرادیان مزدوج سریع‌تر از روش شبه نیوتون همگرا می‌شود.

چون در این روش جهت حرکت حداکثر بهینگی ممکن را ندارد، عمل جستجوی خط نسبت به روش شبه نیوتون ضروری‌تر است.

ثابت می‌شود که این روش هم برای تابع درجه دوم حداکثر در n گام به بهینه‌ی مطلق می‌رسد. در مواردی که با تعداد زیادی متغیر (در حدود ۱۰۰۰ یا بیشتر) سر و کار داریم، یعنی مسایل بهینه‌سازی بزرگ، این روش مناسب است.

۶-۲-۲ روش حداقل مربعات غیرخطی: روش گاوس ـ نیوتون

فرض می‌کنیم تابع هدف دارای ساختار درجه دوم (quadratic) باشد:

$$F(\boldsymbol{\theta}) = \sum_{i=1}^{N} f^2(i, \boldsymbol{\theta})$$

که در آن $\boldsymbol{\theta}$ بردار پارامترها به صورت

$$\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \ldots \ \theta_n]^T$$

باشد.

اگر f تابعی خطی از $\boldsymbol{\theta}$ باشد، با روش حداقل مربعات خطی $f(i, \boldsymbol{\theta}) = \sum_{j=1} Ja_{ji}\theta_j$ سر و کار خواهیم داشت و اگر f تابعی غیرخطی از $\boldsymbol{\theta}$ باشد، با روش حداقل مربعات غیرخطی سر و کار خواهیم داشت.

برای محاسبه‌ی گرادیان و هسین، بردار $\boldsymbol{f}$ را به صورت

$$\boldsymbol{f} = [f(1, \boldsymbol{\theta}) \ f(2, \boldsymbol{\theta}) \ \ldots \ f(N, \boldsymbol{\theta})]^T$$

تعریف می‌کنیم. با این تعریف داریم:

$$F(\boldsymbol{\theta}) = \sum_{i=1}^{N} f^2(t, \boldsymbol{\theta}) = \boldsymbol{f}^T \boldsymbol{f}$$

گرادیان را به صورت زیر محاسبه می‌کنیم:

$$\nabla F(\boldsymbol{\theta}) = \boldsymbol{g} = \frac{\partial F}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial F}{\partial \theta_۱} & \frac{\partial F}{\partial \theta_۲} & \cdots & \frac{\partial F}{\partial \theta_n} \end{bmatrix}$$

$$\boldsymbol{g}_i = \frac{\partial I}{\partial \theta_j} = ۲ \sum_{i=۱}^{N} f(i, \boldsymbol{\theta}) \frac{\partial f(i, \boldsymbol{\theta})}{\partial \theta_j}, \qquad i = ۱, ۲, \ldots, n$$

با تعریف

$$J = \left[\frac{\partial f(i, \boldsymbol{\theta})}{\partial \theta_j} \right]_{N \times n}$$

خواهیم داشت

$$\boldsymbol{g} = ۲ J^T \boldsymbol{f}$$

هسین را به صورت زیر محاسبه می‌کنیم:

$$H = \nabla^۲ F(\boldsymbol{\theta}) = \left[\frac{\partial^۲ F(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right]_{n \times n}$$

داریم:

$$\begin{aligned} \frac{\partial^۲ F(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} &= \frac{\partial}{\partial \theta_i} \left(\frac{\partial F(\boldsymbol{\theta})}{\partial \theta_j} \right) \\ &= \frac{\partial}{\partial \theta_i} \left(۲ \sum_{i=۱}^{N} f(i, \boldsymbol{\theta}) \frac{\partial f(i, \boldsymbol{\theta})}{\partial \theta_j} \right) \\ &= ۲ \sum_{l=۱}^{N} \left(\frac{\partial f(l, \boldsymbol{\theta})}{\partial \theta_i} \frac{\partial f(l, \boldsymbol{\theta})}{\partial \theta_j} + f(l, \boldsymbol{\theta}) \frac{\partial^۲ f(l, \boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right) \end{aligned}$$

بنابراین

$$H = [H_{ij}]_{n \times n} = ۲ J^T J + ۲ \sum_{l=۱}^{N} f(l, \boldsymbol{\theta}) \nabla^۲_{\boldsymbol{\theta}} f(l, \boldsymbol{\theta})$$

از آنجا که هدف می‌نیمم کردن $\sum f^۲$ است، می‌توان نوشت

$$H \approx ۲ J^T J$$

با توجه به آن که $f_i()$ها عموماً معرف خطا هستند، فرض کوچک بودن آن‌ها فرض چندان دور از واقعیتی نیست. بر این اساس دو روش بنا می‌شود:

در روش گاوس ـ نیوتون از رابطه‌ی تکرار زیر استفاده می‌شود:

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \alpha_i (J_{i-1}^T J_{i-1})^{-1} J_{i-1}^T \boldsymbol{f}_{i-1}$$

این روش در ترکیب با جستجوی خط برای α_i بسیار مؤثر است و نیازی به محاسبه‌ی مشتقات دوم هم ندارد.

۷-۲-۲ روش حداقل مربعات غیرخطی: روش لِونبرگ ـ مارکوارد

مشکل روش گاوس ـ نیوتون در این است که ممکن است ماتریس $J_{i-1}^T J_{i-1}$ وارون‌ناپذیر باشد. برای حل این مشکل، از تکنیک تنظیم (regularization) در بهنیه‌سازی استفاده می‌کنیم: افزودن ضریبی از ماتریس همانی. این کار ما را به روش لِونبرگ ـ مارکوارد (Levenberg-Marquard) می‌رساند.

رابطه‌ی تکرار در این روش به صورت زیر است

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \alpha_i (J_{i-1}^T J_{i-1} + \gamma_i I_n)^{-1} J_{i-1}^T \boldsymbol{f}_{i-1}$$

$\gamma_i I_n$ مشخص‌کننده‌ی یک تغییر جهت به اندازه‌ای کوچک است. برای تعیین γ_i به این نکته توجه می‌کنیم که هرچه γ_i کوچک‌تر باشد، بهتر است. اگر ماتریس وارون‌پذیر باشد، $\gamma = 0$ را در نظر می‌گیریم.

اگر $A_{n\times n} = J_i^T J_i$ را در نظر بگیریم، داریم:

$$AV = VD \Rightarrow A = VDV^{-1}$$

که در آن D ماتریس قطری مقادیر ویژه و V ماتریس بردارهای ویژه‌ی متناظر A است و از آنجا

$$A + \gamma I = VDV^{-1} + \gamma I = VDV^{-1} + \gamma VV^{-1} = V(D + \gamma I)V^{-1}$$

اگر مقادیر ویژه‌ی A، λ_1، λ_2، ...، λ_n باشد، آن‌گاه مقادیر ویژه‌ی $A + \gamma I$، $\lambda_1 + \gamma$، $\lambda_2 + \gamma$، ...، $\lambda_n + \gamma$ خواهد بود.

عدد وضعیت ماتریس A به صورت زیر تعریف می‌شود:

$$\text{condition number} = \left|\frac{\lambda_{\max} + \gamma}{\lambda_{\min} + \gamma}\right| \approx \left|\frac{\lambda_{\max}}{\gamma}\right|$$

با شرط $|\lambda_{\min}| \le \gamma \le |\lambda_{\max}|$، γ آن قدر بزرگ انتخاب می‌شود که ماتریس از بدوضعی درآید.

فصل ۳

بهینه‌سازی سراسری

۳-۱ تکرار جستجوی محلی

یک راه رسیدن به بهینه‌ی مطلق (سراسری)، تکرار یک الگوریتم بهینه‌سازی محلی است از شرایط اولیه‌ی متفاوت است. این روش در یادگیری شبکه‌های عصبی کاربرد دارد.

۳-۲ جستجو با جهت تصادفی

روش جستجو با جهت تصادفی، برای می‌نیمم کردن $E(\boldsymbol{x})$ از رابطه‌ی تکرار زیر استفاده می‌کند:

$$\boldsymbol{x}^{+} = \boldsymbol{x}^{-} - \eta \boldsymbol{p}^{-}$$

که در آن $\boldsymbol{p}^-$ یک جهت تصادفی است و η می‌تواند به صورت تصادفی یا تطبیقی تعیین شود. در هر مرحله‌ای از تکرار که $E(\boldsymbol{x}^+) > E(\boldsymbol{x}^-)$ باشد، حرکت پذیرفته می‌شود.

۳-۳ تافت شبیه‌سازی‌شده

در روش تافت شبیه‌سازی شده (simulated annealing) برای می‌نیمم کردن $E(\boldsymbol{x})$ روال زیر به کار گرفته می‌شود:

۱) انتخاب $\boldsymbol{x}$ اولیه به تصادف

۲) انتخاب $\boldsymbol{p}$ جهت حرکت اولیه به تصادف

۳) $\boldsymbol{x}^+ = \boldsymbol{x}^- - \eta \boldsymbol{p}^-$

۴) اگر $E(\boldsymbol{x}^+) \leq E(\boldsymbol{x}^-)$ باشد، حرکت پذیرفته می‌شود،
اگر $E(\boldsymbol{x}^+) > E(\boldsymbol{x}^-)$ باشد، حرکت با یک احتمال پذیرفته می‌شود که این احتمال تابعی از $\Delta E = E(\boldsymbol{x}^+) - E(\boldsymbol{x}^-)$ است.

۵) در صورت برقراری شرایط همگرایی توقف و در غیر این صورت پرش به گام (۲)

احتمال پذیرش حرکت نامناسب، با توزیع بولتزمن داده می‌شود:

$$P(\Delta E) = \frac{1}{1 + e^{-\Delta E/T(t)}}, \qquad \Delta E > 0$$

این احتمال عددی بین $\frac{1}{2}$ و ۱ است. برای پیاده‌سازی یک ϵ به تصادف با توزیع یکنواخت بین $[0, 1]$ انتخاب می‌شود و در صورتی که $P(\Delta E) < \epsilon$ باشد، حرکت پذیرفته می‌شود.

$T(t)$ بیانگر تابع دماست که به مرور زمان کاهش می‌یابد.

$$T(t) = \frac{T_0}{1 + \ln t}, \qquad t = 1, 2, 3, \ldots$$

اگر نرخ $T(t)$ کوچکتر یا مساوی با $\frac{T_0}{1+\ln t}$ باشد، تضمین همگرایی وجود دارد.

این الگوریتم می‌تواند با چند شرط اولیه آغاز شود.

انتخاب T_0 به آزمون و خطا نیاز دارد.

اگر $P(\Delta E) = \frac{T(t)}{T(t)+\Delta E}$ باشد، آن‌گاه تابع دما می‌تواند به صورت $T(t) = \frac{T_0}{1+t}$ انتخاب شود.

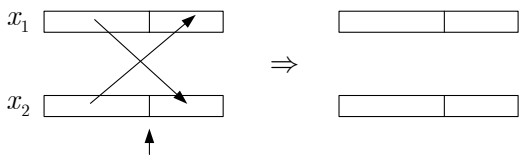
۳-۴ الگوریتم ژنتیک

ابتدا چند اصطلاح را مرور می‌کنیم:

کروموزوم یک بازنمایی دودویی از ویژگی‌ها

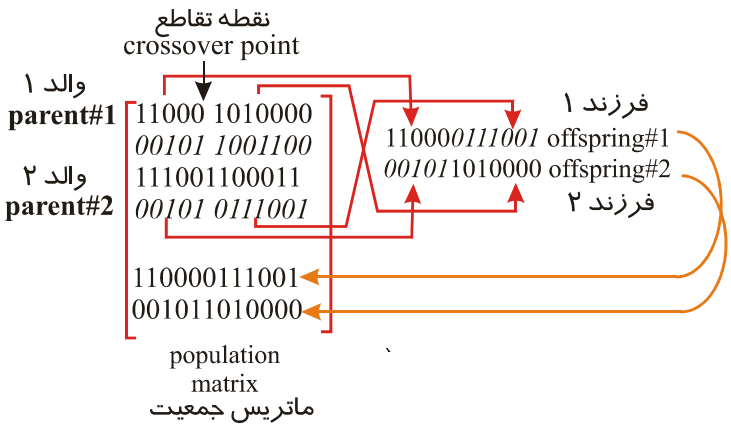
جمعیت گردایه‌ای از کروموزوم‌ها

افراد جمعیت را به تصادف جفت می‌کنیم (طول کروموزوم‌ها یکسان است)، یک نقطه‌ی تصادفی در طول کروموزوم انتخاب می‌نماییم و از آن نقطه بخش‌های سمت راست و چپ دو کروموزم جفت را با هم تعویض می‌کنیم. این عمل **تقاطع** (crossover) نام دارد و با احتمال P_C انجام می‌شود.



یک بردار به اندازه‌ی کروموزوم در نظر می‌گیریم که همه‌ی عناصر آن صفر باشد. هر عنصر بردار با احتمال P_M از ۰ به ۱ تغییر پیدا خواهد کرد. در کروموزوم iام، بیت‌های نظیر یک‌ها را مکمل می‌کنیم و این کار را برای همه‌ی کروموزوم‌ها تکرار می‌نماییم. به این عملیات **جهش** (mutation) می‌گوییم.

تابع برازش (fitness)، E، میزان خوبی یک کروموزوم را نشان می‌دهد. برای همه‌ی کروموزوم‌ها E را محاسبه می‌کنیم.



هر تکرار الگوریتم ژنتیک که با یک جمعیت انجام می‌شود، یک **نسل** (generation) نام دارد. عمل انتخاب کروموزوم‌های جمعیت در یک نسل **انتخاب** (selection) نامیده می‌شود. برای این کار عددی تصادفی بین 0 و مجموع همه‌ی مقایر برازش کروموزوم‌های جمعیت فعلی تولید می‌شود و مطابق با آن یک $\boldsymbol{x}$ برنده شده و بر این اساس n کروموزوم انتخاب می‌شود.

برای استفاده از الگوریتم‌های ژنتیک، بایستی پارامترهای زیر تعیین شوند:

- انتخاب ساختار کروموزوم
- انتخاب تابع برازش
- انتخاب احتمال جهش P_M و احتمال تقاطع P_C

مراجع

[1] Fletcher, R., *Practical Methods of Optimization*, Vol. I: Unconstrained Optimization, Wiley, New York, 1980.

[2] Haupt, L. H and Haupt, E. H. *Practical Genetic Algorithms*, Wiley, 2004.