پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

مروری بر مسایل بهینه‌سازی برای یادگیری ماشین

نگارنده

شکیبا رهنما

استاد راهنما: دکتر مجید سلیمانی دامنه

پایان‌نامه برای دریافت درجه کارشناسی
در رشته ریاضیات و کاربردها

مرداد ۱۴۰۰

چکیده

یادگیری ماشین به طور گسترده‌ای در زمینه‌های مختلف مورد استفاده قرار می‌گیرد و به سرعت در حال توسعه است. روش‌های بهینه‌سازی در الگوریتم‌های یادگیری ماشین نقشی اساسی دارند. هنگام کار با داده‌های بسیار زیاد، پیچیدگی مدل افزایش پیدا می‌کند و به دنبال آن روش‌های بهینه‌سازی با چالش‌های بیشتری روبه‌رو می‌شوند. کارهای زیادی برای بهبود روش‌های بهینه‌سازی در یادگیری ماشین انجام شده است. در این پژوهش، ابتدا مسائل بهینه‌سازی در یادگیری ماشین را بیان می‌کنیم. سپس، برخی از روش‌های بهینه‌سازی را معرفی می‌کنیم ودر ادامه کاربردهای این روش‌ها را در زمینه‌های مختلف یادگیری ماشین توضیح می‌دهیم. در آخر، به برخی از چالش‌ها و مسائل باز بهینه‌سازی در یاگیری ماشین می‌پردازیم.

کلمات کلیدی: روش‌های بهینه‌سازی، یادگیری ماشین، شبکه‌های عصبی، یادگیری تقویتی

پیشگفتار

در چند سال اخیر، یادگیری ماشین توسعه و پیشرفت قابل توجهی داشته، به یکی از موضوعات محبوب برای تحقیقات تبدیل شده و در بسیاری از زمینه‌ها مانند ترجمه ماشینی، تشخیص گفتار، تشخیص تصویر و ... نقش به‌سزایی داشته است. از همان ابتدا، یادگیری ماشین از فرمول‌بندی‌ها و الگوریتم‌های بهینه‌سازی استفاده کرده است. توسعه روش‌های بهینه‌سازی به رفع چالش‌های قابل توجه بوجود آمده در یادگیری ماشین کمک فراوانی کرده و به دلیل کاربرد گسترده و ویژگی‌های نظری، این روش‌ها در یادگیری ماشین از اهمیت ویژه‌ای برخوردارند. روش‌های بهینه‌سازی موثری برای توسعه یادگیری ماشین ارائه شده است که باعث بهبود عملکرد و کارایی روش‌های یادگیری ماشین شده است. روش‌های بهینه‌سازی را از نظر استفاده از مشتق می‌توان به سه دسته تقسیم کرد:

۱. روش‌های بهینه‌سازی مرتبه اول

۲. روش‌های بهینه‌سازی مرتبه بالاتر

۳. روش‌های بهینه‌سازی بدون مشتق

توسعه سریع روش‌های بهینه‌سازی مرتبه اول، به دلیل موفقیت آن‌ها در کارهای مختلف از جمله یادگیری ماشین، پردازش سیگنال، تصویربرداری و نظریه کنترل است. انواع روش گرادیان کاهشی تصادفی در سالهای اخیر به طور گسترده‌ای مورد استفاده قرار گرفته و با سرعت بالایی در حال تکامل است. عمده روش‌های مرتبه اول توانایی ارائه جواب‌هایی با دقت نه چندان بالا با پیچیدگی محاسباتی کم دارند که آن‌ها را به مجموعه‌ای جذاب برای حل مسائل بهینه‌سازی در مقیاس بزرگ تبدیل می‌کند [۹].

در مقایسه با روش‌های بهینه‌سازی مرتبه اول، روش‌های مرتبه بالاتر با سرعت بیشتری همگرا می‌شوند و در آن‌ها اطلاعات انحنا جهت جستجو را موثرتر می‌کند، اما با چالش‌های زیادی روبه‌رو هستند. روش‌های شبه نیوتن یا نیوتن با استفاده از اطلاعات مرتبه دوم می‌توانند به دقت بالاتر و همگرایی سریعتر دست پیدا کنند، ولی دشواری روش‌های

مرتبه بالاتر در همین عملکرد و ذخیره کردن معکوس ماتریس هسین است. برای حل این مشکل، روش‌هایی بر پایه روش نیوتن توسیع داده شده‌اند که تلاش می‌کنند تقریبی برای ماتریس هسی بدست آورند [۱۵، ۱۶].

روش‌های بهینه‌سازی بدون مشتق معمولا در حالتی استفاده می‌شوند که مشتق تابع هدف وجود ندارد یا محاسبه آن دشوار است. در بیشتر مسایل علمی یادگیری ماشین، اغلب به مجموعه داده‌های بزرگ و مراحل پیچیده برای ارزیابی عملکرد و شیب هدف برخورد می‌کنیم و بنابراین، به حداقل اطلاعات در مورد تابع هدف نیاز است، یعنی فقط در هر نقطه به مقدار تابع نیاز داریم. دو ایده اصلی در این روش وجود دارد: یکی، جستجوی ابتکاری بر اساس قوانین آزمایشی و دیگری برازش تابع هدف با نمونه‌ها است.

بسیاری از مسائل یادگیری ماشین را می‌توان پس از مدل سازی به عنوان مسائل بهینه‌سازی حل کرد. بهینه‌سازی در زمینه‌های شبکه عصبی عمیق، یادگیری تقویتی، یادگیری متا، استنباط تغییرات و زنجیره مارکوف مونت کارلو با مشکلات و چالش‌های مختلفی روبه‌رو است.

شبکه‌های عصبی عمیق دسته قدرتمندی از الگوریتم‌های یادگیری ماشین هستند که توانایی های یادگیری ممتاز و متمایزی در سال‌های اخیر نشان داده‌اند. تکنیک‌های یادگیری عمیق در کارهای مختلف بینایی رایانه‌ای مانند طبقه بندی تصویر، تشخیص اشیا و درک صحنه به نتایج پیشرفته رسیده‌اند. شبکه عصبی عمیق در شناخت الگو موفق عمل کرده است و از روش‌های آن می‌توان به شبکه عصبی کانوولوشن و شبکه عصبی بازگشتی که نقش مهمی در زمینه‌های مختلف یادگیری ماشین ایفا می‌کنند نام برد. شبکه‌های عصبی کانوولوشن با موفقیت در زمینه‌های زیادی مانند پردازش تصویر، پردازش ویدئو و پردازش زبان طبیعی استفاده شده است. شبکه عصبی بازگشتی یکی از انواع مدل‌های دنباله‌ای است که در پردازش زبان طبیعی پرکاربرد و در زمینه‌های پردازش تصویر و پردازش ویدئو بسیار محبوب است.

الگوریتم‌های گرادیان تصادفی به طور گسترده‌ای در شبکه‌های عصبی عمیق استفاده می‌شوند. هنگام استفاده از الگوریتم‌های مبتنی بر گرادیان تصادفی ممکن است با مشکلاتی مانند نوسان نرخ یادگیری در مرحله بعدی آموزش روبه‌رو شویم که به مشکلات واگرایی منجر شود. بنابراین، تلاش الگوریتم‌های بهینه‌سازی بهبود نرخ همگرایی بر اساس کاهش واریانس است. علاوه بر این، ترکیب گرادیان کاهشی تصادفی و ویژگی‌های انواع این روش در بهبود بهینه‌سازی موثر است و تغییر یک الگوریتم تطبیقی به گرادیان کاهشی می‌تواند همگرایی را بهبود بخشد [۱۳].

یادگیری تقویتی [۲۱]، شاخه‌ای از یادگیری ماشین است که ریشه آن در روانشناسی یادگیری حیوانات است. ایده اصلی از پاداش یادگیرنده (عامل) برای اقدامات درست و مجازات برای اقدامات اشتباه است. به طور مستقیم،

یادگیری تقویتی یک فرایند آزمون و خطا، همراه با یادگیری است. عامل براساس اقدامات موجود در محیط فعلی تصمیم گیری می‌کند و از طریق بازخورد مطلوب بودن عمل، می‌آموزد کدام عمل با کدام حالت بهتر است. به طور کلی عامل با مکانیسم آزمون و خطا با محیط ارتباط برقرار می‌کند و با به حداکثر رساندن پاداش کل، یک سیاست بهینه را یاد می‌گیرد. یادگیری تقویتی عمیق ترکیبی از تکنیک‌های یادگیری تقویتی و یادگیری عمیق است که عامل یادگیری تقویتی را قادر می‌سازد تا درک خوبی از محیط خود داشته باشد. به طور معمول، در یادگیری تقویتی و یادگیری تقویتی عمیق از الگوریتم‌های بهینه‌سازی تصادفی استفاده می‌شود.

یادگیری متا [۱۱] اخیرا به شاخه‌ای محبوب در یادگیری ماشین تبدیل شده است. هدف یادگیری متا، طراحی مدلی است که بتواند با کمترین تعداد نمونه خود را با محیط جدید سازگار کند. متا یادگیری معمولاً به الگوریتم‌هایی از یادگیری ماشین اشاره دارد که از خروجی الگوریتم‌های دیگر یادگیری ماشین، می‌آموزند. این به معنای استفاده از الگوریتم‌هایی است که می‌آموزند چگونه بهترین ترکیب را از الگوریتم‌های یادگیری ماشین انجام دهند.

بهینه‌سازی سهم زیادی در توسعه پیشرفت یادگیری ماشین داشته است؛ با این وجود، هنوز چالش‌ها و مشکلات بسیاری برای مسائل بهینه‌سازی در یادگیری ماشین وجود دارد. به عنوان مثال، اگر داده‌های کافی در آموزش شبکه عصبی عمیق وجود نداشته باشد منجر به ایجاد واریانس زیاد و بیش پردازش می‌شود. علاوه بر این، بهینه‌سازی غیرمحدب یکی از مشکلات شبکه عصبی عمیق است که باعث می‌شود بهینه‌سازی به جای رسیدن به مقدار بهینه سراسری به یک مقدار بهینه موضعی برسد و مشکلات دیگری که در ادامه برخی از آن‌ها را بیان می‌کنیم.

هدف این پژوهش جمع بندی، تجزیه و تحلیل روش‌های بهینه‌سازی کلاسیک و مدرن از دیدگاه کاربرد در یادگیری ماشین است. برخی مسائل یادگیری ماشین را از دید بهینه‌سازی مرور می‌کنیم. الگوریتم‌های کلاسیک بهینه‌سازی و آخرین پیشرفت‌های آنها در یادگیری ماشین را شرح می‌دهیم. کاربرد روش‌های بهینه‌سازی در برخی از زمینه‌های یادگیری ماشین را بررسی می‌کنیم. در پایان، چالش‌ها و مسائل باز در روش‌های بهینه‌سازی را مرور می‌کنیم.

فهرست مطالب

فصل ۱

مدل کردن یادگیری ماشین به عنوان یک مساله بهینه‌سازی

ساخت مدل و تابع هدف از مراحل اولیه ساخت یک مساله یادگیری ماشین است و با مشخص شدن تابع هدف، از روش‌های بهینه‌سازی تحلیلی و عددی برای حل آن استفاده می‌کنیم. با توجه به مدل‌سازی و مساله بهینه‌سازی مربوطه، الگوریتم‌های یادگیری ماشین را می‌توان به یادگیری تحت نظارت، یادگیری نیمه نظارتی، یادگیری بدون نظارت و یادگیری تقویتی تقسیم کرد. یادگیری تحت نظارت به مسائل طبقه بندی و رگرسیون و یادگیری بدون نظارت به خوشه بندی و کاهش بعد تقسیم می‌شوند که جلوتر به آن‌ها می‌پردازیم [۱۰].

۱.۱ بهینه‌سازی در یادگیری تحت نظارت

در یادگیری تحت نظارت، هدف اصلی یافتن یک تابع بهینه است به طوری که تابع زیان از نمونه‌های آزمایشی را به حداقل برساند:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta)), \tag{۱.۱}$$

که N تعداد نمونه‌های آزمایشی، θ پارامتر تابع بهینه f، x_i بردار ویژگی نمونه i–ام، y_i برچسب متناظر نمونه i–ام و L تابع زیان است. در یادگیری تحت نظارت، از تابع های زیان می‌توان به فاصله اقلیدسی، آنتروپی متقاطع

و موارد دیگر اشاره کرد. معمولا در رگرسیون از مینیمم کردن مربع خطاها روی نمونه‌های آزمایشی برای تابع زیان استفاده می‌شود. فرم رایج دیگر، مینیمم کردن ریسک است که می‌توان برای آن روش ماشین بردار پشتیبان (SVM) را نام برد.

۲.۱ بهینه‌سازی در یادگیری نیمه نظارتی

یادگیری نیمه نظارتی روشی بین یادگیری تحت نظارت و بدون نظارت است، که داده برچسب دار و بدون برچسب را در طول مراحل آموزش یکپارچه می‌کند و شامل روش‌های مختلفی از جمله طبقه بندی، رگرسیون، خوشه بندی و کاهش بعد است. از روش ماشین بردار پشتیبان نیمه نظارتی برای معرفی بهینه‌سازی در یادگیری ماشین استفاده می‌کنیم. ماشین بردار پشتیبان نیمه نظارتی یک روش یادگیری است که می‌تواند با طبقه بندی دودویی و بخشی از مجموعه آموزشی که برچسب دارند، مساله را حل کند. مجموعه D^l و D^u را به ترتیب داده‌های برچسب زده شده و بدون برچسب در نظر بگیرید:

$$D^l = \{\{x^1, y^1\}, \{x^2, y^2\}, ..., \{x^l, y^l\}\}\,,\ D^u = \{x^{l+1}, x^{l+2}, ..., x^N\}$$

که $N = l + u$. با اضافه کردن متغیر کمکی ζ^i، از ϵ^i به عنوان خطای طبقه بندی متغیرهای بدون برچسب که برچسب واقعی آنها درست است و از z^i به عنوان خطای طبقه بندی غلط که برچسب واقعی آنها غلط است، استفاده می‌کنیم. بنابراین، مساله ماشین بردار پشتیبان نیمه نظارتی به صورت زیر در می‌آید:

$$\begin{aligned}
&\min \parallel \omega \parallel + C\left[\sum_{i=1}^{l}\zeta^i + \sum_{j=l+1}^{N}\min(\epsilon^i, z^i)\right], \\
&\text{subject to} \\
&y^i(w^T x^i + b) + \zeta^i \geq 1, \zeta \geq 0, i = 1, ..., l, \\
&w^T x^j + b + \epsilon^j \geq 1, \epsilon \geq 0, j = 1, ..., N, \\
&-(w^T x^i + b) + z^j \geq 1, z^j \geq 0,
\end{aligned} \tag{۲.۱}$$

که C یک ضریب جریمه است.

۳.۱ بهینه‌سازی در یادگیری بدون نظارت

الگوریتم خوشه بندی، نمونه‌ها را به چند خوشه تبدیل می‌کند به طوریکه اختلاف درون هر خوشه کمترین مقدار ممکن و اختلاف بین خوشه‌ها بیشترین حد ممکن را داشته باشند. خوشه بندی با رویکرد k-میانگین را می‌توان به صورت زیر مدل کرد:

$$\min_{S} \sum_{k=1}^{K} \sum_{x \in S_k} \| x - \mu_k \|_2^2, \tag{۳.۱}$$

که در آن K تعداد خوشه‌ها، x بردار ویژگی نمونه‌ها، μ_k مرکز خوشه k-ام و S_k مجموعه نمونه خوشه k-ام. مفهوم تابع هدف به حداقل رساندن مجموع واریانس‌های همه خوشه‌ها است.

الگوریتم کاهش بعد، اطلاعات اصلی داده‌ها را پس از تبدیل آنها به فضایی با بعد کمتر حفظ می‌کند. تحلیل مولفه اصلی [۱۲] از جمله این الگوریتم‌هاست که در آن تابع هدف برای مینیمم کردن خطای مدل به صورت زیر است:

$$\min \sum_{i=1}^{N} \| \bar{x}^i - x^i \|_2^2 \quad \text{where} \quad \bar{x}^i = \sum_{j=1}^{D'} z_j^i e_j, D \geq D' \tag{۴.۱}$$

که در آن، N تعداد نمونه‌ها، x_i بردار d بعدی، $\bar{x}_i$ بازسازی x_i، z_i تصویر در بعد d' و e_j پایه متعامد استاندارد در بعد d' می‌باشد.

٤.۱ بهینه‌سازی در یادگیری تقویتی

یادگیری تقویتی بر خلاف یادگیری تحت نظارت و بدون نظارت، به دنبال یافتن یک تابع استراتژی بهینه است که خروجی آن با محیط تغییر کند. برای یک استراتژی قطعی، تابعی پوشا از حالت s به عمل a یک هدف یادگیری است. برای یک استراتژی نامشخص، احتمال اجرای هر عمل، هدف یادگیری است. در هر حالت، تابع عمل به صورت $a = \pi(s)$ که $\pi(s)$ تابع سیاست است، مشخص می‌شود.

$$\max_{\pi} V_\pi(s) \quad \text{where} \quad V_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S_t = s\right], \tag{۵.۱}$$

که در آن، $V_\pi(s)$ تابع ارزش حالت s تحت سیاست π، r پاداش و $\gamma \in [0, 1]$ ضریب تنزیل است.

فصل ۲

روش‌ها و پیشرفت‌های بهینه‌سازی پایه

روش‌های پایه بهینه‌سازی را برحسب میزان استفاده از مشتق می‌توان به روش‌های بهینه‌سازی مرتبه اول، روش‌های بهینه‌سازی مرتبه بالا و روش‌های بهینه‌سازی بدون مشتق تقسیم کرد. این روش‌ها سابقه طولانی دارند و به طور مداوم در حال پیشرفت هستند. آن‌ها در بسیاری از کاربردهای عملی عملکرد خوبی داشته‌اند.
علاوه بر این روش‌های پایه، رویکرد پیش شرطی یک تکنیک مفید برای بهبود روش‌های بهینه‌سازی است. استفاده از پیش شرطی مناسب می‌تواند تعداد تکرارها را کاهش داده و ویژگی‌های طیفی بهتری بدست آورد. در انتهای این بخش، ابزارهای رایج بهینه‌سازی موجود را در جدولی خلاصه می‌کنیم.

۱.۲ روش‌های مرتبه اول

در زمینه یادگیری ماشین، بیش‌ترین استفاده از روش‌های بهینه‌سازی مرتبه اول عمدتاً بر اساس گرادیان کاهشی است. در این بخش، برخی از الگوریتم‌ها را همراه با توسعه روش‌های گرادیان کاهشی بیان کرده و در ادامه، روش مسیر متناوب ضرایب و روش فرانک–ولف در بهینه‌سازی عددی را معرفی می‌کنیم.

۱.۱.۲ گرادیان کاهشی

روش گرادیان کاهشی پایه‌ای ترین و رایج‌ترین روش بهینه‌سازی است. ایده روش گرادیان کاهشی این است که متغیرها به طور تکراری در جهت مخالف گرادیان تابع هدف به روز می‌شوند. مقدار تابع هدف به روز می‌شود تا

به تدریج به مقدار مطلوب همگرا شود.

الگوریتم تندترین کاهش [۲]، یک الگوریتم کاملاً شناخته شده است. ایده آن انتخاب یک جهت جستجوی مناسب در هر تکرار است به صورتیکه سریع‌تر به مینیمم مقدار تابع هدف برسیم. گرادیان کاهشی و تندترین کاهش یکسان نیستند، زیرا خلاف جهت گرادیان تابع هدف همیشه سریع‌ترین کاهش را نمی‌دهد. گرادیان کاهشی نمونه‌ای از استفاده از نرم اقلیدسی در تندترین کاهش است.

برای یک مدل رگرسیون خطی، فرض می‌کنیم $f_\theta(x)$ تابعی است که باید یاد بگیرد، $L(\theta)$ تابع زیان است و θ پارامتری است که باید بهینه شود. هدف این است که تابع زیان با رابطه زیر را مینیمم کنیم:

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y^i - f_\theta(x^i))^2, \tag{۱.۲}$$

$$f_\theta(x) = \sum_{j=1}^{D}\theta_j x_j, \tag{۲.۲}$$

که N، تعداد نمونه‌های آموزشی، D تعداد ویژگی‌های ورودی، x_i، متغیر مستقل با $x^i = (x_1^i, ..., x_D^i)$ برای $i = 1, ..., N$ و y^i خروجی هدف است. گرادیان کاهشی، دو مرحله زیر به طور متناوب طی می‌کند.

۱) مشتق از $L(\theta)$ را برای هر θ_j بدست بیاور:

$$\frac{\partial L(\theta)}{\partial \theta_j} = -\frac{1}{N}\sum_{i=1}^{N}(y^i - f_\theta(x^i))x_j^i, \tag{۳.۲}$$

۲) هر θ_j را در خلاف جهت گرادیان به روز رسانی کن تا تابع ریسک را به حداقل برسد:

$$\acute{\theta}_j = \theta_j + \eta.\frac{1}{N}\sum_{i=1}^{N}(y^i - f_\theta(x^i))x_j^i. \tag{۴.۲}$$

نرخ یادگیری توسط پارامتر η کنترل می‌شود. پیاده‌سازی روش گرادیان کاهشی ساده است و وقتی تابع هدف محدب باشد، جواب بدست آمده بهینه سراسری است.

در مثال رگرسیون خطی بالا، توجه داشته باشید که از تمام داده‌های آموزش در هر مرحله تکرار استفاده می‌شود. بنابراین به روش گرادیان کاهشی، گرادیان کاهشی انبوه نیز گفته می‌شود. اگر تعداد نمونه‌ها N و بُعد x، D باشد، پیچیدگی محاسبات برای هر تکرار، $O(ND)$ خواهد بود. با این حال، کار با داده‌ها در مقیاس بزرگ هنوز دشوار است. بنابراین، روش گرادیان کاهشی تصادفی پدیدار می‌شود.

۲.۱.۲ گرادیان کاهشی تصادفی

از آن‌جا که گرادیان کاهشی در هر تکرار برای داده‌های در مقیاس بزرگ دارای پیچیدگی محاسباتی زیادی است، گرادیان کاهشی تصادفی (SGD) پیشنهاد شد. ایده گرادیان کاهشی تصادفی استفاده از یک نمونه به طور تصادفی برای به روزرسانی گرادیان در هر تکرار به جای محاسبه مستقیم مقدار دقیق گرادیان است. الگوریتم گرادیان کاهشی تصادفی مستقل از تعداد نمونه‌ها است و می‌تواند به سرعت همگرایی زیرخطی برسد. گرادیان کاهشی تصادفی زمان به روزرسانی را هنگامی که تعداد نمونه‌ها زیاد است، کاهش می‌دهد و مقداری از افزونگی محاسباتی را حذف می‌کند، که به طور قابل توجهی محاسبه را سریع‌تر می‌کند.

تابع زیان (۱.۲) را می‌توان به صورت معادله زیر نوشت:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^i - f_\theta(x^i))^2 = \frac{1}{N}\sum_{i=1}^{N}\mathrm{cost}(\theta,(x^i,y^i)). \qquad (۵.۲)$$

که در آن تابع cost، تابع هزینه است. اگر یک نمونه تصادفی i در گرادیان کاهشی تصادفی انتخاب شود، تابع زیان، $L^*(\theta)$ خواهد بود:

$$L^*(\theta) = \mathrm{cost}(\theta,(x^i,y^i)) = \frac{1}{2}(y^i - f_\theta(x^i))^2 \qquad (۶.۲)$$

به روزرسانی گرادیان در گرادیان کاهشی تصادفی به جای همه نمونه‌ها در هر تکرار، از نمونه تصادفی i استفاده می‌کند،

$$\acute{\theta} = \theta + \eta(y^i - f_\theta(x^i))x^i. \qquad (۷.۲)$$

از آن‌جا که گرادیان کاهشی تصادفی در هر تکرار فقط از یک نمونه استفاده می‌کند، پیچیدگی محاسباتی برای هر تکرار، $O(D)$ است که D تعداد ویژگی‌ها است. وقتی تعداد نمونه‌ها زیاد است سرعت به روزرسانی برای هر تکرار گرادیان کاهشی تصادفی بسیار سریع‌تر از گرادیان کاهشی است. گرادیان کاهشی تصادفی تکرارها را افزایش می‌دهد، اما تعداد تکرار افزایش یافته در مقایسه با پیچیدگی محاسباتی بالای ناشی از تعداد زیادی نمونه، ناچیز است. بنابراین، در مقایسه با روش‌های انبوه، گرادیان کاهشی تصادفی می‌تواند به طور موثری پیچیدگی محاسباتی را کاهش داده و همگرایی را تسریع بخشد.

با این حال، یک مشکل در گرادیان کاهشی تصادفی این است که جهت گرادیان به دلیل نویز اضافی که با انتخاب

تصادفی وارد می‌شود، در نوسان است و بر خلاف گرادیان کاهشی که همیشه در امتداد جهت منفی گرادیان به سمت مقدار مطلوب حرکت می‌کند، واریانس گرادیان‌ها در گرادیان کاهشی تصادفی زیاد و جهت حرکت در گرادیان کاهشی تصادفی ناپایدار است. برای سازش بین دو روش، روش گرادیان کاهشی نیمه انبوه (MSGD)، پیشنهاد شد.

MSGD از b نمونه توزیع شده یکسان و مستقل (b معمولاً از ۵۰ تا ۲۵۶ است) به عنوان مجموعه نمونه برای به روزرسانی پارامترها در هر تکرار استفاده می‌کند. واریانس گرادیان‌ها را کاهش می‌دهد و همگرایی را پایدارتر می‌کند که به بهبود سرعت بهینه‌سازی کمک می‌کند. برای اختصار، ما در بخش‌های بعدی MSGD را گرادیان کاهشی تصادفی می‌نامیم [۲، ۱۸].

گرادیان کاهشی تصادفی شانس بیش‌تری برای یافتن جواب بهینه سراسری برای مسائل پیچیده دارد. گرادیان کاهشی ممکن است باعث شود که تابع هدف برای یک مساله چند حالته در مینیمم موضعی قرار گیرد. نوسان در گرادیان کاهشی تصادفی کمک می‌کند تا تابع هدف به مینیمم ممکن دیگر برسد. با این حال نوسان در گرادیان کاهشی تصادفی ممکن است روند همگرایی را کم و بیش کُند کند.

در مورد استفاده از گرادیان کاهشی تصادفی در فرآیند بهینه‌سازی، مورد مهم دیگر، نرخ یادگیری است. نرخ یادگیری خیلی کم، منجر به کاهش سرعت همگرایی می‌شود، در حالی که نرخ یادگیری بسیار زیاد، مانع از همگرایی می‌شود و تابع زیان را در کمینه نوسان می‌دهد. یکی از راه‌حل‌های این مساله، تعیین یک لیست از پیش تعیین شده از نرخ یادگیری و تنظیم آن در طی فرایند یادگیری است. با این حال، این لیست‌ها یا آستانه‌ها باید با توجه به ویژگی‌های مجموعه داده از قبل تعریف شوند. هم‌چنین استفاده از نرخ یادگیری یکسان برای همه پارامترها نامناسب است. اگر داده‌ها کم باشد و ویژگی‌ها در فرکانس‌های مختلف وجود داشته باشد، انتظار نمی‌رود متغیرهای مربوطه را با همان نرخ یادگیری به روز کند. معمولاً انتظار می‌رود که ویژگی‌های کم‌تر اتفاق افتاده، نرخ یادگیری بالاتری داشته باشند.

علاوه بر نرخ یادگیری، چگونگی جلوگیری از گرفتار شدن تابع هدف در تعداد بی نهایت مینیمم موضعی، یک چالش رایج است. برخی پژوهش‌ها نشان داده‌اند که این مشکل از مقادیر مینیمم موضعی ناشی نمی‌شود، بلکه از نقطه زینی ناشی می‌شود. گرادیان یک نقطه زینی در یک جهت مثبت و در جهت دیگر منفی است و مقادیر گرادیان در همه جهات صفر است. فرار از این نقاط برای گرادیان کاهشی تصادفی مساله مهمی است.

۳.۱.۲ گرادیان کاهشی شتابدار نستروف

اگرچه گرادیان کاهشی تصادفی محبوب است و به طور گسترده‌ای مورد استفاده قرار می‌گیرد، اما روند یادگیری آن گاهی طولانی می‌شود. چگونگی تنظیم نرخ یادگیری، سرعت بخشیدن به همگرایی و جلوگیری از گرفتار شدن در مینیمم موضعی از اهداف مهم این روش هستند. کارهای زیادی برای بهبود گرادیان کاهشی تصادفی ارائه شده است. به عنوان مثال، ایده تکانه یا گشتاور پیشنهاد شد. مفهوم گشتاور از مکانیک و فیزیک گرفته شده است که اینرسی اجسام را شبیه‌سازی می‌کند. ایده استفاده از گشتاور در گرادیان کاهشی تصادفی تا حدودی، حفظ تأثیر جهت به روزرسانی در تکرار بعدی است.

روش گشتاور در هنگام برخورد با انحنای زیاد می‌تواند همگرایی را تسریع بخشد. الگوریتم گشتاور، متغیر v را به عنوان سرعت معرفی می‌کند، که نشان دهنده جهت و سرعت حرکت است. در روش گرادیان کاهشی، به روز رسانی سرعت، $v = \eta.(-\frac{L(\theta)}{\partial\theta})$ در هر زمان است. با استفاده از الگوریتم گشتاور، مقدار به روزرسانی v فقط مقدار گرادیان کاهشی محاسبه شده توسط $\eta.(-\frac{\partial L(\theta)}{\partial\theta})$ نیست و عامل ضریب اصطکاک را نیز در نظر می‌گیرد که به عنوان به روزرسانی قبلی v^{old} ضرب شده در یک ضریب گشتاور بین $[0\ ،\ 1]$ نشان داده می‌شود. به طور کلی، فرمول به صورت زیر بیان می‌شود:

$$v = \eta.(-\frac{\partial L(\theta)}{\partial\theta}) + v^{old}.mtm, \tag{۸.۲}$$

به طوری که mtm ضریب گشتاور است. اگر گرادیان فعلی با سرعت قبلی v^{old} موازی باشد، سرعت قبلی می‌تواند این جستجو را سریع کند. گشتاور مناسب در سرعت بخشیدن به همگرایی هنگامی که نرخ یادگیری کم است، نقش دارد. اگر مشتق به صفر میل کند، برای رسیدن به تعادل هم‌چنان به روزرسانی می‌شود و با اصطکاک کاهش می‌یابد. اگر گرادیان فعلی مخالف v^{old} به روزرسانی قبلی باشد، مقدار v^{old} تأثیر کمی در این جستجو خواهد داشت.

روش گشتاور با یک ضریب گشتاور مناسب در کاهش نوسان همگرایی وقتی که نرخ یادگیری زیاد است، نقش مثبتی دارد. نحوه انتخاب اندازه مناسب ضریب گشتاور نیز مسئله ساز است. اگر ضریب گشتاور کوچک باشد، بدست آوردن اثر بهبود سرعت همگرایی دشوار است. اگر ضریب گشتاور بزرگ باشد، ممکن است نقطه فعلی از بهینگی خارج شود. بسیاری از آزمایش‌های عددی تأیید کرده‌اند که مناسب‌ترین تنظیم برای ضریب گشتاور ۰.۹ است.

گرادیان کاهشی شتاب یافته نستروف (NAG) نسبت به روش گشتاور قبلی پیشرفت بیش‌تری می‌کند. در گشتاور نستروف، گشتاور $v^{old}.mtm$ به θ اضافه می‌شود، که به صورت $\tilde{\theta}$ نشان داده می‌شود. هنگام به روزرسانی از گرادیان $\tilde{\theta}$ استفاده می‌شود. فرمول‌های دقیق به روزرسانی برای پارامترهای θ به شرح زیر است:

$$\begin{cases} \tilde{\theta} = \theta + v^{old}.mtm, \\ v = v^{old}.mtm + \eta. - (\frac{\partial L(\tilde{\theta})}{\partial \theta}) \\ \acute{\theta} = \theta + v \end{cases} \quad (۹.۲)$$

بهبود گشتاور نستروف بیش از گشتاور در به روزرسانی گرادیان است. از فرمول به روزرسانی می‌توان دریافت که گشتاور نستروف در مقایسه با روش گشتاور قبلی شامل اطلاعات گرادیان بیش‌تری است. توجه داشته باشید که گشتاور نستروف هنگام استفاده از بهینه‌سازی تصادفی، همگرایی را از $O(\frac{1}{k})$ (بعد از k مرحله) به $O(\frac{1}{k^2})$ بهبود می‌بخشد.

موضوع قابل توجه دیگر، چگونگی تعیین اندازه نرخ یادگیری است. اگر جستجو به نقطه بهینه نزدیک‌تر باشد احتمال وقوع نوسان بیش‌تر است. بنابراین، نرخ یادگیری باید تنظیم شود. ضریب نزول نرخ یادگیری d معمولاً در روش گشتاور گرادیان کاهشی تصادفی استفاده می‌شود، که باعث می‌شود با دوره تکرار نرخ یادگیری، کاهش یابد. فرمول نزول نرخ یادگیری به صورت زیر تعریف شده است:

$$\eta_t = \frac{\eta_0}{1 + d.\acute{t}} \quad (۱۰.۲)$$

که در آن η_t، نرخ یادگیری در t-امین تکرار، η_0، نرخ یادگیری اصلی و d عدد اعشاری در بازه $[۰،۱]$ است. همان‌طور که از فرمول دیده می‌شود، هرچه d کوچک‌تر باشد، نزول سرعت یادگیری نیز کندتر خواهد بود. نرخ یادگیری هنگامی که $d = 0$ است بدون تغییر باقی می‌ماند و هنگامی که $d = 1$ نرخ یادگیری با سرعت کم‌تری کاهش می‌یابد.

٤.١.٢ روش نرخ یادگیری تطبیقی

نرخ یادگیری تنظیم شده به صورت دستی به میزان زیادی بر تأثیر روش گرادیان کاهشی تصادفی موثر است. تعیین مقدار مناسب نرخ یادگیری یک مسئله پیچیده است. برخی از روش‌های انطباقی برای تنظیم خودکار نرخ یادگیری

پیشنهاد شده است. این روش‌ها فاقد تنظیم پارامتر هستند، به سرعت همگرا می‌شوند و اغلب نتایج بدی به همراه ندارند. آن‌ها به طور گسترده‌ای در شبکه‌های عصبی عمیق استفاده می‌شوند.

ساده‌ترین پیشرفت در گرادیان کاهشی تصادفی، AdaGrad [۸] است. AdaGrad نرخ یادگیری را بر اساس گرادیان قبلی در برخی از تکرارهای گذشته تنظیم می‌کند. فرمول‌های به روزرسانی به شرح زیر است:

$$\begin{cases} g_t = \frac{\partial L(\theta_t)}{\partial \theta}, \\ V_t = \sqrt{\sum_{i=1}^{t} (g_i)^2 + \epsilon} \\ \theta_t + 1 = \theta_t - \eta \frac{g_t}{V_t} \end{cases} \tag{۱۱.۲}$$

به طوری که g_t گرادیان پارامتر θ در تکرار t، V_t جمع گرادیان قبلی پارامتر θ در تکرار t، و θ_t مقدار پارامتر θ در تکرار t است.

تفاوت بین AdaGrad و گرادیان کاهشی این است که طی فرآیند به روز رسانی پارامتر، نرخ یادگیری دیگر ثابت نیست، بلکه با استفاده از تمام گرادیان‌های قبلی جمع شده تا این تکرار محاسبه می‌شود. یکی از مزایای اصلی AdaGrad این است که نیاز به تنظیم دستی نرخ یادگیری را از بین می‌برد. اکثر پیاده‌سازی‌ها از مقدار پیش فرض ۰.۰۱ برای η در بالا استفاده می‌کنند.

اگرچه AdaGrad به طور انطباقی نرخ یادگیری را تنظیم می‌کند، اما هم‌چنان دو مسئله دارد. ۱) الگوریتم هنوز باید نرخ یادگیری سراسری η را به صورت دستی تنظیم کند. ۲) با افزایش زمان آموزش، گرادیان جمع شده بزرگ و بزرگ‌تر می‌شود و باعث می‌شود که نرخ یادگیری به صفر برسد، در نتیجه به روزرسانی پارامتر بی‌اثر می‌شود.

AdaGrad برای حل مسئله به AdaDelta [۲۲] و RMSProp ارتقا یافته است. ایده این است که تمام گرادیان‌های قبلی را جمع نکنید، بلکه فقط بر روی گرادیان‌های در یک دوره تمرکز کنید:

$$V_t = \sqrt{\beta V_t - 1 + (1 - \beta)(g_t)^2}, \tag{۱۲.۲}$$

به طوری که β، پارامتر تنزیل است. RMSProp و AdaDelta هر دو به طور مستقل در همان زمان توسعه یافته‌اند و این از نیاز به حل و فصل کاهش نرخ یادگیری AdaGrad ناشی می‌شود. برآورد لحظه‌ای تطبیقی (Adam) [۱۴] یکی دیگر از روش‌های پیشرفته گرادیان کاهشی تصادفی است که برای هر پارامتر نرخ یادگیری

تطبیقی را معرفی می‌کند. این روش، ترکیبی از روش یادگیری انطباقی و روش‌های گشتاور است:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{۱۳.۲}$$

$$V_t = \beta_2 V_{t-1} + (1-\beta_2)(g_t)^2, \tag{۱۴.۲}$$

به طوری که β_1 و β_2 نرخ‌های تنزیل هستند. فرمول به روز رسانی شده نهایی برای پارامتر θ عبارت است از:

$$\theta_{t+1} = m_t - \eta \frac{\sqrt{1-\beta_2}}{1-\beta_1} \frac{m_t}{V_t + \epsilon} \tag{۱۵.۲}$$

پیشنهاد می‌شود که مقادیر پیش فرض β_1 و β_2 و ϵ به ترتیب به مقادیر ۰.۹، ۰.۹۹۹ و 10^{-8} تنظیم شود. Adam در عمل خوب کار می‌کند و در مقایسه با دیگر الگوریتم‌های یادگیری انطباقی نتایج مطلوبی دارد.

۵.۱.۲ روش‌های کاهش واریانس

با توجه به مقدار زیاد اطلاعات در نمونه‌های آموزشی، روش‌های گرادیان کاهشی تصادفی از زمانی که پیشنهاد شده‌اند بسیار مورد استفاده هستند ولی فقط می‌توانند با نرخ زیر خطی همگرا شود و واریانس گرادیان اغلب بسیار زیاد است. چگونگی کاهش واریانس و بهبود گرادیان کاهشی تصادفی به همگرایی خطی همیشه یک مسئله مهم بوده است.

روش میانگین گرادیان تصادفی (SAG) یک روش کاهش واریانس است که برای بهبود سرعت همگرایی پیشنهاد شده است. الگوریتم میانگین گرادیان تصادفی، پارامتر d را طوری می‌سازد که مجموع N گرادیان آخر را در حافظه ذخیره کند به صورتی که g_i با استفاده از یک نمونه تصادفی $i \in \{1, ..., N\}$ محاسبه شده باشد. پیاده‌سازی دقیق این است که نمونه‌ای از آن را انتخاب کنید تا d به صورت تصادفی به روز شود و از d برای به روز رسانی پارامتر θ در تکرار t استفاده کنید:

$$\begin{cases} d = d - \hat{g}_{i_t} + g_{i_t}(\theta_{t-1}), \\ g_{i_t} = g_{i_t}(\theta_{t-1}), \\ \theta_t = \theta_{t-1} - \frac{\alpha}{N} d \end{cases} \tag{۱۶.۲}$$

به طوری که آیتم به روز رسانی شده d، با جایگزینی گرادیان قدیمی g_{i_t} در d با $g_{i_t}(\theta_{t-1})$ در تکرار t محاسبه می‌شود. α یک ثابت است که نرخ یادگیری را نشان می‌دهد. بنابراین، هر به روزرسانی فقط نیاز به محاسبه گرادیان یک نمونه دارد، نه گرادیان همه نمونه‌ها. از نظر محاسباتی تفاوتی با گرادیان کاهشی تصادفی ندارد، اما از نظر حافظه بسیار بزرگ‌تر است. این یک روش معمول استفاده از فضا برای صرفه‌جویی در زمان است. نشان داده شده است که میانگین گرادیان تصادفی یک الگوریتم با همگرایی خطی است که بسیار سریع‌تر از گرادیان کاهشی تصادفی است و دارای مزایای زیادی نسبت به الگوریتم های گرادیان تصادفی دیگر است. با این حال فقط در مواردی کاربرد دارد که تابع زیان هموار و تابع هدف محدب باشد، مانند مسائل پیش‌بینی خطی محدب. در این حالت، میانگین گرادیان تصادفی به سرعت همگرایی بیش‌تری نسبت به گرادیان کاهشی تصادفی دست می‌یابد. علاوه بر این، تحت برخی مسائل خاص، حتی می‌تواند همگرایی بهتری نسبت به گرادیان کاهشی استاندارد داشته باشد.

کاهش واریانس گرادیان تصادفی از آن‌جا که روش میانگین گرادیان تصادفی فقط در توابع هموار و محدب کاربرد دارد و نیاز به ذخیره گرادیان هر نمونه دارد، استفاده از آن در شبکه‌های عصبی غیر محدب مناسب نیست. روش کاهش واریانس گرادیان تصادفی (SVRG) برای بهبود عملکرد بهینه‌سازی در مدل‌های پیچیده ارائه شده است.

الگوریتم SVRG با محاسبه گرادیان همه نمونه‌ها در هر w تکرار به جای هر تکرار، بازه میانگین گرادیان $\tilde{\mu}$ را حفظ می‌کند:

$$\tilde{\mu} = \frac{1}{N}\sum_{i=1}^{N} g_i(\tilde{\theta}) \qquad (۱۷.۲)$$

به طوری که $\tilde{\theta}$، پارامتر به روزرسانی فاصله است. پارامتر فاصله $\tilde{\mu}$ شامل متوسط تمام گرادیان نمونه‌ها در زمان گذشته برای هر بازه زمانی w است. SVRG به طور تصادفی $i_t \in \{1, ..., N\}$ یکنواخت انتخاب می‌کند و به روزرسانی‌های گرادیان را برای پارامترهای فعلی اجرا می‌کند:

$$\theta_t = \theta_{t-1} - \eta.(g_{i_t}(\theta_{t-1})) - g_{i_t}(\tilde{\theta}) + \tilde{\mu}) \qquad (۱۸.۲)$$

گرادیان در هر به روز رسانی حداکثر دو بار محاسبه می‌شود. بعد از w تکرار، $\tilde{\theta} \leftarrow \theta_w$ را انجام دهید و w تکرار بعدی را شروع کنید. از طریق این به روزرسانی، θ و پارامتر به روزرسانی فاصله $\tilde{\theta}$ به θ^* بهینه و سپس $\tilde{\mu} \to 0$ و

رابطه زیر همگرا می‌شود:

$$g_{i_t}(\theta_{t-1}) - g_{i_t}(\tilde{\theta}) + \tilde{\mu} \to g_{i_{t-1}}(\theta_{t-1}) - g_{i_t}(\theta^*) \to 0 \qquad (۱۹.۲)$$

SVRG مفهومی حیاتی به نام کاهش واریانس را پیشنهاد می‌کند. این مفهوم مربوط به تجزیه و تحلیل همگرایی گرادیان کاهشی تصادفی است که در آن لازم است فرض کنیم که یک حد بالایی ثابت برای واریانس گرادیان‌ها وجود دارد. این حد بالای ثابت نشان می‌دهد که گرادیان کاهشی تصادفی نمی‌تواند به همگرایی خطی دست یابد. با این حال، در SVRG، می‌توان حد بالای واریانس را به دلیل آیتم ویژه به روزرسانی $g_{i_t}(\theta_{t-1}) - g_{i_t}(\tilde{\theta}) + \tilde{\mu}$ به طور مداوم کاهش داد؛ بنابراین به همگرایی خطی دست می‌یابد. استراتژی‌های SAG و SVRG مربوط به کاهش واریانس است. در مقایسه با میانگین گرادیان تصادفی، SVRG نیازی به نگه داشتن تمام گرادیان‌ها در حافظه ندارد، به این معنی که منابع حافظه ذخیره می‌شوند و می‌توان آن‌ها را برای مسائل پیچیده به طور کارآمد اعمال کرد. آزمایشات نشان داده است که عملکرد SVRG در یک شبکه عصبی غیر محدب قابل توجه است.

۶.۱.۲ روش مسیر متناوب ضرایب

روش ضرایب لاگرانژ تعمیم یافته یک روش رایج برای حل مسائل بهینه‌سازی با قیدهای خطی است. این روش در مقایسه با روش ضرایب لاگرانژی ساده، مسائل را با افزودن یک جمله جریمه به تابع هدف، آسان‌تر حل می‌کند. مثال زیر را در نظر بگیرید،

$$min\left\{\theta_1(x) + \theta_2(y) | Ax + By = b, x \in X, y \in Y\right\} \qquad (۲۰.۲)$$

تابع لاگرانژ تعمیم یافته برای مسئله بالا به صورت زیر است:

$$\mathcal{L}_\beta(x, y, \lambda) = \theta_1(x) + \theta_2(y) - \lambda^T(Ax + By - b) + \frac{\beta}{2}||Ax + By - b||^2 \qquad (۲۱.۲)$$

وقتی با روش ضرایب لاگرانژی تعمیم یافته حل شد، تکرار مرحله t_{th} آن از λ_t داده شده شروع می‌شود و بهینه‌سازی به نتیجه زیر می‌رسد:

$$\begin{cases} (x_{t+1}, y_{t+1}) = argmin\left\{\mathcal{L}_\beta(x, y, \lambda_t) | x \in X, y \in Y\right\}, \\ \lambda_{t+1} = \lambda_t - \beta(Ax_{t+1} + y_{t+1} - b). \end{cases} \qquad (۲۲.۲)$$

با جدا کردن (x, y) ، روش ضرایب لاگرانژ تعمیم یافته را می‌توان به روش مسیر متناوب ضرایبADMM [۳] کاهش داد. تکرار مرحله t_{th} آن با (y_t, λ_t) داده شده شروع می‌شود و به شرح زیر است:

$$\begin{cases} x_{t+1} = argmin \left\{ \theta_1(x) - (\lambda_t)^T Ax + \frac{\beta}{2} ||con_x||^2 | x \in X \right\}, \\ y_{t+1} = argmin \left\{ \theta_2(y) - (\lambda_t)^T Bx + \frac{\beta}{2} ||con_y||^2 | y \in Y \right\} \\ \lambda_{t+1} = \lambda_t - \beta(Ax_{t+1} + By_{t+1} - b), \end{cases} \quad (۲۳.۲)$$

به طوری که $con_x = Ax + By_t - b$ و $con_y = Ax_{t+1} + By - b$.

پارامتر جریمه β بر میزان همگرایی ADMM تاثیر دارد. هرچه β بزرگ‌تر باشد، جریمه‌های جمله ثابت بیش‌تر است. به طور کلی، یک دنباله افزایشی $\{\beta_t\}$ می‌تواند به جای β ثابت جایگزین شود. به ویژه، یک معیار تنظیم خودکار که $\{\beta_t\}$ را بر اساس مقدار فعلی $\{x_t\}$ در طول تکرار تنظیم می‌کند، برای حل برخی از مسائل بهینه‌سازی محدب پیشنهاد می‌شود.

روش ADMM در مسئله بهینه‌سازی محدب، یک مسئله بزرگ را به چند مسئله کوچک تقسیم می‌کند و سپس آنها را حل کند. از لحاظ تئوری، چارچوب ADMM می‌تواند بسیاری از مسائل بهینه‌سازی در مقیاس بزرگ را حل کند. با این حال، هنوز هم برخی از مشکلات در کاربردهای عملی وجود دارد. به عنوان مثال، اگر از معیار توقف برای تعیین وجود همگرایی استفاده کنیم، باقی مانده‌های اصلی و باقی مانده‌های دوگان هر دو مربوط به β هستند و منجر به مشکل در برآورده شدن شرایط همگرایی خواهد شد.

۷.۱.۲ روش فرانک-ولف

در سال ۱۹۵۶، فرانک و ولف الگوریتمی را برای حل مسائل با قیود خطی پیشنهاد کردند. ایده اصلی تقریب تابع هدف با یک تابع خطی، سپس حل برنامه‌ریزی خطی برای یافتن جهت کاهنده و در نهایت یک جستجوی یک بُعدی در راستای جهت در ناحیه شدنی است. این روش را روش خطی ساخت تقریبی نیز می‌نامند.

در این‌جا یک مثال ساده از روش فرانک- ولف ارائه می‌دهیم. مساله بهینه‌سازی زیر را در نظر بگیرید:

$$\begin{cases} min & f(x), \\ s.t. & A(x) = b, \\ & x \geq 0, \end{cases} \tag{۲۴.۲}$$

به طوری که A، یک ماتریس تمام رتبه سطری $m \times n$ و ناحیه شدنی، $S = \{x | Ax = b, x \geq 0\}$ است. تابع f را به صورت خطی در x_0 بسط دهید، $f(x) \approx f(x_0) + \nabla f(x)^T (x - x_0)$، و آن را در مساله بالا جایگزین کنید. پس داریم:

$$\begin{cases} min & f(x_t) + \nabla f(x_t)^T (x - x_t), \\ s.t. & x \in S \end{cases} \tag{۲۵.۲}$$

که معادل است با

$$\begin{cases} min & \nabla f(x_t)^T x, \\ s.t. & x \in S \end{cases} \tag{۲۶.۲}$$

بردار x_t نقطه متناظر با تکرار t-ام است. فرض کنید y_t یک جواب بهینه برای مساله بالا باشد، پس داریم:

$$\begin{cases} \nabla f(x_t)^T y_t < \nabla f(x_t)^T x_t, \\ \nabla f(x_t)^T (y_t - x_t) < 0 \end{cases} \tag{۲۷.۲}$$

بنابراین $y_t - x_t$ یک جهت کاهنده f در x_t است. مراحل این روش در الگوریتم ۱ نشان داده شده است.

الگوریتم خواص زیر را دارد.

(۱) x_t نقطه کان- تاکر (۲۴.۲) است هنگامی که $\nabla f(x_t)^T (y_t - x_t) = 0$.

(۲) از آن‌جا که y_t یک جواب بهینه برای مسئله (۲۶.۲) است، بردار d_t حاصل $d_t = y_t - x_t$ است و جهت کاهنده f در نقطه x_t است مادامی که $\nabla f(x_t)^T (y_t - x_t) \neq 0$.

الگوریتم ۱ روش فرانک-ولف

Input: $x_0, \epsilon \geq 0, t := 0$

Output: x^*

$y_t \leftarrow min \nabla f(x_t)^T x$

while $|\nabla f(x_t)^T (y_t - x_t)| > \epsilon$ **do**

$\lambda_t = argmin_{0 \leq \lambda \leq 1} f(x_t + \lambda(y_t - x_t))$

$x_{t+1} \approx x_t + \lambda_t (y_t - x_t)$

$t := t + 1$

$y_t \rightarrow min \nabla f(x_t)^T x$

end while

$x^* \approx x_t$

الگوریتم فرانک- ولف یک روش تکراری مرتبه اول برای حل مسائل بهینه‌سازی محدب با شرایط محدود است. این الگوریتم شامل تعیین جهت نزولی عملی و محاسبه اندازه گام جستجو است. این الگوریتم با همگرایی سریع در تکرارهای اولیه و کندتر در مراحل بعدی شناخته می‌شود. هنگامی که نقطه تکرار نزدیک به جواب بهینه باشد، جهت جستجو و جهت گرادیان تابع هدف به صورت متعامد است. چنین جهتی بهترین جهت رو به پایین نیست تا الگوریتم فرانک-ولف از نظر انتخاب جهت‌های نزولی بهبود یافته و گسترش یابد.

۸.۱.۲ خلاصه

روش‌های بهینه‌سازی مرتبه اول بیان شده را از نظر ویژگی‌ها، مزایا و معایب در جدول ۱.۲ خلاصه می‌کنیم.

جدول ۱.۲: خلاصه‌ای از روش‌های بهینه‌سازی مرتبه اول

روش	ویژگی‌ها	مزایا	معایب
GD	مساله را در راستای جهت گرادیان کاهشی حل می‌کند. این روش با سرعت خطی همگرا می‌شود.	وقتی تابع هدف محدب است، جواب بهینه سراسری است.	در هر به روزرسانی، گرادیان کل نمونه‌ها باید محاسبه شود، بنابراین محاسبات زیاد است.
SGD	پارامترهای به روزرسانی با استفاده از نمونه‌گیری تصادفی محاسبه می‌شوند. این روش با سرعت زیرخطی همگرا می‌شود.	زمان محاسبه هر به روزرسانی به تعداد کل نمونه‌های آموزش بستگی ندارد و در هزینه‌های زیاد محاسبه صرفه جویی می‌شود.	انتخاب نرخ یادگیری مناسب دشوار است و استفاده از نرخ یادگیری یکسان برای همه پارامترها مناسب نیست و ممکن است در نقطه زینی به دام بیفتد.
NAG	با جمع کردن گرادیان قبلی به عنوان تکانه، گرادیان کاهشی فعلی را تسریع کرده و به روزرسانی گرادیان را با تکانه انجام می‌دهد .	هنگامی که جهت گرادیان تغییر می‌کند، می‌تواند سرعت به روزرسانی و نوسان را کاهش دهد. وقتی جهت گرادیان باقی بماند، می‌تواند به روزرسانی پارامتر را سریع کند.	انتخاب نرخ یادگیری مناسب دشوار است.
AdaGrad	نرخ یادگیری با توجه به جمع مربع‌های تمام گرادیان‌های قبلی تنظیم می‌شود.	در مرحله اولیه آموزش، مجموع گرادیان کوچک‌تر، نرخ یادگیری و سرعت یادگیری سریع‌تر است. نرخ یادگیری هر پارامتر تطبیقی تنظیم می‌شود.	با افزایش زمان آموزش، گرادیان جمع شده بزرگ‌تر خواهد شد و باعث صفر شدن نرخ یادگیری می‌شود. برای مسائل غیر محدب مناسب نیست.

روش	ویژگی‌ها	مزایا	معایب
AdaDelta RMSProp	تغییر روش جمع گرادیان از جمع گرادیان کل به میانگین نمایی.	مشکل یادگیری در اواخر مرحله AdaGrad را بهبود بخشید. برای بهینه‌سازی مسائل غیرمحدب مناسب است.	در اواخر مرحله آموزش، ممکن است روند به روزرسانی در مینیمم موضعی تکرار شود.
Adam	روش‌های تطبیقی و تکانه را ترکیب می‌کند. برای تنظیم نرخ یادگیری هر پارامتر، از تقریب لحظه ای مرتبه اول و تقریب تکانه مرتبه دوم استفاده می‌کند.	روند گرادیان کاهشی نسبتاً پایدار است. برای اکثر مسائل بهینه‌سازی غیر محدب با مجموعه داده‌های بزرگ و فضای بعد بالا مناسب است.	این روش ممکن است در برخی موارد همگرا نباشد.
SAG	گرادیان قبلی هر نمونه و جمع گرادیان‌های همه نمونه‌ها در حافظه حفظ می‌شود. برای هر به روزرسانی، یک نمونه به طور تصادفی انتخاب می‌شود. مقدار گرادیان مجدداً محاسبه و به عنوان جهت استفاده می‌شود.	یک الگوریتم همگرای خطی است که بسیار سریع‌تر از گرادیان کاهشی تصادفی است.	این روش فقط برای توابع هموار و محدب قابل استفاده است و نیاز به ذخیره گرادیان هر نمونه را دارد. استفاده از آن در شبکه‌های عصبی غیر محدب مناسب نیست.
SVRG	به جای گرادیان هر نمونه، گرادیان متوسط در بازه‌های مشخص ذخیره می‌شود. جمع گرادیان در هر تکرار با محاسبه گرادیان‌ها با توجه به پارامترهای قبلی و پارامترهای فعلی برای نمونه‌های تصادفی انتخاب شده به روز می‌شود.	این روش نیازی به حفظ تمام گرادیان‌های حافظه ندارد که باعث صرفه جویی در منابع حافظه می‌شود. یک الگوریتم همگرای خطی است.	برای استفاده از آن در شبکه‌های عصبی بزرگ‌تر و عمیق‌تر که هزینه آموزش آن‌ها مسئله مهمی است، هنوز تحقیقات بیش‌تری لازم است.

روش	ویژگی‌ها	مزایا	معایب
ADMM	این روش با اضافه کردن یک جریمه به تابع هدف و تفکیک متغیرها به ریز مسائل که با تکرار قابل حل هستند، مسائل بهینه‌سازی را با محدودیت‌های خطی حل می‌کند.	این روش با استفاده از عملگرهای قابل تفکیک در مسئله بهینه‌سازی محدب، یک مسئله بزرگ را به چند مسئله کوچک تقسیم می‌کند. برای حل بیش‌تر مسائل بهینه‌سازی مقیاس بزرگ عملی است.	مانده اصلی و مانده دوگان هر دو مربوط به پارامتر جریمه است که تعیین مقدار آن دشوار است.
Frank-Wolfe	این روش تابع هدف را با یک تابع خطی تقریب می‌زند. برنامه‌ریزی خطی را برای یافتن جهت کاهشی حل می‌کند و یک جستجوی یک بُعدی را در امتداد جهت در فضای شدنی انجام می‌دهد.	این روش می‌تواند مسائل بهینه‌سازی با محدودیت‌های خطی را که سرعت همگرایی آن‌ها در تکرارهای اولیه سریع است، حل کند.	در این روش هنگامی که نقطه تکرار نزدیک به جواب بهینه باشد، جهت جستجو و گرادیان تابع هدف به صورت متعامد است. چنین جهتی مطلوب نیست.

۲.۲ روش‌های مرتبه بالا

از روش‌های مرتبه دوم می‌توان برای مسائل با تابع هدف غیرخطی استفاده کرد. آن‌ها با استفاده از اطلاعات انحنا کارایی را بهبود می‌بخشند. این بخش با معرفی روش گرادیان مزدوج آغاز می‌شود، روشی که فقط به اطلاعات مشتق مرتبه اول برای برنامه‌ریزی درجه دوم نیاز دارد. معایب روش تندترین کاهش را ندارد و از ذخیره و محاسبه ماتریس هسین نیوتن جلوگیری می‌کند. سپس، روش شبه نیوتن با استفاده از اطلاعات مرتبه دوم شرح داده شده است. اگرچه همگرایی الگوریتم می‌تواند تضمین شود، محاسبات زیادی دارد و به ندرت برای حل مسائل بزرگ یادگیری ماشین استفاده می‌شود. در سال‌های اخیر، با بهبود روش‌های بهینه‌سازی مرتبه بالا، روش‌های بیش تری برای مدیریت داده‌های بزرگ با استفاده از تکنیک‌های تصادفی، ارائه شده است. از این منظر، ما در مورد چندین

روش مرتبه بالا از جمله روش تصادفی شبه نیوتن و انواع آن‌ها بحث می‌کنیم. این الگوریتم‌ها به ما این امکان را می‌دهد تا از روش‌های مرتبه بالا برای پردازش داده‌های بزرگ استفاده کنیم.

۱.۲.۲ روش گرادیان مزدوج

گرادیان مزدوج (CG) [۱۵] یکی از موثرترین روش‌ها برای حل دستگاه معادلات خطی بزرگ است. هم‌چنین می‌تواند برای حل مسائل بهینه‌سازی غیرخطی استفاده شود. روش‌های مرتبه اول ساده هستند اما سرعت همگرایی آهسته‌ای هم دارند و روش‌های مرتبه دوم به منابع زیادی نیاز دارند. بهینه‌سازی گرادیان مزدوج یک الگوریتم میانی است، که فقط می‌تواند از اطلاعات مرتبه اول برای برخی مسائل استفاده کند، اما مانند روش‌های مرتبه بالا سرعت همگرایی را تضمین می‌کند.

در اوایل دهه ۱۹۶۰، یک روش گرادیان مزدوج برای حل یک سیستم خطی پیشنهاد شد، که جایگزینی برای روش حذف گاوسی است. سپس در سال ۱۹۶۴، روش گرادیان مزدوج برای کنترل بهینه‌سازی غیرخطی گسترش یافت. سال‌ها الگوریتم‌های مختلفی بر اساس این روش ارائه شدند که برخی از آن‌ها بسیار مورد استفاده قرار گرفته‌اند. ویژگی‌های اصلی این الگوریتم‌ها این است که سرعت همگرایی بیش‌تری نسبت به تندترین کاهش دارند.

یک سیستم خطی را در نظر بگیرید:

$$A\theta = b, \tag{۲۸.۲}$$

که در آن A یک ماتریس متقارن و معین مثبت $n \times n$ است. ماتریس A و بردار b مشخص هستند و باید مقدار θ را بدست آوریم. مسئله بالا می‌تواند به عنوان یک مسئله بهینه‌سازی در نظر گرفته شود که یک تابع درجه دوم را به حداقل برساند،

$$min_\theta \quad F(\theta) = \frac{1}{2}\theta^T A\theta - b\theta + c, \tag{۲۹.۲}$$

گرادیان $F(\theta)$ را می‌توان با محاسبه ساده بدست آورد: $r(\theta) = \nabla F(\theta) = A\theta - b$.

ماتریس $n \times n$ معین مثبت متقارن A و دو بردار غیر صفر d_i ، d_j A-مزدوج هستند اگر

$$d_i^T A d_j = 0 \tag{۳۰.۲}$$

به مجموعه‌ای از بردارهای غیر صفر $\{d_1, d_2, d_3, , d_n\}$ نسبت به A- مزدوج گفته می‌شود اگر هر دو بردار متمایز در آن، A-مزدوج باشد.

θ_0 یک نقطه شروع است، $\{d_t\}_{t-1}^{n-1}$ مجموعه‌ای از جهت‌های مزدوج است. به طور کلی، می‌توان دنباله به روزرسانی $\{\theta_1, \theta_2, \ldots, \theta_n\}$ را با یک فرمول تکراری تولید کرد:

$$\theta_{t+1} = \theta_t + \eta_t d_t \tag{۳۱.۲}$$

اندازه گام η_t را می‌توان با جستجوی خطی بدست آورد، که به معنای انتخاب η_t برای به حداقل رساندن تابع هدف $f(.)$ در طول $\theta_t + \eta_t d_t$ است. پس از انجام برخی محاسبات، فرمول بروزرسانی η_t به صورت زیر است:

$$\eta_t = \frac{r_t^T r_t}{d_t^T A d_t}. \tag{۳۲.۲}$$

جهت جستجو d_t با ترکیب خطی از مانده منفی و جهت جستجوی قبلی بدست می‌آید:

$$d_t = -r_t + \beta_t d_{t-1} \tag{۳۳.۲}$$

به طوری که r_t می‌تواند با $r_t = r_{t-1} + \eta_{t-1} A d_{t-1}$ به روز رسانی شود. β_t پارامتر به روز رسانی است که می‌تواند با برآورده کردن شرط A-مزدوج بودن d_t و d_{t-1} یعنی $d_t^T A d_{t-1} = 0$، تعیین شود. با ضرب هر دو طرف معادله بالا در $d_{t-1}^T A$، می‌توان β_t با رابطه زیر بدست آورد:

$$\beta_t = \frac{d_{t-1}^T A r_t}{d_{t-1}^T A d_{t-1}^T} \tag{۳۴.۲}$$

نسخه ساده β_t عبارت است از:

$$\beta_t = \frac{r_t^T r_t}{r_{t-1}^T r_{t-1}} \tag{۳۵.۲}$$

روش CG دارای این ویژگی است که فقط با استفاده از بردار قبلی d_{t-1}، یک بردار جدید d_t تولید می‌کند که نیازی به دانستن بردارهای قبلی $d_0, d_1, d_2, \ldots, d_{t-2}$ نیست. الگوریتم گرادیان مزدوج خطی را در الگوریتم ۲ نشان داده‌ایم.

الگوریتم ۲ الگوریتم روش گرادیان مزدوج

Input: A, B, θ

Output: The solution θ^*

$r_0 = A\theta_0 - b$

$d_0 = -r_0, t = 0$

while Unsatisfied convergense condition **do**

$\eta_t = \frac{r_t^T r_t}{d_t^T A d_t}$

$\theta_{t+1} = \theta_t + \eta_t d_t$

$r_{t+1} = r_t + \eta_t A d_t$

$\beta_{t+1} = \frac{r_{t+1}^T r_{t+1}}{r_t^T r_t}$

$d_{t+1} = -r_{t+1} + \beta_{t+1} d_t$

$t = t + 1$

end while

۲.۲.۲ روش‌های شبه نیوتن

گرادیان کاهشی از اطلاعات مرتبه اول استفاده می‌کند، اما همگرایی آن کُند است. بنابراین، ایده طبیعی استفاده از اطلاعات مرتبه دوم، به عنوان مثال، روش نیوتن است. ایده اصلی روش نیوتن استفاده از مشتق مرتبه اول و مشتق مرتبه دوم برای تقریب تابع هدف با یک تابع درجه دوم و سپس حل مینیمم تابع درجه دوم است.

فرمول یک تکرار روش نیوتن در حالت تک متغیره به صورت زیر است:

$$\theta_{t+1} = \theta_t - \frac{f\prime(\theta_t)}{f\prime\prime(\theta_t)} \tag{۳۶.۲}$$

که در آن f تابع هدف است. به طور کلی، فرمول یک تکرار روش نیوتن با ابعاد بالا به صورت زیر است:

$$\theta_{t+1} = \theta_t - \nabla^2 f(\theta_t)^{-1} \nabla f(\theta_t) \tag{۳۷.۲}$$

که در آن $\nabla^2 f$ ماتریس هسین f است. اگر نرخ یادگیری (طول گام) معرفی شود، فرمول تکرار به صورت زیر

نشان داده می‌شود:

$$d_t = -\nabla^2 f(\theta_t)^{-1} \nabla f(\theta_t)$$

$$\theta_{t+1} = \theta_t + \eta_t d_t \tag{۳۸.۲}$$

که در این‌جا d_t جهت روش نیوتن است، η_t طول گام است. از نظر هندسی، روش نیوتن متناسب کردن سطح موضعی موقعیت فعلی با یک سطح درجه دو است، در حالی‌که روش گرادیان کاهشی برای متناسب کردن سطح موضعی فعلی با یک صفحه است.

روش شبه نیوتون: روش نیوتن یک الگوریتم تکراری است که در هر مرحله نیاز به محاسبه معکوس ماتریس هسین تابع هدف دارد و ذخیره‌سازی و محاسبه را بسیار پرهزینه می‌کند. برای غلبه بر ذخیره‌سازی و محاسبه گران قیمت، الگوریتمی تقریبی در نظر گرفته شد که روش شبه نیوتن نامیده می‌شود. ایده اساسی روش شبه نیوتن استفاده از ماتریسی معین مثبت برای تقریب معکوس ماتریس هسین است. بنابراین پیچیدگی عملیات ساده می‌شود. روش شبه نیوتن یکی از موثرترین روش‌ها برای حل مسائل بهینه‌سازی غیرخطی است. علاوه بر این، مشتق مرتبه دوم مستقیماً در روش شبه نیوتون مورد نیاز نیست، بنابراین گاهی اوقات کارآمدتر از روش نیوتن است. در بخش زیر، چندین روش شبه نیوتن را مرور خواهیم کرد که در آن‌ها ماتریس هسین و معکوس آن با روش‌های مختلف تقریب می‌یابند.

شرط شبه نیوتن: ابتدا شرط شبه نیوتن را معرفی می‌کنیم. با فرض این‌که تابع هدف f را می‌توان با یک تابع درجه دوم تقریب زد، می‌توانیم f_θ را با سری تیلور در $\theta = \theta_{t+1}$ بسط دهیم. به عنوان مثال،

$$f(\theta) \approx f(\theta_{t+1} + \nabla f(\theta_{t+1})^T (\theta - \theta_{t+1}) + \frac{1}{2}(\theta - \theta_{t+1})^T \nabla^2 f(\theta_{t+1})(\theta - \theta_{t+1}) \tag{۳۹.۲}$$

سپس می‌توانیم گرادیان دو طرف معادله فوق را محاسبه کرده و بدست آوریم:

$$\nabla f(\theta_t) \approx \nabla f(\theta_{t+1}) + \nabla^2 f(\theta_{t+1})(\theta - \theta_{t+1}) \tag{۴۰.۲}$$

$\theta = \theta_t$ را قرار دهید. داریم:

$$\nabla f(\theta_t) \approx \nabla f(\theta_{t+1}) + \nabla^2 f(\theta_{t+1})(\theta - \theta_{t+1}) \tag{۴۱.۲}$$

برای نشان دادن ماتریس تقریبی ماتریس هسین از B استفاده کنید. روابط $s_t = \theta_{t+1} - \theta_t$ و $u_t = \nabla f(\theta_{t+1})$ را در نظر بگیرید، ماتریس B_{t+1} به صورت زیر می‌شود:۰

$$u_t = B_{t+1} s_t \tag{۴۲.۲}$$

این معادله را شرط شبه نیوتن یا معادله سکانت می‌نامند.

جهت جستجوی روش شبه نیوتن به صورت زیر است:

$$d_t = -B_t^{-1} g_t \tag{۴۳.۲}$$

که در آن g_t گرادیان f و به روز رسانی شبه نیوتن به صورت زیر است:

$$\theta_{t+1} = \theta_t + \eta_t d_t \tag{۴۴.۲}$$

طول گام η_t برای برآوردن شرایط ولف انتخاب شده است، که مجموعه‌ای از نامساوی‌ها برای جستجوی خط $min_{\eta_t} = f(\theta_t + \eta_t d_t)$ است. برخلاف روش نیوتن، روش شبه نیوتن از B_t برای تقریب ماتریس هسین استفاده می‌کند. در ادامه برخی از روش‌های شبه نیوتن را معرفی می‌کنیم، که در آن‌ها از H_t برای بیان معکوس B_t استفاده می‌شود: $H_t = B_t^{-1}$.

روش DFP: در دهه ۱۹۵۰، یک فیزیکدان به نام ویلیام سی دیویدون، روش جدیدی را برای حل مسائل غیرخطی پیشنهاد کرد. سپس فلچر و پاول این روش را توسعه دادند، که به دنبال آن تحقیقات زیادی در اواخر دهه ۱۹۶۰ و اوایل دهه ۱۹۷۰ آغاز شد. DFP[۷] اولین روش شبه نیوتن است که با توجه به حروف اول نام سه مبدع آن نامگذاری شده است. فرمول تصحیح DFP یکی از خلاقانه‌ترین ابداع‌ها در زمینه بهینه‌سازی غیر خطی است که به صورت زیر است:

$$B_{t+1}^{(DFP)} = (I - \frac{u_t s_t^T}{u_t^T s_t}) B_t (I - \frac{s_t u_t^T}{u_t s_t^T}) + \frac{u_t u_t^T}{u_t^T s_t} \tag{۴۵.۲}$$

فرمول به روز رسانی H_{t+1} به صورت زیر است:

$$H_{t+1}^{(DFP)} = H_t - \frac{H_t u_t u_t^T H_t}{u_t^T H_t u_t} + \frac{s_t s_t^T}{h_t^T s_t} \tag{۴۶.۲}$$

روش BFGS : برویدن، فلچر، گلدفارب، شانو روش BFGS [۶] را پیشنهاد کردند، که در آن B_{t+1} به صورت زیر به روز رسانی می‌شود:

$$B_{t+1}^{(BFGS)} = B_t - \frac{B_t s_t s_t^T B_t}{s_t^T B_t s_t} + \frac{u_t u_t^T}{u_t^T s_t} \qquad (۴۷.۲)$$

به روز رسانی مربوط به H_{t+1} به صورت زیر است:

$$H_{t+1}^{(BFGS)} = (I - \frac{s_t u_t^T}{s_t^T u_t}) H_t (I - \frac{u_t s_t^T}{s_t^T u_t}) + \frac{u_t s_t^T}{s_t^T u_t} \qquad (۴۸.۲)$$

الگوریتم شبه نیوتن هنوز نمی‌تواند مسائل بهینه‌سازی در مقیاس بزرگ را حل کند زیرا این روش دنباله ماتریس‌هایی را برای تقریب ماتریس هسین ایجاد می‌کند. ذخیره این ماتریس‌ها به خصوص برای مسائل با ابعاد بالا به مصرف منابع رایانه ای نیاز دارد. با توجه به چالش نگهداری این ماتریس‌ها در رایانه، محدود کردن استفاده از آن حتی به مسائل کوچک و متوسط، غیرممکن است.

روش L-BFGS: روش شبه نیوتن حافظه محدود، به نام L-BFGS [۵]، بهبودی مبتنی بر روش شبه نیوتن است که در برخورد با وضعیت بُعد بالا عملیاتی است. این روش به جای حفظ و محاسبه تقریب $n \times n$ از هسین، فقط چند بردار n بعدی را ذخیره می‌کند. ایده اصلی L-BFGS ذخیره دنباله بردارهایی در محاسبه تقریب H_{t+1}، به جای ذخیره ماتریس کامل H_t است. L-BFGS ادغام بیش‌تری برای فرمول بروزرسانی H_{t+1} ایجاد می‌کند و با فرمول زیر کار می‌کند:

$$H_{t+1} = (I - \frac{s_t u_t^T}{u_t^T s_t}) H_t (I - \frac{u_t s_t^T}{u_t^T s_t}) + \frac{s_t s_t^T}{u_t^T s_t -} = V_t^T H_t V_t + \rho s_t s_t^T \qquad (۴۹.۲)$$

که در آن

$$V_t = I - p u_t s_t^T, \quad \rho_t = \frac{1}{s_t^T u_t} \qquad (۵۰.۲)$$

رابطه بالا به این معنی است که تقریب معکوس هسین H_{t+1} را می‌توان با استفاده از جفت دنباله $\{s_l.u_l\}_{l=t-p+1}^{t}$ بدست آورد. اگر جفت‌های $\{s_l.y_l\}_{l=t-p+1}^{t}$ را بدانیم، H_{t+1} قابل محاسبه است. به عبارت دیگر، LBFGS به جای ذخیره و محاسبه ماتریس کامل H_{t+1} ، فقط آخرین جفت‌های p از $\{s_l, y_l\}$ را محاسبه می‌کند. با توجه به معادله، می‌توان به یک روش بازگشتی دست یافت. هنگامی که آخرین مراحل p ذخیره می‌شود، محاسبه H_{t+1}

می‌تواند به صورت زیر بیان شود:

$$\begin{aligned} H_{t+1} =& (V_t^T V_{t-1}^T \dots V_{t-p+1}^T) H_t^0 (V_{t-p+1}^T V_{t-p+2}^T \dots V_t) \\ &+ \rho_{t-p+1} (V_t^T V_{t-1}^T \dots V_{t-p+2}^T) s_{t-p+1} s_{t-p+2}^T (V_{t-p+3} \dots V_t) \\ &+ \dots \\ &+ \rho_t s_t s_t^T \end{aligned} \tag{۵۱.۲}$$

جهت به روز رسانی $d_t = H_t g_t$ را می‌توان محاسبه کرد، جایی که g_t گرادیان تابع هدف f است. شرح دقیق این روش‌ها در الگوریتم‌های ۳ و ۴ آورده شده است.

الگوریتم ۳ دو حلقه بازگشتی برای $H_t g_t$

Input: $\nabla f_t, u_t, s_t$

Output: $H_{t+1} g_{t+1}$

$g_t = \nabla f_t$

$H_t^0 = \frac{s_t u_t}{||u_t||^2} I$

for l = t-1 to t-p **do**

$\eta_t = \rho_l s_l^T g_{l+1}$

$g_l = g_{l+1} - \eta_l u_l$

end for

$r_{t-p-1} = H_t^0 g_{t-p}$

for l = t-p to t-1 **do**

$\beta_l = \rho_l u_l^T \rho_{l-1}$

$\rho_l = \rho_{l-1} + s_l(\eta_l - \beta_l)$

end for

$H_{t+1} g_{t+1} = \rho$

الگوریتم ٤ BFGS

Input: $\theta_0 \in R^n, \epsilon > 0$

Output: the solution θ^*

$t = 0$

$g_0 = \nabla f_0$

$u_0 = 1$

$s_0 = 1$

while $||g_t|| < \epsilon$ **do**

Choose H_t^0,for example $H_t^0 = \frac{s_t^T u_t}{||u_t||^2} I$

$g_t = \nabla f_t$

$d_t = -H_t g_t$ from Algorithm L-BFGS two-loop recursion for $H_t g_t$

Search a step size η_t through Wolf rule

$\theta_{t+1} = \theta_t + \eta_t d_t$

if $k > 0$ **then**

Discard the vector pairs_{t-p}, y_{t-p} from storage

end if

Compute and save

$s_t = \theta_{t+1} - \theta_t, u_t = g_{t+1} - g_t$

$t = t + 1$

end while

۳.۲.۲ روش تصادفی شبه نیوتن

در بسیاری از مدل‌های یادگیری ماشین در مقیاس بزرگ، استفاده از الگوریتم تقریب تصادفی با هر مرحله به روزرسانی بر اساس یک زیر مجموعه آموزش نسبتاً کوچک، ضروری است. الگوریتم‌های تصادفی اغلب بهترین عملکرد تعمیم را در سیستم‌های یادگیری در مقیاس بزرگ بدست می‌آورند. روش شبه نیوتون فقط از اطلاعات

گرادیان مرتبه اول برای تقریب ماتریس هسین استفاده می‌کند. می‌توان روش شبه نیوتن را با روش تصادفی ترکیب کرد به طوری که روی مسائل در مقیاس بزرگ خوب عمل کند. روش‌های Online-BFGS و online-LBFGS دو نوع از BFGS هستند [۱۹].

مینیمم کردن یک تابع تصادفی محدب

$$min_{\theta \in R} \quad F(\theta) = E\,[\,f(\theta, \xi)] \tag{۵۲.۲}$$

که در آن ξ نشان دهنده یک نمونه (یا مجموعه‌ای از نمونه‌ها) متشکل از یک جفت ورودی-خروجی $(x.y)$ است، در نظر بگیرید. در یادگیری ماشین، x به طور معمول ورودی را نشان می‌دهد و y خروجی هدف است. f معمولاً به شکل زیر است:

$$f(\theta; \xi) = f(\theta; x_i; y_t) = l(h(w; x_i); y_i) \tag{۵۳.۲}$$

که در آن h یک مدل پیش‌بینی و l یک تابع زیان است. تابع $f_i(\theta) = f(\theta; x_i; y_i)$ را تعریف می‌کنیم و از زیان برای تعریف هدف استفاده می‌کنیم:

$$F(\theta) = \frac{1}{N} \sum_{i=1}^{N} f_i(\theta) \tag{۵۴.۲}$$

به طور معمول، اگر مقدار زیادی از داده‌های آموزش برای آموزش مدل‌های یادگیری ماشین استفاده شود، انتخاب بهتر استفاده از گرادیان تصادفی نیمه انبوه است:

$$\nabla F_{S_t}(\theta_t) = \frac{1}{c} \sum_{i \in S_t} \nabla f_i(\theta_t) \tag{۵۵.۲}$$

که در آن زیر مجموعه $S_t \subset \{1, 2, 3, \ldots, N\}$ به طور تصادفی انتخاب شده است. c کاردینالیتی S_t است و $c \ll N$. فرض کنید $S_t^H \subset \{1, 2, 3, \ldots, N\}$ یک زیر مجموعه تصادفی از نمونه‌های آموزش است و تقریب هسین می‌تواند به صورت زیر باشد:

$$\nabla^2 F_{S_t}(\theta_t) = \frac{1}{ch} \sum_{i \in S_t^H} \nabla^2 f_i(\theta_t) \tag{۵۶.۲}$$

که در آن ch کاردینالیتی S_t^H است. با استفاده از گرادیان تصادفی، یک روش مستقیم برای توسعه روش شبه نیوتن تصادفی، تبدیل گرادیان‌های قطعی به گرایان‌های تصادفی در کل تکرارها است، مانند online-BFGS و online-LBFGS. به طور خاص، به دنبال BFGS توضیح داده شده در بخش قبلی، S_t، u_t به صورت زیر در می‌آید:

$$S_t := \theta_{t+1} - \theta_t u_t := \nabla F_{S_t}(\theta_{t+1} - \nabla F_{S_t}(\theta_t) \tag{۵۷.۲}$$

یکی از معایب این روش این است که هر تکرار به دو تقریب گرادیان احتیاج دارد. علاوه بر این، به روزرسانی تقریب‌های معکوس هسین در هر مرحله ممکن است منطقی نباشد. پس روش شبه نیوتن تصادفی (SQN) پیشنهاد شد، که استفاده از ضرب بردار–هسین نمونه‌ها برای به روزرسانی H_t است. در همین حال، یک روش موثر، برای جدا کردن گرادیان تصادفی و محاسبات تخمین انحنا، برای دستیابی به یک تقریب هسین پایدار، شده است:

$$\nabla F(\theta_{t+1}) - \nabla F(\theta_t) \approx \nabla^2 F(\theta_t)(\theta_{t+1} - \theta_t) \tag{۵۸.۲}$$

u_t را می‌توان به صورت زیر بازنویسی کرد:

$$u_t := \nabla^2 F_{S_t^H}(\theta_t) \tag{۵۹.۲}$$

روش دقیق شبه نیوتن تصادفی در الگوریتم ۵ نشان داده شده است.

الگوریتم ۵ شبه نیوتن تصادفی

Input: $\theta_0, V, m.\eta_t$

Output: The solution θ^*

for l =1,2,3,... **do**

$s'_t = H_t g_t$ using to two-loop recutsion 3

$s_t = -\eta_t s'_t$

$\theta_{t+1} = \theta_t + s'_t$

if update pairs, **then**

compute s_t and u_t

Add a new displacement pair $\{s_t, u_t\}$ to V

if $|V| > m$, **then**

Remove the eldest paire from V

end if

end if

end for

٤.٢.٢ روش بهینه‌سازی بدون هسین

ایده اصلی روش Hessian-Free (HF) مشابه روش نیوتن است که از اطلاعات مشتق مرتبه دوم استفاده می‌کند. تفاوت در این است که محاسبه مستقیم ماتریس هسین، در روش بدون هسین ضروری نیست.

یک تقریب درجه دوم موضعی $Q_\theta(d_t)$ از تابع هدف F در اطراف پارامتر θ را در نظر بگیرید:

$$F(\theta_t + d_t) \approx Q_\theta(d_t) = F(\theta_t) + \nabla F(\theta_t)^T d_t + \frac{1}{2} d_t^T B_t d_t \qquad (۶۰.۲)$$

که در آن d_t جهت جستجو است. روش HF از روش گرادیان مزدوج برای محاسبه جواب تقریبی d_t از سیستم خطی زیر استفاده می‌کند:

$$B_t d_t = -\nabla F(\theta_t) \qquad (۶۱.۲)$$

که در آن $B_t = H(\theta_t)$ ماتریس هسین است، اما در عمل B_t اغلب به صورت $B_t = H(\theta_t)+\lambda I, \lambda \geq 0$ تعریف می‌شود. سپس به روزرسانی جدید به صورت زیر بدست می‌آید:

$$\theta_{t+1} = \theta_t + \eta_t d_t \qquad (۶۲.۲)$$

که در آن η_t طول گام است که کاهش کافی در تابع هدف را تضمین می‌کند و معمولاً با جستجوی خطی بدست می‌آید. چارچوب اساسی بهینه‌سازی HF در الگوریتم ۶ نشان داده شده است.

الگوریتم ٦ روش بهینه‌سازی بدون هسین

Input: $\theta_0, \nabla f(\theta_0), \lambda$

Output: The solution θ^*

$t = 0$

repeat

$g_t = \nabla f(\theta_t)$

Compute λ by some methods

$d_t = CG(B_t, -g_t)$

$\theta_{t+1} = \theta_t + \eta_t d_t$

$t = t + 1$

until satisfying convergence condition

۵.۲.۲ روش ناحیه اطمینان

روند به روزرسانی اکثر روش‌های معرفی شده در بالا را می‌توان به صورت $\theta_t + \eta_t d_t$ توصیف کرد. جابه جایی نقطه در جهت t را می‌توان به صورت s_t نوشت. از روش ناحیه اطمینان (TRM) [۱۵] می‌توان برای مسائل بهینه‌سازی d نامقید استفاده کرد که در آن تغییر مکان مستقیماً بدون جهت جستجوی d_t تعیین می‌شود. برای مسئله $min f_\theta(x)$، TRM از بسط مرتبه دوم تیلور برای تقریب تابع هدف $f_\theta(x)$ ، که به صورت $q_t(s)$ نشان داده می‌شود، استفاده می‌کند. جستجو در محدوده ناحیه اطمینان با شعاع ∇_t انجام می‌شود. این مسئله را می‌توان چنین توصیف کرد:

$$\begin{cases} minq_t(s) = f_\theta(x_t) + g_t^T s + \frac{1}{2} s^T B_t s \\ s.t. ||s_t|| \leq \Delta_t \end{cases} \tag{۶۳.۲}$$

که در آن g_t گرادیان تقریبی تابع هدف $f(x)$ در نقطه تکرار فعلی x_t، $g_t \approx \nabla f(x_t)$ است، B_t یک ماتریس متقارن است، که تقریب ماتریس هسین $\nabla^2 f_\theta(x_t)$ است و $\Delta_t > 0$ شعاع ناحیه اطمینان است. قسمت اصلی TRM به روزرسانی Δ_t است. مقدار واقعی کاهش در t-امین تکرار به صورت زیر است:

$$\Delta f_t = f_t - f(x_t + s_t) \tag{۶۴.۲}$$

پیش‌بینی کاهش در t-امین تکرار به صورت زیر است:

$$\Delta q_t = f_t q(s_t) \tag{۶۵.۲}$$

نسبت r_t برای اندازه‌گیری درجه تقریبی هر دو تعریف شده است:

$$r_t = \frac{\Delta f_t}{\Delta q_t} \tag{۶۶.۲}$$

این نشان می‌دهد که وقتی r_t به ۱ نزدیک است، مدل واقع بینانه‌تر از حد انتظار است و پس از آن باید بسط Δ_t را در نظر بگیریم. در همان زمان، مدل کاهش زیادی را پیش‌بینی می‌کند و وقتی r_t نزدیک به ۰ باشد، کاهش کم است و پس از آن باید Δ_t را کاهش دهیم. علاوه بر این، اگر r_t بین ۰ و ۱ باشد، می‌توانیم Δ_t را بدون تغییر در نظر بگیریم. آستانه‌های ۰ و ۱ به طور کلی به عنوان مرزهای چپ و راست r_t تنظیم می‌شوند.

۶.۲.۲ خلاصه

روش‌های بهینه‌سازی مرتبه بالا ذکر شده را از نظر خصوصیات، مزایا و معایب در جدول ۲ خلاصه می‌کنیم.

جدول ۲.۲: خلاصه‌ای از روش‌های بهینه‌سازی مرتبه بالا

روش	ویژگی‌ها	مزایا	معایب
گرادیان مزدوج	یک روش بین روش‌های گرادیان مرتبه اول و مرتبه دوم است. مجموعه‌ای از جهت‌های مزدوج را با استفاده از گرادیان‌ها می‌سازد و در راستای جهت مزدوج، مینیمم تابع هدف را پیدا می‌کند.	روش گرادیان مزدوج فقط گرادیان مرتبه اول را محاسبه می‌کند، اما همگرایی سریع‌تری نسبت به روش کاهشی دارد.	در مقایسه با روش گرادیان مرتبه اول، محاسبه گرادیان مزدوج پیچیده‌تر است.
روش نیوتن	روش نیوتن سریعتر از روش گرادیان کاهشی مرتبه اول، معکوس ماتریس هسین را محاسبه می‌کند.	از گرادیان مرتبه دوم که همگرایی سریع‌تری نسبت به گرادیان مرتبه اول دارد، استفاده می‌کند. روش نیوتن تحت شرایط خاص دارای همگرایی درجه دوم است.	برای محاسبه و ذخیره معکوس ماتریس هسین هر تکرار، به زمان محاسبه و فضای ذخیره‌سازی زیادی نیاز دارد.
شبه نیوتن	روش شبه نیوتن از یک ماتریس تقریبی برای تقریب ماتریس هسین یا معکوس آن استفاده می‌کند. روش‌های معروف شبه نیوتن شامل DFP ، BFGS و LBFGS است.	روش شبه نیوتن نیازی به محاسبه معکوس ماتریس هسین، که باعث کاهش زمان محاسبه می‌شود، ندارد. روش شبه نیوتون می‌تواند به همگرایی فوق خطی برسد.	روش شبه نیوتن به یک فضای ذخیره‌سازی بزرگ نیاز دارد، که برای بهینه‌سازی مسائل در مقیاس بزرگ مناسب نیست.

روش	ویژگی‌ها	مزایا	معایب
شبه نیوتون تصادفی	روش شبه نیوتون تصادفی از تکنیک‌های بهینه‌سازی تصادفی استفاده می‌کند. روش‌های نمایش online- LBFGS و SQN از جمله این روش‌ها است.	روش شبه نیوتون تصادفی می‌تواند برای مسائل بزرگ یادگیری ماشین به کار آید.	در مقایسه با روش گرادیان تصادفی، محاسبات این روش پیچیده‌تر است.
بدون هسین	روش HF با استفاده از گرادیان مزدوج از محاسبه معکوس هسین جلوگیری می‌کند.	روش HF می‌تواند از اطلاعات گرادیان مرتبه دوم استفاده کند اما نیازی به محاسبه مستقیم ماتریس‌های هسین ندارد. بنابراین، برای بهینه‌سازی ابعاد بالا مناسب است.	هزینه محاسبات برای ضرب ماتریس-بردار در روش HF با افزایش داده‌های آموزشی به صورت خطی افزایش می‌یابد. برای مسائل در مقیاس بزرگ، خوب کار نمی‌کند.

۳.۲ بهینه‌سازی بدون مشتق

برای برخی از مسائل بهینه‌سازی، مشتق تابع هدف ممکن است وجود نداشته باشد یا محاسبه آن آسان نباشد. راه‌حل مناسب در این حالت، بهینه‌سازی بدون مشتق است. عمدتاً دو نوع ایده برای بهینه‌سازی بدون مشتق وجود دارد. یکی استفاده از الگوریتم‌های ابتکاری است. انواع مختلفی از روش‌های بهینه‌سازی ابتکاری وجود دارد از جمله الگوریتم‌های ژنتیک، کلونی مورچه‌ها و بهینه‌سازی ازدحام ذرات. این روش‌های ابتکاری معمولاً مقادیر تقریبی بهینه سراسری را ارائه می‌دهند و پشتیبانی نظری ضعیفی دارند. مورد دیگر برازش یک تابع مناسب با توجه به نمونه‌های تابع هدف است. روش مختصات کاهشی، یک الگوریتم بدون مشتقات معمول است، و می‌تواند به راحتی برای مسائل یادگیری ماشین گسترش یابد و استفاده شود. در این بخش، به طور عمده روش مختصات کاهشی را معرفی می‌کنیم.

روش مختصات کاهشی یک الگوریتم بهینه‌سازی بدون مشتق برای توابع چند متغیره است. ایده آن این است که

می‌توان جستجوی یک بعدی را به ترتیب در امتداد هر محور انجام داد و مقادیر به روز شده برای هر بعد را بدست آورد. این روش برای برخی از مسائلی که در آن‌ها تابع زیان مشتق ناپذیر است، مناسب است.
مجموعه‌ای از پایه‌های $e_1.e_2.\ldots.e_D$ را در فضای خطی به عنوان جهت جستجو انتخاب می‌کند و مقدار تابع هدف را در هر جهت به حداقل می‌رساند. برای تابع هدف $L(\Theta)$، که در آن Θ_t قبلاً بدست آمده است، بُعد j-ام از Θ^{t+1} به صورت زیر حل می‌شود:

$$\theta_{t+1} = argmin_{\theta_j \in R} L(\theta_1^{t+1}.\ldots.\theta_{j-1}^{t+1}.\theta_j.\theta_{j+1}^{t}.\ldots.\theta_D^t) \qquad (۶۷.۲)$$

بنابراین، $L(\Theta^{t+1}) \leq L(\Theta^t) \leq \cdots \leq L(\Theta^0)$ برقرار است. همگرایی این روش مشابه روش گرادیان کاهشی است. ترتیب به روزرسانی می‌تواند یک ترتیب دلخواه از e_1 تا e_D در هر تکرار باشد.
تفاوت اصلی بین مختصات کاهشی و گرادیان کاهشی این است که هر جهت به روزرسانی در روش گرادیان کاهشی با گرادیان موقعیت فعلی تعیین می‌شود که ممکن است با هیچ محور مختصاتی موازی نباشد.
در روش مختصات کاهشی، جهت بهینه‌سازی از ابتدا تا انتها ثابت است. نیازی به محاسبه گرادیان تابع هدف نیست. در هر تکرار، به روزرسانی فقط در راستای یک محور انجام می‌شود و بنابراین محاسبه روش مختصات کاهشی حتی برای برخی از مسائل پیچیده، ساده است. برای تسریع در همگرایی می‌توان از یک دستگاه مختصات مناسب استفاده کرد.

٤.٢ پیش شرطی‌سازی در بهینه‌سازی

پیش شرطی‌سازی یک رویکرد بسیار مهم در روش‌های بهینه‌سازی است [۴]. استفاده از پیش شرطی‌سازی مناسب می‌تواند تعداد تکرار الگوریتم‌های بهینه‌سازی را کاهش دهد.
برای بسیاری از روش‌های مهم تکرار شونده، همگرایی تا حد زیادی به خصوصیات طیفی ماتریس ضرایب بستگی دارد. به عنوان مثال، اگر M یک تقریب معکوس ناپذیر از ماتریس ضرایب A باشد، سیستم تبدیل به صورت زیر است:

$$M^{-1}A\theta = M^{-1}b \qquad (۶۸.۲)$$

همان جواب سیستم $A\theta = b$ را دارد. اما ممکن است حل سیستم بالا آسان‌تر باشد و خصوصیات طیفی ماتریس ضرایب $M^{-1}A$ بهتر باشد.

در بیش‌تر سیستم‌های خطی، به عنوان مثال، $A\theta = b$، ماتریس A اغلب پیچیده است و حل سیستم را دشوار می‌کند. بنابراین، برای ساده‌سازی این سیستم به مقداری تبدیلات نیاز است. M را پیش شرط می‌نامند. اگر ماتریس پس از استفاده از پیش شرط وضوح ساختاری داشته و یا اسپارس باشد، برای محاسبه مفید است.

الگوریتم گرادیان مزدوج که قبلاً ذکر شد متداول‌ترین روش بهینه‌سازی با پیش شرطی‌سازی است که سرعت همگرایی را افزایش می‌دهد. شرح این روش در الگوریتم ۷ آمده است.

الگوریتم ۷ پیش شرطی‌سازی روش گرادیان مزدوج

Input: A, θ_t, M, b

Output: the solution θ^*

$f_0 = f(\theta_0)$

$g_0 = \nabla f(\theta_0) = A\theta_0 - b$

y_0 is the solution of $My = g_0$

$d_0 = -g_0$

$t = t$

while $g_t \neq 0$ **do**

$\eta_t = \frac{g_t^T y_t}{d_t^T A d_t}$

$\theta t + 1 = \theta_t + \eta_t d_t$

$g_{t+1} = g_t + \eta_t A d_t$

y_{t+1} =solution of $My = g_t$

$\beta_{t+1} = \frac{g_{t+1}^T y_{t+1}}{g_t^T d_t}$

$d_{t+1} = -y_{t+1} + \beta_{t+1} d_t$

$t = t + 1$

end while

۵.۲ مجموعه ابزارهای عمومی برای بهینه‌سازی

برای روش‌های پایه‌ای بهینه‌سازی که به طور گسترده در یادگیری ماشین استفاده می‌شوند، برنامه هایی وجود دارد که کار را بسیار راحت تر می‌کنند. مجموعه ابزارهای بهینه‌سازی مشترک موجود را جمع آوری کرده و در جدول ۲.۳ به طور خلاصه بیان می‌کنیم.

جدول ۳.۲: ابزارهای موجود

ابزار	زبان	توضیح
CVX	متلب	یک سیستم مدل‌سازی مبتنی بر متلب برای بهینه‌سازی محدب است اما نمی‌تواند از پس مسائل بسیار بزرگ برآید. http://cvxr.com/cvx/download/
CVXPY	پایتون	ک پکیج پایتون است که توسط گروه بهینه‌سازی محدب دانشگاه استنفورد برای حل مشکلات بهینه‌سازی محدب تهیه شده است. http://www.cvxpy.org/
CVXOPT	پایتون	می‌تواند برای مدیریت بهینه‌سازی محدب استفاده شود. توسط مارتین اندرسن، یواخیم دال، و لیون واندنبرگ ساخته شده است. http://cvxopt.org/
APM	پایتون	برای بهینه‌سازی در مقیاس بزرگ مناسب است و می‌تواند مسائل برنامه نویسی خطی، برنامه‌نویسی درجه دوم، برنامه نویسی عدد صحیح، بهینه‌سازی غیرخطی و غیره را حل کند. http://apmonitor.com/wiki/index.php/Main/PythonApp

ابزار	زبان	توضیح
SPAMS	سی پلاس پلاس	یک جعبه ابزار بهینه‌سازی برای حل مسائل مختلف تقریب اسپارس است، که توسط جولین مایرال توسعه و نگهداری می‌شود. رابط‌های موجود شامل Matlab ، R ، Python و ++ C هستند. http://spams-devel.gforge.inria.fr/
minConf	متلب	می‌توان برای بهینه‌سازی توابع چند متغیره قابل تفکیک که محدودیت‌های ساده دارند استفاده کرد. این مجموعه‌ای از توابع متلب است که در آن‌ها روش‌های زیادی برای انتخاب وجود دارد. https://www.cs.ubc.ca/ schmidtm/Software/ minConf.html
tf.train. optimizer	پایتون؛ سی پلاس پلاس؛ CUDA	کلاس بهینه‌سازی مبتدی‌ای است که معمولاً به طور مستقیم فراخوانی نمی‌شود و از اغلب زیر کلاس‌های آن استفاده می‌شود. این ابزار الگوریتم‌های بهینه‌سازی کلاسیک مانند گرادیان کاهشی و AdaGrad را پوشش می‌دهد. https://www.tensorflow.org/apiguides/python / train

فصل ۳

توسیع برنامه‌هایی کاربردی برای زمینه‌های منتخب یادگیری ماشین

بهینه‌سازی یکی از هسته‌های اصلی یادگیری ماشین است. بسیاری از روش‌های بهینه‌سازی در مواجهه با مسائل مختلف یادگیری ماشین و محیط‌های کاربردی بیش‌تر توسعه یافته است. زمینه‌های یادگیری ماشینی که در این بخش انتخاب شده‌اند عمدتاً شامل شبکه‌های عصبی عمیق، یادگیری تقویتی، استنتاج تنوع و زنجیره مارکوف مونت کارلو است.

۱.۳ بهینه‌سازی در شبکه‌های عصبی عمیق

شبکه عصبی عمیق (DNN) در سال‌های اخیر بحث داغی در جامعه یادگیری ماشین است. روش‌های بهینه‌سازی متعددی برای شبکه عصبی عمیق وجود دارد. در این بخش، آن‌ها را از دید روش‌های بهینه‌سازی مرتبه اول و بهینه‌سازی مرتبه بالا معرفی می‌کنیم.

۱.۱.۳ روش گرادیان مرتبه اول در شبکه‌های عصبی عمیق

روش بهینه‌سازی گرادیان تصادفی و انواع آن به طور گسترده‌ای در شبکه عصبی عمیق استفاده شده و عملکرد خوبی را نشان داده‌اند. گرادیان کاهشی تصادفی عامل کم شدن نرخ یادگیری را معرفی می‌کند و AdaGrad تمام گرادیان‌های قبلی را جمع می‌کند تا نرخ یادگیری آن‌ها به طور مداوم کاهش یافته و به صفر برسد. با این حال نرخ یادگیری این دو روش باعث می‌شود که در مراحل بعدی بهینه‌سازی، به روزرسانی کُند شود. AdaDelta، RMSProp، Adam و سایر روش‌ها از میانگین نمایی برای به روزرسانی‌های موثر و ساده‌سازی محاسبه استفاده می‌کنند. این روش‌ها از میانگین نمایی برای کاهش مسائل ناشی از افت سریع نرخ یادگیری استفاده می‌کنند اما نرخ یادگیری فعلی را با اتکا به چند گرادیان محدود می‌کنند. ردی و همکاران از یک مثال ساده بهینه‌سازی محدب استفاده کردند تا نشان دهدند که الگوریتم‌های RMSProp و Adam نمی‌توانند همگرا باشند. تقریباً همه الگوریتم‌هایی که به پنجره‌ای با اندازه ثابت از گرادیان‌های گذشته متکی هستند از این مشکل رنج می‌برند، از جمله AdaDelta و تخمین شتاب تطبیقی تسریع شده (Nadam) [۱].

برای اطمینان از همگرایی بهتر است حافظه بلند مدت گرادیان‌های گذشته به جای میانگین حرکت نمایی گرادیان‌ها را در نظر بگیرید. نسخه جدید روش Adam، به نام AmsGrad، از یک تصحیح ساده برای اطمینان از همگرایی مدل با مزایای اصلی استفاده می‌کند. در مقایسه با روش Adam، AmsGrad تغییرات زیر را در تقریب های مرتبه اول و مرتبه دوم ایجاد می‌کند:

$$\begin{cases} m_t = \beta_{1t} m_{t-1} + (1-\beta_{1t}) g_t \\ V_t = \sqrt{\beta_2 V_{t-1} + (1-\beta_2) g_t^2} \\ \hat{V}_t = max(\hat{V}_{t-1}.V_t) \end{cases} \tag{۱.۳}$$

که در آن β_{1t} ثابت نیست و با گذشت زمان کاهش می‌یابد و β_2 یک نرخ یادگیری ثابت است. تصحیح در حرکت مرتبه دوم V_t انجام می‌شود. $\hat{V}_t$ در تکرار تابع هدف استفاده می‌شود. روش AmsGrad حافظه طولانی مدت گرادیان‌های قبلی را بر اساس روش Adam می‌گیرد و همگرایی را در مرحله بعد تضمین می‌کند. علاوه بر این، تنظیم پارامترهای β_1 ، β_2 همزمان به همگرایی تا حدی کمک می‌کند. به عنوان مثال، β_1 می‌تواند به عنوان $B_{1t} = \frac{\beta_t}{t}, \beta_{1t} \leq \beta_1$ ، برای هر $t \in [T]$ کم شود. β_2 را می‌توان برای هر $t \in [T]$ به عنوان

$\beta_{2t} = 1 - \frac{1}{t}$ قرار داد.

ایده دیگری برای ترکیب گرادیان کاهشی تصادفی و Adam برای حل مسئله غیر همگرا ارائه شد. الگوریتم‌های تطبیقی مانند Adam به سرعت همگرا می‌شوند و برای پردازش داده‌های پراکنده مناسب هستند. گرادیان کاهشی تصادفی می‌تواند به نتایج دقیق‌تری همگرا شود. ترکیب گرادیان کاهشی تصادفی و Adam مزایای هر دو روش را دارد. ابتدا با Adam به سرعت کاهش می‌یابد و سپس برای بهینه‌سازی دقیق بر اساس پارامترها در یک نقطه، به سراغ گرادیان کاهشی تصادفی می‌رود. این استراتژی به عنوان تغییر از Adam به SGD، (SWATS) نامگذاری شده است. دو مشکل اساسی در SWATS وجود دارد. یکی این است که چه زمانی از Adam به گرادیان کاهشی تصادفی تغییر کند، دیگری این است که چگونه نرخ یادگیری را پس از تغییر الگوریتم بهینه‌سازی تنظیم کنید. روش SWATS در زیر با جزئیات شرح داده شده است. جهت حرکت d^{Adam} پارامتر تکرار t از Adam است:

$$d^{Adam} = \frac{\eta^{Adam}}{V_t} m_t \tag{۲.۳}$$

که در آن η^{Adam} نرخ یادگیری Adam است. جهت d^{SGD} پارامتر در تکرار t از گرادیان کاهشی تصادفی است:

$$d_t^{SGD} = \eta^{SGD} g_t \tag{۳.۳}$$

جهت گرادیان کاهشی تصادفی را می‌توان به نرخ یادگیری در امتداد جهت Adam و جهت متعامد آن تجزیه کرد. اگر گرادیان کاهشی تصادفی به پایان برسد اما Adam به دلیل حرکت پس از انتخاب جهت بهینه‌سازی، کار خود را به پایان نرساند، در امتداد مسیر Adam برای روش SWATS بهتر است. همزمان SWATS با حرکت در جهت متعامد مسیر بهینه خود را تنظیم می‌کند.

$$Proj_{Adam} d_t^{SGD} = Adam_t \tag{۴.۳}$$

و راه‌حل بدست آمده

$$\eta_t^{SGD} = \frac{(d_t^{Adam})^T d_t^{Adam}}{(d_t^{Adam})^T g_t} \tag{۵.۳}$$

است که در آن $proj_{Adam}$ به معنای تصویر در جهت Adam است. برای کاهش نویز، می‌توان از یک میانگین برای تخمین نرخ یادگیری استفاده کرد:

$$\lambda_t^{SGD} = \beta_2 \lambda_{t-1}^{SGD} + (1 - \beta_2)\eta_t^{SGD} \tag{۶.۳}$$

$$\lambda_t^{\sim SGD} = \frac{\lambda_t^{SGD}}{1 - \beta_2} \tag{۷.۳}$$

که در آن λ_t^{SGD} اولین لحظه یادگیری نرخ η^{SGD} است و $\lambda_t^{\sim SGD}$ نرخ یادگیری گرادیان کاهشی تصادفی پس از تبدیل است. برای نقطه تغییر، اغلب از رابطه $|\lambda_t^{\sim SGD} - \lambda_t^{SGD}| < \epsilon$ استفاده می‌شود. اگرچه برای انتخاب این معیار، هیچ دلیل ریاضی دقیقی وجود ندارد، اما در انواع برنامه‌ها عملکرد خوبی دارد. برای اثبات ریاضی نقطه تغییر، تحقیقات بیش‌تری می‌تواند انجام شود. اگرچه SWATS مبتنی بر Adam است، اما این روش سوئیچینگ برای سایر روش‌ها مانند AdaGrad و RMSProp نیز قابل استفاده است. این روش نسبت به فراپارامترها حساس نیست و می‌تواند یک جواب بهینه قابل مقایسه با گرادیان کاهشی تصادفی اما با سرعت آموزش بیش‌تر در شبکه‌های عمیق بدست آورد.

اخیراً برخی از محققان در تلاش اند روش‌های تطبیقی را توضیح و بهبود دهند. استراتژی‌های آن‌ها می‌تواند مانند بالا با تکنیک‌های سوئیچینگ ترکیب شود تا عملکرد الگوریتم را افزایش دهد. شبکه‌های عصبی کاملاً متصل نمی‌توانند داده‌های متوالی مانند متن و صدا را پردازش کنند. شبکه عصبی بازگشتی (RNN) نوعی شبکه عصبی است که برای پردازش داده‌های متوالی مناسب‌تر است. به طور کلی استفاده از روش‌های مرتبه اول برای بهینه‌سازی شبکه عصبی بازگشتی موثر نیست، زیرا گرادیان کاهشی تصادفی و روش‌های مختلف آن برای یادگیری طولانی مدت در مسائل دنباله ای دشوار است.

در سال‌های اخیر، یک روش طراحی شده برای پارامتر تصادفی با استفاده از گرادیان کاهشی تصادفی با حرکت بدون اطلاعات انحنا، نتایج خوبی در آموزش شبکه عصبی بازگشتی بدست آورده است. روش‌های بهینه‌سازی مرتبه اول برای آموزش شبکه عصبی بازگشتی توسعه یافته‌اند، اما هنوز با مشکل همگرایی کُند در شبکه عصبی بازگشتی عمیق روبه‌رو هستند. روش‌های بهینه‌سازی مرتبه بالا با استفاده از اطلاعات انحنا می‌توانند همگرایی نزدیک به مقدار بهینه را تسریع کنند و در بهینه‌سازی شبکه عصبی عمیق موثرتر باشند.

۲.۱.۳ روش گرادیان مرتبه بالا در شبکه‌های عصبی عمیق

روش بهینه‌سازی مرتبه اول را که در شبکه عصبی عمیق استفاده می‌شود شرح داده‌ایم. از آن‌جا که اکثر شبکه عصبی عمیق از داده‌های در مقیاس بزرگ استفاده می‌کنند، نسخه‌های مختلفی از روش‌های گرادیان تصادفی توسعه یافته و دارای عملکرد خوبی هستند. برای استفاده کامل از اطلاعات گرادیان، روش مرتبه دوم به تدریج در شبکه عصبی عمیق اعمال می‌شود. در این بخش، به طور عمده روش بدون هسین در شبکه عصبی عمیق را معرفی می‌کنیم.

روش (HF) مدت طولانی است که در زمینه بهینه‌سازی مورد مطالعه قرار گرفته است، اما برای کار با شبکه‌های عصبی مستقیماً مناسب نیست. از آن‌جا که تابع هدف در شبکه عصبی عمیق محدب نیست، ماتریس دقیق هسین ممکن است معین مثبت نباشد. بنابراین، لازم است برخی اصلاحات انجام شود تا روش HF در شبکه‌های عصبی اعمال شود [۱۷].

ماتریس تعمیم یافته گاوس-نیوتن: یک راه‌حل، استفاده از ماتریس تعمیم یافته گاوس-نیوتن (GGN) است که می‌تواند به عنوان تقریب ماتریس هسین در نظر گرفته شود. ماتریس GGN یک ماتریس نیمه معین مثبت است، که از مشکل هسین منفی جلوگیری می‌کند. حداقل دو روش برای ماتریس GGN وجود دارد. هر دوی آن‌ها که $f(\theta)$ را می‌توان به عنوان ترکیبی از دو تابع $f(\theta) = Q(f(\theta))$ بیان کرد که در آن $f(\theta)$ تابع هدف است و Q محدب است. ماتریس G در GGN به شکل زیر است:

$$G = J^T Q^n J \tag{۸.۳}$$

که در این‌جا J ماتریس ژاکوبی F است.

روش‌های میرایی: یکی دیگر از اصلاحات در روش بدون هسین استفاده از روش‌های مختلف میرایی است. به عنوان مثال، میرایی تیخونوف، یکی از معروف‌ترین روش‌های میرایی، با معرفی جریمه درجه دو در مدل درجه دوم اجرا می‌شود. یک جریمه درجه دو $\frac{\lambda}{2} d^T d$ به مدل درجه دوم اضافه می‌شود:

$$Q(\theta) := Q(\theta) + \frac{lambda}{2} d^T d = f(\theta_t) + \nabla f(\theta_t)^T d + \frac{1}{2} d^T B d \tag{۹.۳}$$

که در آن $B = H + \lambda I$ و $\lambda > 0$ مقاومت میرایی را تعیین می‌کند که یک پارامتر مقیاسی است. بنابراین، Bv به صورت $Bv = (H + \lambda I)^v = Hv + \lambda v$ نوشته می‌شود. روش اصلی میرایی تیخونوف در آموزش شبکه عصبی بازگشتی خوب نیست. با توجه به ساختار پیچیده شبکه عصبی بازگشتی، تقریب درجه دوم موضعی در

جهات خاص در فضای پارامتر، حتی در فواصل بسیار کوچک، شاید بسیار نادرست باشد. روش میرایی تیخونوف تنها با افزایش جریمه در همه جهات می‌تواند این را جبران کند. بنابراین، میرایی ساختاری که عملکرد را بسیار بهتر می‌کند، پیشنهاد می‌شود. حال به طور خلاصه روش HF با میرایی ساختاری را معرفی می‌کنیم. در نظر بگیرید $e(x.\theta)$ تابع مقدار بردار θ که می‌تواند در هنگام محاسبه $f(x.\theta)$ به عنوان مقادیر میانی تفسیر شود، که در آن $f(x.\theta)$ تابع هدف است. به عنوان مثال، $e(x.\theta)$ ممکن است حاوی تابع فعال‌سازی برخی از لایه‌های پنهان در شبکه‌عصبی باشد. میرایی ساختاری را می‌توان چنین تعریف کرد

$$R(\theta) = \frac{1}{|s|} \sum_{(x,y)\in s} D(e(x.\theta).e(x.\theta_t)) \qquad (۱۰.۳)$$

که در آن D یک تابع فاصله یا یک تابع زیان است. با در نظر داشتن فاصله بین $e(x.\theta)$، می‌توان از تغییر زیادی در $e(x.\theta)$ جلوگیری کرد و به صورت زیر نوشت:

$$Q_\theta(d)' = Q_\theta(d) + \mu R(d + \theta_t) + \frac{\lambda}{2} d^T d \qquad (۱۱.۳)$$

که در آن μ و λ دو پارامتر هستند که باید به صورت پویا تنظیم می‌شوند. d جهت در تکرار t ام است. برای این‌که روش میرایی مبتنی بر جریمه بهتر کار کند، پارامترهای میرایی را می‌توان به طور مداوم تنظیم کرد. برای تنظیم مستقیم λ از روش اکتشافی به سبک لونبرگ-مارکوارت (Levenberg-Marquardt) استفاده شد. روش اکتشافی لونبرگ-مارکاد به شرح زیر توصیف شده است:

۱) اگر $\gamma < \frac{1}{4}$ آن‌گاه $\lambda \leftarrow \frac{3}{4}\lambda$

۲) اگر $\gamma > \frac{3}{4}$ آن‌گاه $\lambda \leftarrow \frac{2}{3}\lambda$

که در آن γ "نرخ کاهش" و به فرم زیر است:

$$\gamma = \frac{f(\theta_{t-1} + d_t)}{M_{t-1}(d_t)} \qquad (۱۲.۳)$$

۲.۳ بهینه‌سازی در یادگیری تقویتی

یادگیری تقویتی (RL) یک زمینه تحقیقاتی مهم در یادگیری ماشین و هم‌چنین یکی از محبوب‌ترین مباحث است. عامل یادگیری تقویتی عمیق، در یادگیری مهارت‌های پیچیده موفق بوده و از طریق مکانیزم آزمون و خطا با محیط

ارتباط برقرار می‌کند و با به حداکثر رساندن پاداش‌های تجمعی استراتژی‌های بهینه را می‌آموزد [۲۰]. چند مفهوم از یادگیری تقویتی را در ادامه بیان می‌کنیم:

۱) عامل(Agent): انجام اقدامات مختلف با توجه به وضعیت محیط خارجی و تنظیم استراتژی با توجه به پاداش محیط خارجی.

۲) محیط: کلیه موارد خارج از عامل که تحت تأثیر اقدامات عامل قرار خواهند گرفت و با تغییر وضعیت، پاداش عامل را فراهم کند.

۳) حالت s: توصیف محیط.

۴) عمل a: توصیف رفتار عامل.

۵) پاداش $r_t(s_{t-1}.a_{t-1}.s_t$: مقدار بازده بازه زمانی در زمان t.

۶) سیاست$\pi(a|s)$: تابعی که عامل با توجه به وضعیت فعلی s تصمیم به عمل a می‌گیرد.

۷) احتمال تبدیل حالت$p(s'|s.a)$: توزیع احتمالی که محیط در لحظه بعد پس از انتخاب عامل یعنی a بر اساس وضعیت فعلی s، به حالت s' برسد.

۸)$p(s'.r|s.a)$: احتمال تبدیل به حالت s' و بدست آوردن پاداش r، در جایی که عامل در حالت s است و عمل a را انتخاب می‌کند.

بسیاری از مسائل یادگیری تقویتی را می‌توان با فرآیند تصمیم‌گیری مارکوف $< S.A.P.\gamma.r > (MDF)$، توصیف کرد که در آن S فضای حالت، A فضای عملیاتی، P تابع احتمال، r تابع پاداش و γ ضریب تخفیف $0 < \gamma < 1$[۲۱] است. در هر زمان، عامل یک حالت را می‌پذیرد و یک عمل را انتخاب می‌کند. عامل از محیط بازخورد دریافت می‌کند و سپس به حالت بعدی می‌رود. هدف از یادگیری تقویتی، یافتن استراتژی است که به ما امکان می‌دهد حداکثر پاداش کل با تنزیل γ زیر را بدست آوریم. بازده تنزیل با فرمول زیر محاسبه می‌شود:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \qquad (۱۳.۳)$$

از این مرحله، یادگیری تقویتی به دو دسته تقسیم می‌شود. یکی یادگیری تقویتی مبتنی بر مدل که MDP کل مدل را می‌داند (از جمله احتمال P و تابع پاداش r) و دیگری روش بدون مدل است که MDP در آن تعریف نمی‌شود. تابع مقدار رایج، استفاده از تابع مقدار حالت است:

$$V_\pi(s) = E_\pi \left[G_t | S_t = s \right] \tag{۱۴.۳}$$

که بازده مورد انتظار اجرای سیاست π از حالت s است. تابع مقدار حالت-عمل نیز ضروری است که بازده مورد انتظار برای انتخاب عمل a تحت حالت s و سیاست π است:

$$Q_\pi(s, a) = E_\pi \left[G_t | S_t = s, A_t = a \right] \tag{۱۵.۳}$$

تابع مقدار حالت فعلی s را می‌توان با تابع مقدار حالت بعدی s' محاسبه کرد. $V_\pi(s)$ و $Q_\pi(s.a)$ با روابط زیر توصیف می‌شوند:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a \left[r(s, a, s') + \gamma V_\pi(s') \right] \tag{۱۶.۳}$$

$$Q_\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r(s, a, s\prime) + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a') \right] \tag{۱۷.۳}$$

روش‌های یادگیری تقویتی بسیاری براساس تابع ارزش وجود دارد. آن‌ها روش‌های مبتنی بر ارزش نامیده می‌شوند، که نقش مهمی در یادگیری تقویتی دارند. به عنوان مثال، Q-learning و SARSA دو روش محبوب هستند. روش مبتنی بر سیاست، سیاست $\pi_\theta(a|s)$ را بهینه و پارامترها را با θ با گرادیان کاهشی به روزرسانی می‌کند.

۳.۳ بهینه‌سازی در متایادگیری

متایادگیری یک رویکرد پژوهشی دیگر در زمینه یادگیری ماشین است. یادگیری ماشین بدین صورت است که ابتدا مقدار زیادی داده در یک کار خاص بدست آید و سپس از داده‌ها برای آموزش مدل استفاده می‌شود. در یادگیری ماشین، داده‌های آموزشی کافی تضمین دستیابی به عملکرد خوب است. با این حال، انسان‌ها می‌توانند فقط با چند نمونه آموزش، که بسیار کارآمدتر از روش‌های سنتی یادگیری ماشین هستند، کارهای جدید را پردازش کنند. نکته اصلی این است که مغز انسان "چگونگی یادگیری" را آموخته است و می‌تواند از دانش و تجربه گذشته برای وظایف جدید استفاده کامل کند. بنابراین، چگونگی ساختن ماشین‌ها با توانایی یادگیری کارآمد مانند انسان

یک مسئله در یادگیری ماشین است.

هدف از متایادگیری طراحی مدلی است که بتواند با استفاده از کم‌ترین تعداد نمونه، به خوبی آموزش ببیند. به طور کلی، روش‌های متایادگیری را می‌توان در سه نوع زیر خلاصه کرد: روش‌های مبتنی بر متریک، روش‌های مبتنی بر مدل و روش‌های مبتنی بر بهینه‌سازی. در این بخش، روی روش‌های متایادگیری مبتنی بر بهینه‌سازی تمرکز می‌کنیم. در متایادگیری، معمولاً بعضی کارها با نمونه‌های آموزشی کافی و یک کار جدید فقط با چند نمونه آموزش وجود دارد. ایده اصلی را می‌توان به شرح زیر توصیف کرد: در مرحله متا آموزش، یک عمل τ را از مجموعه عمل T، که شامل $(D_\tau^{train}, D_\tau^{test})$ است، انتخاب کنید. برای عمل τ، پارامتر بهینه‌ساز θ را با نمونه‌های آموزشی D_τ^{test} آموزش و به روز سانی کنید، پارامتر متا بهینه‌ساز ϕ را با نمونه‌های آزمایشی D_τ^{train} به روز کنید. روند انتخاب از مدل‌ها و به روز رسانی پارامترها چندین بار تکرار می‌شود. در مرحله متا آزمون، از بهینه‌ساز آموزش‌دیده برای یادگیری یک عمل جدید استفاده می‌شود.

از آن‌جا که هدف از متایادگیری دستیابی به یادگیری سریع است، یک نکته اساسی این است که گرادیان کاهشی را با دقت بیش‌تری در بهینه‌سازی انجام دهید. در بعضی از روش‌های متایادگیری، فرایند بهینه‌سازی خود می‌تواند به عنوان یک مسئله یادگیری برای یادگیری گرادیان پیش‌بینی به جای الگوریتم گرادیان کاهشی باشد. شبکه‌های عصبی با گرادیان اصلی به عنوان ورودی و گرادیان پیش‌بینی به عنوان خروجی اغلب به عنوان یک متا بهینه‌ساز استفاده می‌شود. شبکه عصبی با استفاده از نمونه‌های آموزشی و آزمایشی سایر عمل‌ها آموزش داده می‌شود و در عمل جدید استفاده می‌شود. به روزرسانی پارامتر در روند آموزش به شرح زیر است:

$$\theta_{t+1} = \theta_t + N(g(\theta_t), \phi) \tag{۱۸.۳}$$

که در آن θ_t پارامتر مدل در تکرار t و N متا بهینه‌ساز با پارامتر ϕ است که یاد می‌گیرد چگونه گرادیان را پیش‌بینی کند. پس از آموزش، N متا بهینه‌ساز و پارامتر آن ϕ با توجه به مقدار افت در نمونه‌های آزمایش به روز می‌شوند. این آزمایشات تأیید کرده‌اند که یادگیری بهینه‌سازهای عصبی در مقایسه با پیشرفته‌ترین روش‌های بهینه‌سازی گرادیان تصادفی تطبیقی که در یادگیری عمیق استفاده می‌شود، سودمند است.

فصل ٤

چالش‌ها و مسائل باز

با افزایش پیچیدگی مدل‌های یادگیری ماشین، روش‌های بهینه‌سازی در یادگیری ماشین هنوز هم با چالش‌هایی روبه‌رو هستند. در این بخش، ما در مورد مشکلات و چالش‌های حل نشده برخی از روش‌های بهینه‌سازی در یادگیری ماشین بحث می‌کنیم، که ممکن است پیشنهادها یا ایده‌هایی برای تحقیقات آینده ایجاد کند و کاربرد وسیع‌تری از روش‌های بهینه‌سازی در یادگیری ماشین را ارائه دهد.

١.٤ چالش‌ها در شبکه‌های عصبی عمیق

در بهینه‌سازی شبکه عصبی عمیق هنوز چالش‌های زیادی وجود دارد. یکی از چالش‌ها عدم وجود داده‌های کافی برای آموزش، و دیگری تابع هدف غیر محدب در شبکه عصبی عمیق است. به طور کلی، مشکل یادگیری عمیق بر اساس مجموعه داده‌های بزرگ و مدل‌های پیچیده است. برای دستیابی به تأثیرات خوب آموزشی، به تعداد زیادی از نمونه‌های آموزشی نیاز دارد. اما در برخی از زمینه‌های خاص، یافتن مقدار کافی از داده‌های آموزش دشوار است. اگر داده کافی برای تخمین پارامترهای موجود در شبکه‌های عصبی نداشته باشیم، ممکن است منجر به واریانس بالا و بیش برازش شود.

برخی از تکنیک‌ها در شبکه‌های عصبی وجود دارد که می‌تواند برای کاهش واریانس استفاده شود. افزودن تنظیم کننده L_2 به تابع هدف یک روش طبیعی برای کاهش پیچیدگی مدل است. اخیراً، یک روش معمول Dropout است. در فرآیند آموزش، هر نورون مجاز به توقف کار با احتمال p است، از زیرشبکه‌های M

می‌توان با چند بار قرار دادن و برگرداندن، نمونه برداری کرد. امید ریاضی در لایه خروجی به صورت زیر محاسبه می‌شود:

$$O = E_M\left[f(x;\theta,M)\right] = \sum_{i=1}^{M} p(M_i f(x;\theta,M_i)) \tag{۱.۴}$$

که در آن $p(M_i)$ احتمال زیر شبکه است. Dropout می‌تواند از بیش برازش جلوگیری کرده و توانایی تعمیم شبکه را بهبود بخشد، اما عیب آن افزایش زمان آموزش است.

نه تنها بیش برازش، بلکه برخی از جزئیات آموزش نیز به دلیل پیچیدگی شبکه عصبی عمیق بر عملکرد مدل تأثیر می‌گذارد. انتخاب نامناسب نرخ یادگیری و تعداد تکرارها در گرادیان کاهشی تصادفی باعث عدم همگرایی مدل می‌شود، که این باعث می‌شود دقت مدل بسیار تغییر کند. علاوه بر این، استفاده از یک جعبه سیاه نامناسب در ساخت شبکه عصبی ممکن است منجر به عدم امکان ادامه آموزش شود. لذا طراحی یک مدل شبکه عصبی مناسب از اهمیت ویژه‌ای برخوردار است. این تأثیرات در صورت ناکافی بودن داده‌ها بیش‌تر است.

۲.٤ مشکلات مدل‌های دنباله‌ای با داده‌های مقیاس بزرگ

هنگام کار با سری‌های زمانی در مقیاس بزرگ، راه‌حل‌های معمول استفاده از بهینه‌سازی تصادفی، پردازش داده‌ها یا استفاده از محاسبات توزیع شده برای بهبود کارایی محاسبات است. برای یک مدل ترتیبی، تقسیم‌بندی دنباله‌ها می‌تواند بر وابستگی بین داده‌ها در شاخص‌های زمان تأثیر بگذارد. اگر طول دنباله مضربی از اندازه زیرمجموعه انتخابی نباشد، عملیات کلی افزودن مواردی است که از داده‌های قبلی به آخرین دنباله نمونه برداری شده‌اند. این عملیات وابستگی نادرستی را در مدل آموزشی ایجاد می‌کند. بنابراین، تجزیه و تحلیل تفاوت بین پاسخ تقریبی بدست آمده و پاسخ دقیق، قابل بررسی است. به ویژه، در شبکه عصبی بازگشتی، مشکل از بین رفتن گرادیان و گرادیان افزایشی نیز مستعد بروز است. تاکنون، این مشکل به طور کلی با حالت‌های خاص تعامل LSTM و GRU یا برش گرادیان حل شده است. راه‌حل‌های مناسب بهتر برای مقابله با مشکلات شبکه عصبی بازگشتی هنوز جای بررسی دارد.

۳.٤ بهینه‌سازی تصادفی در گرادیان مزدوج

روش‌های تصادفی هنگام برخورد با داده‌های در مقیاس بزرگ، خصوصاً برای بهینه‌سازی مرتبه اول، توانایی بالایی از خود نشان می‌دهند. محققان نیز این ایده تصادفی را با روش‌های بهینه‌سازی مرتبه دوم ترکیب کردند و به نتایج خوبی دست یافتند. روش گرادیان مزدوج یک الگوریتم جذاب است که از مزایای هر دو روش بهینه‌سازی مرتبه اول و دوم برخوردار است. فرم استاندارد گرادیان مزدوج برای تقریب تصادفی مناسب نیست. از طریق استفاده از ضرب گرادیان هسین، روش تصادفی نیز با گرادیان مزدوج ترکیب می‌شود، که در آن برخی نتایج عددی اعتبار الگوریتم را نشان می‌دهند. نسخه دیگر روش گرادیان مزدوج تصادفی از تکنیک کاهش واریانس استفاده می‌کند و فقط با چند تکرار به سرعت همگرا می‌شود و در طی مراحل اجرا به فضای ذخیره‌سازی کم‌تری نیاز دارد. نسخه تصادفی گرادیان مزدوج یک روش بهینه‌سازی با پتانسیل بالا است و ارزش مطالعه و بررسی را دارد.

فصل ۵

سخن پایانی

بهینه‌سازی یکی از مولفه‌های اصلی یادگیری ماشین است. ماهیت اکثر الگوریتم‌های یادگیری ماشین بر اساس مدل‌های بهینه‌سازی و یادگیری پارامترهای تابع هدف از داده‌های در دسترس، است. هنگام کار با داده‌های بزرگ، اثربخشی و کارایی الگوریتم‌های بهینه‌سازی به طور چشم گیری بر عملکرد و کاربرد مدل‌های یادگیری ماشین تاثیر می‌گذارد. برنامه‌ریزی ریاضی اساس بسیاری از مدل‌های یادگیری ماشین را تشکیل می‌دهد که در آن آموزش مدل‌ها یک مساله بهینه‌سازی در مقیاس بزرگ است. این پژوهش با شرح برخی روش‌های بهینه‌سازی و مقایسه آن‌ها با هم دید خوبی برای انتخاب بهترین روش ایجاد می‌کند. مدل‌هایی از یادگیری ماشین و الگوهای جدید در حال ظهور در آموزش ماشین و یادگیری تجربی را مورد بررسی قرار دادیم. روش‌های مهم بهینه‌سازی ریاضی برای بیان این مدل‌های یادگیری ماشین را مرور کردیم. با بهره گیری از فرمولاسیون بهینه‌سازی در مقیاس بزرگ، مدل‌های یادگیری ماشین را به صورت مسائل برنامه‌ریزی ریاضی در نظر گرفته و با روش‌های بهینه‌سازی راه حل‌ها را بررسی کردیم. سپس کاربرد روش‌های بهینه‌سازی در سناریوهای مختلف یادگیری ماشین و رویکردهای بهبود عملکرد آن‌ها را شرح دادیم. امروزه یادگیری ماشین و یادگیری عمیق بسیار محبوب اند و به نوعی در بسیاری از مسایل کاربردی، حیاتی محسوب می‌شوند. از آنجا که یادگیری ماشین در عمق مسائل بهینه‌سازی قرار گرفته است، با بهینه‌سازی پیشرفته‌تر می‌توان توسعه خوبی در آن ایجاد کرد. این پژوهش، با دید کلی و مقایسه‌ای که بین روش‌های بهینه‌سازی دارد، برای کار با مدل‌های یادگیری مفید بوده و درک بالاتری از مسائل فراهم می‌آورد.

کتاب‌نامه

[1] Bai, Jiyang, Ren, Yuxiang, and Zhang, Jiawei. Deam: Adaptive momentum with discriminative weight for stochastic optimization. *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 37–41, 2020.

[2] Beck, Amir. *Introduction to Nonlinear Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[3] Beck, Amir. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[4] Benzi, M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182:418–477, 2002.

[5] Berahas, Albert S, Nocedal, Jorge, and Takac, Martin. A multi-batch l-bfgs method for machine learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[6] Broyden, C. G. The Convergence of a Class of Double-rank Minimization Algorithms: 2. The New Algorithm. *IMA Journal of Applied Mathematics*, 6(3):222–231, 09 1970.

[7] Davidon, William C. Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17, 1991.

[8] Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[9] Dvurechensky, P., Staudigl, Mathias, and Shtern, Shimrit. First-order methods for convex optimization. *ArXiv*, abs/2101.00935, 2021.

[10] Gambella, Claudio, Ghaddar, Bissan, and Naoum-Sawaya, Joe. Optimization for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, May 2021.

[11] Hospedales, Timothy M, Antoniou, Antreas, Micaelli, Paul, and Storkey, Amos J. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[12] Jolliffe, Ian. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[13] Keskar, N. and Socher, R. Improving generalization performance by switching from adam to sgd. *ArXiv*, abs/1712.07628, 2017.

[14] Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *ArXiv*, abs/1412.6980, 2015.

[15] Bazaraa, Mokhtar S., Sherali, Hanif D., and Shetty, C. M. *Nonlinear Programming - Theory and Algorithms, Third Edition*. Wiley, 2005.

[16] Luenberger, David and Ye, Yinyu. *Linear and Nonlinear Programming*. Springer US, 2008.

[17] Martens, James and Sutskever, Ilya. *Training Deep and Recurrent Networks with Hessian-Free Optimization*, pages 479–535. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[18] Ruder, Sebastian. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016.

[19] Schraudolph, Nicol N., Yu, Jin, and Günter, Simon. A stochastic quasi-newton method for online convex optimization. In Meila, Marina and Shen, Xiaotong, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 436–443, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

[20] Sun, Shiliang, Cao, Zehui, Zhu, Han, and Zhao, Jing. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics*, 50:3668–3681, 2020.

[21] Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[22] Zeiler, Matthew D. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

Abstract

Machine learning is widely used in various fields and developing rapidly. Optimization methods play a key role in machine learning algorithms. When working with big data, the complexity of the model increases, and as a result, optimization methods face more challenges. Many works have been done to improve optimization methods in machine learning. In this research, we first address some optimization problems in machine learning. Then, we investigate some optimization methods and explain their applications in different areas of machine learning. Finally, we address some of the challenges and open problems.

University of Tehran

School of Mathematics, Statistics, and Computer Science

An overview of optimization problems for machine learning

Shakiba Rahnama

Supervisor: Dr. Majid Soleimani-damaneh

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc. in Applied Mathematics

July 2021