دانشگاه صنعتی شریف
دانشکده ریاضی و علوم کامپیوتر

گزارش پروژه درس بهینه‌سازی در علوم داده

روش حداقل مربعات با تنظیم خودکار

استاد
دکتر مجتبی تفاق

اعضای تیم
مرضیه عبدالحمدی (۹۹۲۰۴۸۰۲)
مهدی اکرمی (۹۹۲۰۴۷۳۸)

تابستان ۱۴۰۱

فهرست

مقدمه

روش حداقل مربعات بیش از ۲۰۰ سال پیش توسط لژاندر و گاوس معرفی شد. این روش به عنوان یکی از پرکاربردترین روش‌های محاسباتی در زمینه‌های مختلف شامل یادگیری ماشین و آمار، پردازش سیگنال، کنترل، رباتیک و مالی می‌باشد. کاربرد این روش به این دلیل است که جواب تحلیلی ساده‌ای دارد و به راحتی قابل فهم است و همچنین الگوریتم‌های پایدار و کارآمدی برای محاسبه جواب‌های این مسئله توسعه یافته است.

برای در نظر گرفتن اختلاف بین هدف مسئله کمترین مربعات و هدف اصلی مسئله ما، یک روش رایج این است که مسئله کمترین مربعات را طوری تنظیم کنیم که جواب مسئله کمترین مربعات منجر به جواب خوبی برحسب هدف اصلی ما شود. ایده‌ها در این جا شامل اصلاح کردن داده‌ها، اضافه کردن جملات به تابع هزینه(منظم سازی) یا تغییر ابرپارامترها یا وزن‌ها در مسئله حداقل مربعات است.

هدف اصلی این است که فرایند تنظیم حداقل مربعات برای گستره‌ای از کاربردهای برازش داده‌ها بصورت خودکار باشد. مسئله حداقل مربعات توسط ابرپارامترها، پارامترلیزه می‌شود و سپس بصورت خودکار با استفاده از بهینه سازی بر پایه‌ی روش‌های گرادیان‌گونه تنظیم می‌شود تا بهترین عملکرد(یا حداقل عملکرد بهتری) را بدست آوریم. این فرایند به ما اجازه می‌دهد که بصورت خودکار، فضای ابرپارامترها را جستجو کنیم که این مزیت را دارد که پارامترهای بدست آمده، بهتر از پارامترهایی که بصورت دستی تنظیم شده است، باشد و همچنین کمک می‌کند که پارامترهای مناسب را خیلی سریع‌تر از روش‌های تظنیم دستی بدست آوریم. این روش به روش حداقل مربعات با تنظیم خودکار شناخته شده است.

به عنوان یک تست کیس خاص، الگوریتم مطرح شده را بر روی داده‌های MNIST با استفاده از روش ساختن classifier ها به همراه هایپر-پارامتر λ در مساله رگرسیون ریج پیاده‌سازی می‌کنیم.

مسئله حداقل مربعات تنظیم شونده‌ی خودکار

مسئله‌ی حداقل مربعات ماتریسی که وابسته به بردار هایپرپارامترهای $\omega \in \Omega \subseteq \mathcal{R}^p$ می‌باشد، بصورت زیر است:

$$minimize \ \|A(\omega)\theta - B(\omega)\|_F^2,$$

که در اینجا، $\theta \in \mathcal{R}^{n\times m}$ متغیر بهینه‌سازی حداقل مربعات یا ماتریس پارامتر می‌باشد و $A: \Omega \to \mathcal{R}^{k\times n}$ و $B: \Omega \to \mathcal{R}^{k\times m}$ هستند. تعبیر ساده‌ای از رابطه فوق، این است که ماتریس‌های A, B، بردار هایپرپارامتر ω را به داده‌های مسئله حداقل مربعات می‌نگارند. $\|\ .\ \|_F$ در اینجا همان نرم فروبنیوس یعنی جذر جمع مربعات درایه‌های یک ماتریس است. فرض بر این است که ستون‌های $A(\omega)$ مستقل خطی هستند و علاوه بر این، $A(\omega)$ یک ماتریس بلند (tall) می‌باشد ($k \geq n$). تحت این مفروضات، جواب حداقل مربعات، یکتا و بصورت زیر تعیین می‌شود:

$$\theta^{ls}(\omega) = A(\omega)^{\dagger}B(\omega) = \left(A(\omega)^T A(\omega)\right)^{-1} A(\omega)^T B(\omega)$$

که $A(\omega)^{\dagger}$ شبه وارون (Moore-Penrose) می‌باشد. $\theta^{ls}(\omega)$ همان نگاشتی است که بردار هایپرپارامترها را به $\theta^{ls}(\omega) \in \mathcal{R}^{n\times m}$ می‌نگارد.

در خیلی از کاربردها، بیشتر از یک تابع هدف موجود است که این چنین مسئله‌ها بصورت زیر بازنویسی می‌شود:

$$minimize \ \lambda_1\|A_1(\omega)\theta - B_1(\omega)\|_F^2 + \cdots + \lambda_r\|A_r(\omega)\theta - B_r(\omega)\|_F^2$$

که در عبارت فوق، λ_iها، وزن‌های نسبت بوده و مسئله به فرم استاندارد زیر قابل بیان است:

$$A(\omega) = \begin{bmatrix} \sqrt{\lambda_1}A_1 \\ \vdots \\ \sqrt{\lambda_r}A_r \end{bmatrix}, \qquad B(\omega) = \begin{bmatrix} \sqrt{\lambda_1}B_1 \\ \vdots \\ \sqrt{\lambda_r}B_r \end{bmatrix},$$

$$minimize \ \ \|A(\omega)\theta - B(\omega)\|_F^2$$

حل مسئله حداقل مربعات

برای ω داده شده، راه‌های زیادی برای حل مسئله موجود است که از جمله می‌توان به تجزیه QR ماتریس چگال یا تنک (sparse) و روش‌های تکراری CG (گرادیان مزدوج) و LSQR اشاره کرد که کتابخانه‌های زیادی برای حل این مسئله با استفاده از چند CPU یا GPU توسعه یافته است؛ به این صورت که مسئله را باتوجه به ستون‌ها (θ) جدا کرده و با ماتریس‌های A, B بصورت موازی حل میکنند.

فرض کنید A, B بصورت ماتریس‌های چگال ذخیره شده باشند. یک راه مناسب برای پیاده سازی توسط GPU استفاده از فرم گرام $G = A^T A$ در کنار $H = A^T B$ است که درمجموع kn^2، knm فلاپ نیاز دارد. توجه کنید که این ضرب‌ها از عملگرهای سطح۳ BLAS هستند که بصورت کارائی محاسبه میشود. برای محاسبه θ^{ls} می‌توان از طریق تجزیه چولسکی $G = LL^T$ با هزینه n^3 فلاپ و حل معادلات مثلثی $LY = H$ و $L^T\theta^{ls} = Y$ با هزینه n^3 و n^2m فلاپ مسئله را حل کرد که با توجه به $k > n$ هزینه محاسباتی کل از مرتبه $kn(n+m)$ میشود.

اگر $A(\omega)$و $B(\omega)$ ماتریس‌های تنک و بزرگ باشند از نمایش عملگری استفاده میشود به این صورت که $A(\omega)u$ برای هر $u \in R^n$ و $A^T(\omega)v$ برای هر $v \in R^k$قابل محاسبه است (نمایش غیرماتریسی). روش‌های CGو LSQR برای حل مسئله برای هر ستون θ بصورت موازی بکار برده میشود. پیچیدگی مسئله، متغیر نسبت به داده‌ها و ابعاد و میزان تنک بودن و فرض‌های مسئله و میزان دقت مورد نیاز است.

حال به مسئله اصلی باز میگردیم. هدف یافتن ابرپارامتری است که مسئله زیر را بهینه می‌کند:

$$minimize\ F(\omega) = \psi\Big(\theta^{ls}(\omega)\Big) + r(\omega)$$

که $\omega\epsilon\Omega$ و $F: \Omega \to R \cup \{+\infty\}$ است و $\psi: R^{n\times m} \to R$ تابع هدف اصلی و $r: \Omega \to R \cup \{+\infty\}$ تابع منظم ساز بر حسب ω است. وقتی $\omega \notin \Omega$ فرض میکنیم $r(\omega) = \infty$ است.

مسئله توسط A، B، ψ و r مشخص میشود. توجه شود که $r(\omega)$ ممکن است تعدادی پارامتر داشته باشد که از طریق اثرگذاری بر روی هایپرپارامترهای مسئله بر روی پارامتر θ تاثیر بگذارند که به این پارامترها، هایپر-هایپر پارامتر گویند.

این مسئله را میتوان بصورت مسئله قیددار بصورت زیر بیان کرد که در آن θ و ω که مستقل هستند توسط قید زیر به هم مرتبط میشوند:

$$\begin{aligned} minimize \quad & \psi(\theta) + r(\omega) \\ s.t. \quad & A^T(\omega)A(\omega)\theta = A^T(\omega)B(\omega) \end{aligned}$$

حل مسئله تنظیم حداقل مربعات

این مسئله در حالت کلی غیرمحدب بوده و حل دقیق آن سخت یا حتی غیرممکن میشود (یک استثنا مهم حالتی است که ω اسکالر و Ω یک بازه باشد که در آن میتوان $F(\omega)$ روی نقاط مشبک Ω را راحت محاسبه کرد).

با توجه به آنچه گفته شد ناچاریم به روش ابتکاری (heuristic) یا بهینه سازی موضعی متوسل شویم. فرض را بر این میگذاریم که A و B نسبت به ω مشتق پذیر و درنتیجه $\theta^{ls}(\omega)$ نیز مشتق پذیر است. فرض کنید که ψ مشتق پذیر بوده یعنی جمله اول تابع F مشتق پذیر است. در مورد تابع $r(\omega)$ چنین فرضی نمیگذاریم. روشی که برای حل مسئله و بدست آوردن ωاستفاده میکنیم روش proximal gradient است که روشی تکراری بصورت زیر است:

$$\omega^{k+1} = prox_{t^k r}(\omega^k - t^k \nabla_\omega \psi\left(\theta^{ls}(\omega^k)\right)$$

که در آن k شماره تکرار و عملگر proximal نیز بصورت زیر است:

$$prox_{tr}(v) = \underset{\omega}{\operatorname{argmin}}(r(\omega) + \frac{1}{2t}\|\omega - v\|_2^2)$$

در این روش فرض بر این است که $argmin$ وجود دارد. وقتی که یکتا نباشد هر مینیمم کننده ای را میتوان انتخاب کرد فقط نیازمند به این است که tr به راحتی قابل محاسبه باشد.

وقتی که $r = 0$ و $\Omega = R^p$ باشد روش proximal gradient همان روش گرادیان معمولی است و حالت دیگر وقتی که $\Omega \subset R^p$و $r(\omega) = 0\ \forall \omega \in R$ آنگاه عملگر proximal برای tr همان تصویر روی Ω و روش proximal gradient همان روش تصویر گرادیان میشود. برای تضمین طول قدم راه‌های متفاوتی وجود دارد اما روشی که توسط Lall and Boyd (2017) ارائه شده، در اینجا بکار میرود.

این روش با قدم اولیه t^1 شروع شده و اگر تابع هدف اصلی کم شده یا تغییر نکند ($F(\omega^{k+1}) \leq F(\omega^k)$) در اینصورت، مقدار طول قدم را زیادتر و ω جدید را میپذیریم. اگر $F(\omega^{k+1}) > F(\omega^k$) طول قدم را کاهش داده و $\omega^{k+1} = \omega^k$. معمولا برای افزایش طول قدم، $t^{k+1} = 1.2t^k$ و برای کاهش $t^{k+1} = \frac{1}{2}t^k$ استفاده میکنند. الگوریتم بصورت پیش فرض برای تعداد n_{iter} بار اجرا می شود.

اگر $F\left(\omega^{k+1}\right) \leq F\left(\omega^k\right)$ باشد، شرط توقف در مرحله $k+1$بصورت

$$\left\| \frac{\omega^k - \omega^{k+1}}{t^k} + \left(g^{k+1} - g^k\right) \right\|_2 \leq \varepsilon$$

می‌شود. که $g^k = \nabla_{\omega^k} \psi\left(\theta^{ls}\left(\omega^k\right)\right)$ و $\varepsilon > 0$ عدد کوچک دلخواهی است.

(اگر $r = 0$ و $\Omega = R^p$روش پراکسیمال گرادیان همان روش گرادیان معمولی می‌شود.)

الگوریتم کامل به صورت زیر است. توجه کنید که ω_1 ، t^1، n_{iter} مفروضات اولیه الگوریتم است و داده شده است:

برای $k=1$ ، ... ، n_{iter} مراحل زیر را انجام بده.

۱. $\theta^{ls}\left(\omega^k\right)=\left(A^T\left(\omega^k\right)A\left(\omega^k\right)\right)^{-1}A^T\left(\omega^k\right)B\left(\omega^k\right)$ را محاسبه کن.

۲. $g^k=\nabla_\omega\psi\left(\theta^{ls}\left(\omega^k\right)\right)$ را محاسبه کن.

۳. $\omega^{k+\frac{1}{2}}=\omega^k-t^k g^k$ قرار بده.

۴. $\omega^{tent}=prox_{t^k r}\left(\omega^{k+\frac{1}{2}}\right)$ محاسبه کن.

۵. اگر $F\left(\omega^{tent}\right)\le F\left(\omega^k\right)$ قرار بده. اگر $\omega^{k+1}=\omega^{tent},\quad t^{k+1}=1.2t^k$ $\left\|\frac{\omega^k-\omega^{k+1}}{t^k}+\left(g^{k+1}-g^k\right)\right\|_2\le\varepsilon$ از حلقه خارج شو.

۶. درغیر اینصورت: $\omega^{k+1}=\omega^k,\quad t^{k+1}=\frac{t^k}{2}$

اگر r و $\psi\left(\theta^{ls}\right)$ محدب باشند روش پراکسیمال گرادیان به مینیمم سراسری همگرا خواهد شد و اگر چنین شرط‌هایی برقرار نباشد، تحت فرض‌های خاصی روش به یک نقطه پایا همگرا می شود.

محاسبه گرادیان

برای محاسبه گرادیان $g=\nabla_\omega\psi\left(\theta^{ls}\left(\omega\right)\right)$ می‌توان از قاعده مشتق زنجیری استفاده کرد. فرض کنید که $\theta=\theta^{ls}\left(\omega\right)$ محاسبه شده است، در مرحله بعد باید $\nabla_\omega\psi\left(\theta\right)$ محاسبه و سپس $C=\left(A^TA\right)^{-1}\nabla_\omega\psi\left(\theta\right)\in R^{n\times m}$ تشکیل گردد.

اگر ماتریس A بصورت چگال ذخیره شده باشد، با استفاده از تجزیه چولسکی ماتریس گرام $G=A^TA=L^TL$ می توان ماتریس C را محاسبه نمود. اما اگر A بصورت عملگر ذخیره شده باشد برای هرستون C بصورت موازی می‌توان و با استفاده از روش های تکراری می‌توان C را محاسبه نمود.

مشتق نسبت به A به صورت زیر محاسبه می شود. قرار دهید $\theta=\phi\left(A,B\right)=\left(A^TA\right)^{-1}A^TB$

$$D_A\phi\left(A,B\right)\Delta A=\phi\left(A+\Delta A,B\right)-\phi\left(A,B\right)$$

$$\begin{aligned}\phi\left(A+\Delta A,B\right)&=\left(\left(A+\Delta A\right)\left(A+\Delta A\right)\right)^{-1}\left(A^TB+\Delta A^TB\right)\\&\approx\left(A^TA\right)^{-1}\left(I-\Delta A^TA\left(A^TA\right)^{-1}-A^T\Delta A\left(A^TA\right)\right)\left(A^TB+\Delta A^TB\right)\\&\approx\phi\left(A,B\right)+\left(A^TA\right)^{-1}\Delta A^TB-\left(A^TA\right)^{-1}\left(\Delta A^TA+A^T\Delta A\right)\theta\end{aligned}$$

برای ساده سازی از رابطه تخمینی

$$\left(X+Y\right)^{-1}\approx\left(1+X^{-1}Y\right)^{-1}X^{-1}\approx\left(1-X^{-1}Y\right)X^{-1}\approx X^{-1}-X^{-1}YX^{-1}$$

و رابطه $\theta=\left(A^TA\right)^{-1}A^TB$ استفاده شده است، بنابراین داریم.

$$D_A\phi\left(A,B\right)\Delta A=\left(A^TA\right)^{-1}\Delta A^TB-\left(A^TA\right)^{-1}\left(\Delta A^TA+A^T\Delta A\right)\theta$$

اکنون قرار دهید $C=\left(A^TA\right)^{-1}\nabla_\omega\psi\left(\theta\right)$ با توجه به تعریف گرادیان برای فضای ماتریس ها داریم $f=\psi\circ\phi,$

$$D_Af\left(A,B\right)\left(\Delta A\right)=\left\langle\nabla_Af,\Delta A\right\rangle=tr\left(\nabla_Af\Delta A\right)$$

با استفاده از خاصیت زنجیری مشتق و خاصیت چرخشی رد حاصلضرب ماتریس‌ها داریم

$$\begin{aligned}&D_Af\left(A,B\right)\left(\Delta A\right)=D_{\theta^{ls}}\psi\circ D_A\left(\phi\left(A,B\right)\Delta A\right)\\&=tr\left(\nabla_\omega\psi^T\left(\left(A^TA\right)^{-1}\Delta A^TB-\left(A^TA\right)^{-1}\left(\Delta A^TA+A^T\Delta A\right)\theta\right)\right)\\&=tr\left(\left(BC^T-A\theta C^T-AC\theta^T\right)^T\Delta A\right)\end{aligned}$$

پس داریم

$$\nabla_A f = \left(B - A\theta\right)C^T - AC\theta^T$$

مشابها داریم

$$\phi\left(A, B+\Delta B\right) = \left(A^T A\right)^{-1} A^T \left(B+\Delta B\right)$$
$$= \left(A^T A\right)^{-1} A^T B + \left(A^T A\right)^{-1} A^T \Delta B$$

درنتیجه

$$D_B\phi\left(A, B\right)\Delta B = \left(A^T A\right)^{-1} A^T \Delta B$$

با استفاده از خاصیت زنجیری مشتق و خاصیت چرخشی رد حاصلضرب ماتریس‌ها و تعریف گرادیان داریم.

$$D_B\phi\left(A, B\right)\Delta B = D_{\theta^{ls}}\psi \circ D_B\left(\phi\left(A, B\right)\Delta B\right)$$
$$= tr\left(\nabla_\theta \psi^T \left(A^T A\right)^{-1} A^T \Delta B\right)$$
$$= tr\left(\left(AC\right)^T \Delta B\right)$$

و نتیجه مطلوب عبارت است از.

$$\nabla_B f = AC$$

اگر ماتریس A چگال باشد $\nabla_B\psi$ و $\nabla_A\psi$ هم ابعاد ماتریس های B و A می شوند و می توان به صورت صریح محاسبه کرد. پیچیدگی کل محاسبات گرادیان‌های فوق درحالت چگال $kn(n+m)$ می شود. وقتی که A ماتریس تنک باشد محاسبه گرادیان $\nabla_B\psi$ امکان پذیر ولی محاسبه و ذخیره گرادیان $\nabla_A\psi$ به صورت صریح امکان پذیر نیست. درحالتی که متغیر ω فقط زیر مجموعه از درایه های A مانند Γ را تحت تاثیر قرار می‌دهد. یعنی

$$A_{ij}\left(\omega\right) = 0, \quad i, j \notin \Gamma$$

گرادیان $\nabla_A\psi$ همان الگوی تنک بودن ماتریس A راخواهد داشت و فقط نیاز است عبارت زیر حساب شود.

$$\nabla_A\psi = \begin{cases} \left(b_i - \theta a_i\right)c_j^T - a_i^T\left(C\theta^T\right)_j & i, j \in \Gamma \\ 0 & o.w \end{cases}$$

که در اینجا b_i و a_i به ترتیب i امین سطر ماتریس B و A و c_j و $\left(C\theta^T\right)_j$ به ترتیب j امی ستون ماتریس‌های C و $C\theta^T$ هستند.

با دانستن $\nabla_\omega A_{ij}, \nabla_\omega B_{ij} \in R^p$ می توان $g = \nabla_\omega\psi\left(\theta^{ls}\left(\omega\right)\right)$ را به صورت زیر با استفاده ازمشتق زنجیری بدست اورد.

$$g = \sum_{i,j} \left(\nabla_A \psi\right)_{i,j} \left(\nabla_\omega A\right)_{i,j} + \sum_{i,j} \left(\nabla_B \psi\right)_{i,j} \left(\nabla_\omega B\right)_{i,j}$$

اگر که ماتریس ها تنک باشند جمع فوق روی درایه های غیر صفر بسته می شود.

$$g = \sum_{i,j \in \Gamma} \left(\nabla_A \psi\right)_{i,j} \left(\nabla_\omega A\right)_{i,j} + \sum_{i,j} \left(\nabla_B \psi\right)_{i,j} \left(\nabla_\omega B\right)_{i,j}$$

تعمیم مسئله حالت مقید با قید تساوی

در این قسمت مساله را به حالت مقید توسعه می دهیم.

$$\begin{aligned} \min \quad & \frac{\| A(\omega)\theta - B(\omega) \|_F^2}{2} \\ subject.to \quad & C(\omega)\theta = D(\omega) \end{aligned}$$

که $(\theta,\upsilon) \in R^{n\times m} \times R^{d\times m}$ و دوتایی متغیرهای اولیه و دوگان $\theta \in R^{n\times m}, C: \ \Omega \ \rightarrow R^{d\times n}, D: \ \Omega \ \rightarrow R^{d\times m}$ بهینه هستند اگر و تنها اگر در شرایط KKT زیر صدق کنند.

$$\begin{bmatrix} 0 & A(\omega)^T & C(\omega)^T \\ A(\omega) & -I & 0 \\ C(\omega) & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ q \\ \upsilon \end{bmatrix} = \begin{bmatrix} 0 \\ B(\omega) \\ D(\omega) \end{bmatrix}$$

که $q = A(\omega)\theta - B(\omega)$ بردار مانده است. قرار دهید

$$M(\omega) = \begin{bmatrix} 0 & A(\omega)^T & C(\omega)^T \\ A(\omega) & -I & 0 \\ C(\omega) & 0 & 0 \end{bmatrix}$$

اگر ماتریس‌های A, C چگال باشند با استفاده از تجزیه LDL^T می توان مساله فوق را حل کرد. درصورت تنک بودن ماتریس های نامبرده با استفاده از روش های تکرار یا سالورهای مخصوص ماتریس اسپارس LDL^T مساله قابل حل است.

تابع هدف اصلی، تابعی از متغیرهای θ, υ خواهد شد. فرض کنید که $\nabla_\theta \psi, \nabla_\upsilon \psi$ قابل محاسبه باشد ابتدا

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = M(\omega)^{-1} \begin{bmatrix} \nabla_\theta \psi \\ 0 \\ \nabla_\upsilon \psi \end{bmatrix}$$

را محاسبه کنید. گرادیان ψ نسبت به A, B به صورت زیر قابل محاسبه است:

$$\nabla_A \psi = -\left(\upsilon g_1^T + g_2 \theta^T\right), \nabla_B \psi = g_2$$

و مشابها برای C, D داریم.

$$\nabla_C \psi = -\left(\upsilon g_1^T + g_3 \theta^T\right), \nabla_D \psi = g_3$$

محاسبه گرادیان نگاشت جواب، نیازمند به حل یک معادله خطی است، بنابراین پیچیدگی آن به صورت تقریبا به اندازه پیچیدگی محاسبه خود جواب است. در صورتی که فاکتورگیری ها ذخیر شده باشد پیچیدگی خیلی کمتر خواهد شد. اگرماتریس های A, C

تنک باشند فقط نیازمند این هستیم که روی اعضای غیرصفر گرادیان را حساب کنیم.

در زیر به محاسبه دقیق فرمول‌های ذکر شده می‌پردازیم:

$$M = \begin{bmatrix} 0 & A^T(\omega) & C^T(\omega) \\ A(\omega) & -I & 0 \\ C(\omega) & 0 & 0 \end{bmatrix}$$

$$\phi(A) = M^{-1}S = Z, \qquad S = \begin{bmatrix} 0 \\ B(\omega) \\ D(\omega) \end{bmatrix}$$

$$D_A\phi(A)\Delta A = \phi(A + \Delta A) - \phi(A)$$

$$\phi(A + \Delta A) = (M + \Delta M)^{-1}S = (I + M^{-1}\Delta M)M^{-1}S$$

$$\simeq (I - M^{-1}(\Delta M)M^{-1}S) = \phi(A) - M^{-1}(\Delta M)Z$$

در نتیجه

$$D_A\phi(A)\Delta A = M^{-1}(\Delta M)Z$$

قرار دهید $C = M^{-1}\nabla_\theta\psi, F = \psi o\phi$

$$D_A\, f(A)\Delta A = D_{\theta^{ls}}\psi D_A\phi(A)\Delta A = tr(\nabla_\theta\psi^T M^{-1}(\Delta M)Z),$$

$$\begin{cases} \Delta M^T = \Delta M \\ (M^{-1})^T = M^{-1} \end{cases}$$

داریم

$$\Delta M\, C = \begin{bmatrix} 0 & \Delta A^T & 0 \\ \Delta A & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} \Delta A^T g_2 \\ \Delta A g_1 \\ 0 \end{bmatrix}, \qquad C = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

$$tr(AB) = tr(BA), \qquad tr(X^T) = tr(X)$$

در نتیجه داریم

$$tr(Z^T\Delta M\, C)^T = tr((\theta^T, q^T, v^T)\cdot(\Delta A^T g_2, \Delta A g_1, 0))$$

$$= tr((\theta^T\Delta A^T g_2 + q^T\Delta A g_1)^T)$$

$$= tr\big((g_2\theta^T + q^T g_1)\Delta A\big), \qquad q^T = r_{\text{مانده}}$$

$$\nabla_A\psi = \nabla_A f = g_2\theta^T + q^T g_1 = g_2\theta^T + r g_1$$

دوباره قرار دهید

$$\phi(B) = M^{-1}S$$

$$D_B\,\phi(B)\Delta B = \phi(B + \Delta B) - \phi(B)$$

$$= M^{-1}(S + \Delta S) - M^{-1}(S) = M^{-1}\Delta S$$

داریم $f = \psi o \phi$

$$D_B\, f(B)\Delta B = D_{\theta^{ls}}\psi D_B\phi(B)\Delta B = tr(\nabla_\theta \psi^T M^{-1}\Delta S) = tr(\Delta S^T C)$$

$$\Delta S^T = (0, \Delta B^T, 0)$$

$$\Delta S^T C = (0, \Delta B^T, 0)\cdot(g_1, g_2, g_3) = \Delta B^T g_2$$

$$tr\big(\Delta S^T C\big) = tr(g_2^T \Delta B)$$

$$\nabla_B f = \nabla_B \psi = g_2$$

مشابها برای C, D نیز می‌توان محاسبات فوق را انجام داد.

روش حداقل مربعات برای برازش داده ها

در مساله برازش داده ها، باید داده های آموزشی که شامل داده های ورودی به همراه مقدار مربوط به ورودی دردسترس باشد.

هدف برازش کردن پارامترهای، تابع پیش-بینی کننده زیر است.

$$\hat{y} = \phi\left(u, \omega^{feat}\right)^T \theta$$

که $\theta \in R^{n\times m}$ متغیرهای مدل و $\omega^{feat} \in \Omega^{feat} \subseteq R^{p^{feat}}$ هایپر-پارامترهای مهندسی کردن پارامترها است. توجه کنید که Ω^{feat} دامنه هایپر-پارامترهای مهندسی کردن ویژگی ها است و $\phi : U \times \Omega^{feat} \to R^n$ تابع ویژگی سازاست، که مشتق پذیر نسبت به هایپر-پارامترها در نظر گرفته می شود. تابع پیش-بینی کننده نسبت به بردار ویژگی ها خطی است.

برای انتخاب یک مدل باید مساله حداقل مربعات با داده های زیر حل شود.

$$A(\omega) = \begin{bmatrix} e^{\omega_1^{data}} \phi\left(u, \omega^{feat}\right)^T \\ . \\ . \\ . \\ e^{\omega_N^{data}} \phi\left(u, \omega^{feat}\right)^T \\ e^{\omega_1^{reg}} R_1 \\ . \\ . \\ . \\ e^{\omega_d^{reg}} R_1 \end{bmatrix}, B(\omega) = \begin{bmatrix} e^{\omega_1^{data}} y_1 \\ . \\ . \\ . \\ e^{\omega_N^{data}} y_1 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

که $\omega^{data} \in \Omega^{data} \subseteq R^N$ هایپر-پارامترهای وزنی هستند. $R_1, ..., R_d$ ماتریس های منظم ساز با سایز مناسب و $\omega^{reg} \in \Omega^{reg} \subseteq R^d$ هایپر-پارامترهای منظم ساز هستند. مجموعه کل هایپر-پارامترهای به صورت زیر نمایش داده می شود.

$$\omega = \left(\omega^{feat}, \omega^{data}, \omega^{reg}\right) \in \Omega^{feat} \times \Omega^{data} \times \Omega^{reg}$$

فرض براین است که تابع منظم ساز جداپذیر است یعنی

$$r(\omega) = r^{feat}\left(\omega^{feat}\right) + r^{data}\left(\omega^{data}\right) + r^{reg}\left(\omega^{reg}\right)$$

این بدین معناست که عملگرا پراکسیمال جداپذیراست. توابع فوق به ترتیب توابع منظم ساز برای هایپر-پارامترهای ویژگی ها و وزن داده ها و خود تابع منظم ساز است.

تابع هدف حقیقی

فرض کنید که داده های اعتبار سنجی از ورودی های $u_1^{val},...,u_{N_{val}}^{val} \in U$ و $y_1^{val},...,y_{N_{val}}^{val} \in R^m$ تشکیل شده باشند. ابتدا پیش-بینی را انجام می دهیم.

$$\hat{y}_i^{val} = \phi(u_i^{val})\theta^{ls}(\omega)$$

که در اینجا ویژگی ها یا تابع $\phi\left(u,\omega^{feat}\right)$ مشخص و ثابت در نظر گرفته می شود. تابع هدف حقیقی ψدر مساله برازش داده ها با حداقل مربعات را میانگین زیان برای پیشبینی داده های اعتبار سنجی بصورت زیر در نظر میگیریم.

$$\psi(\theta) = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} l\left(\hat{y}_i^{val}, y_i^{val} \right)$$

که $l : R^m \times R^m \rightarrow R$ تابع جریمه است، و فرض می شود که نسبت به متغیر اول خود مشتق پذیراست. توابع حقیقی پیچیده مانند تابع زیان اعتبار سنجی متقاطع با K لایه نیز استفاده کرد.

روش فوق برای خیلی از مسائل برازش داده از جمله رگرسیون و طبقه بندی کاربرد دارد. در رگرسیون خروجی عددی حقیقی و در رگرسیون چند- وظیفه ای خروجی یک بردار با درایه های حقیقی، و درطبقه بندی خروجی متغیر بولی و در طبقه بندی چند-وظیفه‌ای خروجی مشخص کننده کلاس برای داده ورودی است.

تابع جریمه رگرسیون

تابع جریمه در رگرسیون اکثر اوقات به صورت زیر است

$$l\left(\hat{y}, y\right) = \pi\left(r\right)$$

که $r = \hat{y} - y$ مانده و $\pi : R^m \rightarrow R$ تابع جریمه اعمال شده بر روی مانده است. برخی از توابع رایج برای تابع جریمه بصورت زیر هستند.

- تابع مربعی: تابع جریمه مربعی. $\pi\left(r\right) = \| r \|_2^2$
- تابع هوبر: تابع جریمه هوبر، جریمه قدرتمندی به شکل تابع جریمه مربعی برای مانده های کوچک وبصورت ۲-نرمی برای مانده های بزرگ می شود.

$$\pi\left(r\right) = \begin{cases} \| r \|_2^2 & \| r \|_2 \le M \\ M\left(2\| r \|_2^2 - M\right) & \| r \|_2 > M \end{cases}$$

که متغیر M خود یک هایپر-هایپر-پارامتر است.

- دو مربعی: جریمه دو مربعی، جریمه قدرتمند با مقدار ثابت برای مانده های بزرگ است. متغیر M خود یک هایپر-هایپر-پارامتر است

$$\pi\left(r\right) = \begin{cases} \dfrac{M^2}{6}\left(1 - \left(1 - \dfrac{\| r \|_2^2}{M^2}\right)^3\right) & \| r \|_2 \le M \\ \dfrac{M^2}{6} & \| r \|_2 > M \end{cases}$$

تابع جریمه طبقه بندی

برای طبقه بندی، پیش بینی $\hat{y} \in R^m$ متناسب با توزیع احتمال روی m برچسب به صورت زیر داده می شود.

$$\Pr\left(y = e_i\right) = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{m} e^{\hat{y}_j}}, \quad i = 1, 2, ..., m$$

پیش بینی $\hat{y}$ بعنوان یک توزیع احتمال شرطی برروی برچسب های y برای x داده شده تعبیر می شود. تابع زیان آنتروپی متقابل نیز یک تابع پیشنهادی بعنوان تابع جریمه در طبقه بندی محسوب می شود و بصورت زیر است:

$$l\left(\hat{y}, y\right) = -\hat{y}_i + \log\left(\sum_{j=1}^{m} e^{\hat{y}_j}\right) \quad i = 1, 2, ..., m$$

وزن دهی داده ها

در مسئله برازش داده با حداقل مربعات هایپر پارامتر ω_i^{data} به خطای مربعی داده (u_i, y_i) وزن $e^{2\omega_i^{data}}$ را نسبت می‌دهد. اگر ω_i^{data} کوچک باشد داده متناظر نقش کمی در محاسبه پارامتر های مدل دارد و بالعکس. وزن دهی هر داده درمسئله با تابع زیان مربعی همانند این است که تابع زیان غیرمربعی در مسئله استفاده شده است به همین دلیل تنظیم خودکار هایپر-پارامترهای وزن دار کار انتخاب تابع زیان را ساده می کند. اما دامنه هایپر-پارامتر وزن دهی به صورت زیر درنظر گرفته می شود.

$$\Omega^{data} = \left\{ x \,|\, 1^T x = 0 \right\}$$

و این بدین معناست که میانگین هندسی $\exp\left(\omega^{data}\right)$ مساوی یک است. هایپر-پارامتر را می توان به سمت وزن‌های یکسان با استفاده از منظم ساز $r^{data}(\omega) = \frac{\lambda}{2} \| \omega \|_2^2$ یا $r^{data}(\omega) = \lambda \| \omega \|_1$ سوق داد، که مجددا λ یک هایپر-هایپر-پارامتر مثبت است. عملگر پراکسیمال برای $r^{data}(\omega) = \frac{\lambda}{2} \| \omega \|_2^2$ با $\Omega = \Omega^{data}$ در υ با طول گام t متناظر با حل مساله بهینه سازی زیر می شود:

$$\begin{aligned} \min \quad & t\frac{\lambda}{2} \| \omega \|_2^2 + \frac{1}{2} \| \omega - \upsilon \|_2^2 \\ subject.to \quad & 1^T \omega = 0 \end{aligned}$$

که جواب تحلیلی آن به صورت زیر خواهد شد.

$$L = t\frac{\lambda}{2} \| \omega \|_2^2 + \frac{1}{2} \| \omega - \upsilon \|_2^2 + k1\omega^T$$

$$\nabla_\omega L = t\lambda\omega^* + \left(\omega^* - \upsilon\right) + k1 = 0$$

$$\omega^* = \frac{1}{1+t\lambda}\left(\upsilon - k1\right)$$

$$1^T\omega^* = \frac{1}{1+t\lambda}\left(1^T\upsilon - k1^T1\right) = 0$$

$$k = \frac{1^T\upsilon}{p}$$

پس داریم

$$\omega^* = \frac{1}{1+t\lambda}\left(\upsilon - \frac{1^T\upsilon}{p}\right)$$

منظم ساز

هایپر-پارامتر ω^{reg} از طریق $\sum_{i=1}^{d}\exp\left(2\omega_i^{reg}\right)\|R_i\theta\|_F^2$ روی محاسبه پارامترها تاثیر می گذارد. هرجمله $\|R_i\theta\|_F^2$

یک معیار برای میزان پیچیدگی θ در نظر گرفته می شود. ساده ترین شکل ماتریس $R_i = diag\left(e_i\right)$ یا همان ماتریس همانی است که منجر به رگرسیون ریج می شود. انتخاب دیگر برای R_i می تواند ماتریس وقوع گراف برای اعضای هر ستون θ باشد که هایپر-پارامتر منظم ساز در اینجا میزان اهمیت منظم سازی گراف را نشان می دهد.

مهندسی کردن ویژگی ها

هایپر-پارامتر ω^{feat}، ویژگی ساز ها را پارامتریزه می کند. هدف از انتخاب ω^{feat} اینست که خروجی y را بصورت تقریبا خطی از $\phi\left(u,\omega^{feat}\right)$ می کند. که دامنه ورودی اول فرض میشود که فضای برداری U است.

معمولا تابع ϕ بصورت زنجیره ای از تولید ویژگی ها ساخته می شود. یعنی به صورت ترکیبی از توابع مهندسی سازی ویژگی ها به صورت $\phi_1,...,\phi_l$ است پس داریم.

$$\phi = \phi_l \circ . . . \circ \phi_1$$

توجه کنید معمولا آخرین تابع مهندسی سازی یک ثابت به ترکیب توابع قبلی اضافه میکند که نتیجه ترکیب یه تابع آفین می‌شود.

توابع مهندسی سازی ویژگی‌های اسکالر

در این بخش به توابع مهندسی سازی ویژگی ها $\phi : R \rightarrow R$ با این فرض که می توان آنها را به مولفه های یک بردار با هایپر-پارامترهای متفاوت اعمال کرد، خواهیم پرداخت.

- مقیاس بندی: یکی از ساده ترین توابع مهندسی سازی ویژگی ها مقیاس بندی خطی است که به صورت زیر داده می‌شود.

$$\phi\left(x,(a,b)\right)=ax+b$$

یک رویه رایج در برازش داده این است که داده ها را با استفاده از مقیاس بندی به پارامترهای $a=\frac{1}{\sigma},b=-\frac{\mu}{\sigma}$ استاندارد می کنند. در اینجا μ میانگین و σ انحراف معیار استاندارد متغیرتصادفی x هستند. در مساله حداقل مربع با تنظیم خودکار، a و b بر اساس داده ها تعیین می شوند.

- تبدیل توانی: تبدیل توانی به صورت $\phi\left(x,(c,\gamma)\right)=\text{sgn}\left(x-c\right)|x-c|^{\gamma}$ تعریف می شود.تابع sgn همان تابع علامت و مقیاسبند $\gamma \in R$ و پارامتر $c \in R$ هایپر-پارامترهای مساله هستند. برای مقادیر متفاوت γ و c تابع تبدیل متفاوتی بدست می آید. مثلا اگر $\gamma=1,c=0$ تابع تبدیل همانی و اگر $\gamma=0$ تابع تبدیل تعیین کننده این است که x سمت راست یا چپ مقدار c قرار دارد. و اگر $\gamma=\frac{1}{2},c=0$ این تبدیل همان ریشه قدر مطلق می شود. توجه کنید که تبدیل توانی همه جا به غیر از $\gamma=0$ مشتق پذیراست.
- چند جمله ای های اسپلاین: چندجمله ای اسپلاین یک تابع قطعه چند جمله ای است. چند جمله ای اسپلاین متشکل از یک آرایه صعودی از اعداد $z \in R^{k+1}$ که گره های اسپلاین را مشخص می‌کند به همراه درجه d و ضرایب چندجمله ای $f_0, \dots f_{k+1} \in R^{d+1}$ است .

$$\phi\left(x,(z,f_1,\dots,f_{k+1})\right)=\begin{cases}\sum_{i=0}^{d}(f_0)_i\,x^i & x\in(-\infty,z_1)\\ \sum_{i=0}^{d}(f_j)_i\,x^i & x\in(z_j,z_{j+1}) \quad j=1,\dots,k\\ \sum_{i=0}^{d}(f_{k+1})_i\,x^i & x\in(z_{k+1},\infty)\end{cases}$$

توجه داشته باشید که r^{feat},Ω^{feat} برای اعمال شرط پیوستگی و پیوستگی مشتق ها در نقاط گره‌ای به بکار برده می‌شود.

توابع چند بعدی برای مهندسی سازی ویژگی‌ها

- رتبه پایین: ویژگی ساز ϕ می تواند یک تبدیل رتبه پایین به صورت زیر باشد.

$$\phi(x,T)=Tx$$

که $T\in R^{r\times n}$, $r<n$. درعمل یک انتخاب رایج برای عملگر T انتخاب چند بردار اول در تجزیه SVD ماتریس داده است که اب استفاده از تنظیم خود کار انتخاب T به صورت خودکار صورت می‌گیرد.

- شبکه های عصبی: تابع ϕ می تواند یک شبکه عصبی باشد، در اینصورت ω^{feat} همان پارامتر های شبکه عصبی می‌شوند.
- انتخاب ویژگی: کسر f از ویژکی ها را می توان با $\phi(x)=diag(a)x$ انتخاب کرد که

$$\Omega^{feat}=\left\{a\mid a\in\{0,1\}^{n_1},1^T a=\lfloor fn_1\rfloor\right\}$$

که $\lfloor\ \rfloor$ همان تابع جز صحیح کف و f هایپر-هایپر-پارامتر مساله است.

مجموعه تست و توقف زود هنگام

وقتی که تعداد هایپر-پارامترها زیاد می‌شود ریسک برازش بیش ازحد وجود دارد زیرا تنظیم خودکار حداقل مربعات بصورت مستقیم تابع زیان روی مجموعه اعتبار سنجی را کمینه می‌کند. برای تشخیص این برازش بیش از حد، یک مجموعه سوم از داده‌ها به اسم مجموعه داده تست در نظر گرفته می‌شود که در آخر الگوریتم روی این مجموعه محاسبه می شود. مقدار زیان مجموعه اعتبارسنجی لزومی ندارد که تخمین خوبی از اندازه دقیق عملکرد مدل بر روی داده های دیده نشده باشد.

روشی کمی متفاوت برای رویارویی با مساله برازش بیش از حد این است که مقدار زیان روی مجموعه داده تست در هر تکرار محاسبه کرده و وقتی که مقدار زیان روی مجموعه داده تست شروع به افزایش کرد، الگوریتم را متوقف می‌کنند. به این روش توقف زود هنگام می‌گویند. دراین روش نیاز به مجموعه داده چهارمی است که به عنوان مجموعه داده تست نهایی شناخته شده است، وقتی که الگوریتم متوقف شد، روی مجموعه داده نهایی الگوریتم حساب شده و بعنوان عملکرد الگوریتم در نظر گرفته می شود.

یک روش مناسب و قدرتمند استفاده از تابع زیان روش اعتبارسنجی متقابل است. به این ترتیب که گرادیان روش اعتبار سنجی متقابل همان میانگین گرادیان های حساب شده در هر مرحله اعتبار سنجی متقابل است.

داده های MNIST

مجموعه داده MNIST شامل ۶۰۰۰۰ نقطه آموزشی به همراه برچسب که هر کدام آن یک بردار از سایز۷۸۴ (یک بعدی شده ماتریس های تصویری خاکستری ۲۸ در۲۸) هستند. برچسب‌ها شامل ده کلاس متناسب با اعداد ۰تا ۹ می‌شوند. داده‌های تست شامل ۱۰۰۰۰ نقطه آموزشی به همراه برچسب هستند.

مساله ای که با استفاده از این روش حل می‌شود رگرسیون ریج با یک هایپرپارامتر است.

$$|| A\theta - B ||_F^2 + \exp(2\lambda) || \theta ||_F^2$$

برای حل این مساله می‌توان از تابع آنتروپی متقابل و یا خود تابع هدف مساله حداقل مربعات بعنوان تابع حقیقی هدف استفاده کرد.

همانطور که قبلا دیدیم جواب مساله به شکل زیر است:

$$X = \begin{bmatrix} A \\ \exp(\lambda) I \end{bmatrix} \quad C = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad X^T X \theta^{ls} = X^T C$$

توجه کنید که ماتریس C همان مقادیر برچسب به ازای هر نمونه اموزشی است و هر سطر آن یک بردار ۱۰ تایی که دارای یک مولفه غیر صفر و با مقدار یک است و جایگاه آن مولفه نشان دهنده برچسب مورد نظر است. مثلا اگر مولفه دهم یک باشد آن برچسب مساوی عدد ۹ و اگر جایگاه اول مساوی یک باشد برچسب نشان دهنده برچسب صفر است.

اکنون می خواهیم مشتق تابع θ^{ls} برحسب λ را بدست بیاوریم. برای سادگی $D = X^T X$ نشان می دهیم پس رابطه جواب مساله به صورت زیر است:

$$D\theta^{ls} = X^T C$$

با استفاده از خاصیت ضرب داریم:

$$\frac{dD}{d\lambda}\theta^{ls} + D\frac{d\theta^{ls}}{d\lambda} = \frac{dX^T}{d\lambda}C$$

برای محاسبه $\frac{d\theta^{ls}}{d\lambda}$ احتیاج به محاسبه مشتق $\frac{dD}{d\lambda}$ و $\frac{dX^T}{d\lambda}$ داریم زیرا

$$\frac{d\theta^{ls}}{d\lambda} = D^{-1}\left(\frac{dX^T}{d\lambda}C - \frac{dD}{d\lambda}\left(D^{-1}X^T C\right)\right)$$

حال ماتریس $X^T X$ را در نظر بگیرید.

$$D = X^T X = \begin{bmatrix} A^T & \exp(\lambda) I \end{bmatrix} \begin{bmatrix} A \\ \exp(\lambda) I \end{bmatrix} =$$

$$\begin{bmatrix} A^T A + \exp(2\lambda) I \end{bmatrix} \rightarrow \frac{dD}{d\lambda} = 2\exp(2\lambda) I$$

و مشابها برای X^T داریم

$$X^T = \begin{bmatrix} A^T & \exp(\lambda) I \end{bmatrix} \rightarrow \frac{dX^T}{d\lambda} = \begin{bmatrix} 0 & \exp(\lambda) I \end{bmatrix}$$

پس با توجه به محاسبات بالا $\frac{d\theta^{ls}}{d\lambda}$ را می توان حساب کرد. حال تابع هدف حقیقی را تابع آنتروپی متقابل در نظر بگیرید:

$$l\left(\hat{y}, y = e_i \right) = -\hat{y}_i + \log\left(\sum_{j=1}^{m} e^{\hat{y}_j} \right)$$

با اعمال تابع آنتروپی متقابل روی داده های اعتبارسنجی و میانگین گرفتن داریم:

$$\hat{y}_{val} = A_{val}\theta^{ls}, \quad \psi(\theta) = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} l\left(y_i^{val}, \hat{y}_i^{val} \right)$$

در الگوریتم نیاز به محاسبه $g = \nabla_\lambda \psi$ داریم. و چون $\psi(\theta)$ جمع برروی تمام داده ها اعتبارسنجی است فقط کافی است که $\frac{dl}{d\lambda}$ محاسبه شود.

$$\frac{dl}{d\lambda}\left(\hat{y}_j, y = e_i \right) = -(A_{val} \frac{d\theta^{ls}}{d\lambda})_{ji} + \frac{\sum_{l=1}^{m} e^{\hat{y}_{li}} (A_{val} \frac{d\theta^{ls}}{d\lambda})_{li}}{\sum_{l=1}^{m} e^{\hat{y}_{li}}}$$

که در اینجا اندیس اول اندیس مربوط به نمونه و اندیس دوم مربوط به مولفه های نمونه مربوطه است. که با جمع بستن روی تمام نمونه های اعتبارسنجی و میانگین گرفتن تابع g بدست می آید.

اگر تابع هدف حقیقی همان تابع هدف مساله حداقل مربعات باشد داریم:

$$\psi\left(\theta^{ls} \right) = \| X\theta^{ls} - b \|_F^2 + \exp(2\lambda) \| \theta^{ls} \|_F^2$$

و با مشتق گیری داریم:

$$g = \frac{d\psi}{d\lambda} = 2\left(X\theta^{ls} - b \right)^T \frac{d\theta^{ls}}{d\lambda} + 2\exp(2\lambda) \| \theta^{ls} \|_F^2 + 2\exp(2\lambda)\left(\theta^{ls} \right)^T \frac{d\theta^{ls}}{d\lambda}$$

که در اینجا X ماتریس ساخته شده از نمونه ها و b بردار برچسب هاست.

نتایج پیاده سازی

برای پیاده سازی ۲ تابع هدف حقیقی آنتروپی متقابل و تابع هدف حداقل مربعات را در نظر گرفتیم. رویکرد اول، استفاده از روش‌های ساخت classifier ها و ترکیب آن با الگوریتم تنظیم خودکار است؛ در اینجا ابتدا تابع هدف حقیقی را همان تابع هدف مسئله حداقل مربعات قرار داده و در پیاده سازی دیگری، تابع هدف را تابع زیان بر روی داده های اعتبارسنجی قرارداده و با استفاده از روش proximal gradient، تابع هدف را مینیمم سازی می‌کنیم. در کنار آن به مهندسی ویژگی نیز پرداخته‌ایم و تعداد مختلفی ویژگی تصادفی اضافه کرده‌ایم.

اما رویکرد دیگر، ابتدا از هر برچسب، یک بردار سطری به طول ۱۰ ایجاد می‌کنیم و جایگاه متناظر با مقدار برچسب را مقدار ۱ می‌دهیم. برای این بخش از میانگین تابع آنتروپی متقابل بر روی داده‌های اعتبارسنجی، به عنوان تابع هدف اصلی استفاده کرده‌ایم که محاسبات مربوط به آن قبلا در توضیحات آورده شده است.

قابل توجه است که در ادبیات، پیاده سازی های مرتبط با روش تنظیم خودکار، با استفاده از شبکه های عصبی صورت پذیرفته اما در این پروژه، تلاش بر این بوده است که با استفاده از روش‌های ساده بحث شده در کلاس به حل مسئله رگرسیون ریج بپردازیم.

در ادامه نتایج برای پیاده سازی مختلف آورده شده است.

در ابتدا به مقایسه محاسبه θ بااستفاده از دو روش convex optimization و least squares پرداختیم :

Convex Optimization:

```
15.594337 seconds
```

Least Squares:

```
7.582242 seconds
```

در نتیجه از روش حداقل مربعات که زمان کمتری میگیرد، در الگوریتم خود استفاده میکنیم.

- نتایج پیاده‌سازی با رویکرد اول:

Code: first_app_1.ipynb/.jl

Random features	Iterations	Accuracy
0	10	0.86
1200	5	0.9555
1500	5	0.9587

Code: first_app_2.ipynb/.jl

Train set	Random features	Validation set	Test set	Iteration	Accuracy
20000	0	1000	10000	2	0.85
40000	1000	4000	10000	2	0.94
40000	1000	4000	10000	6	0.95

- نتایج پیاده‌سازی با رویکرد دوم:

Code: second_app_1.ipynb/.jl

Train set	Random features	Validation set	Test set	Iteration	Accuracy
40000	1200	4000	10000	5	0.9573
50000	1200	5000	10000	10	0.9576
55000	1500	5000	10000	10	0.9581
59000	2000	1000	10000	5	0.9637
59000	5000	1000	10000	5	0.9726

- نتایج پیاده سازی با استفاده شبکه‌های عصبی ساده در ۱۰ تکرار (code: simple NN MNIST.ipynb/.jl):

test_loss = 0.22957107, test_accuracy = 0.9352

- نتایج پیاده سازی با استفاده از شبکه‌های عصبی پیچشی در ۱۰ تکرار (code: CNN on MNIST.ipynb):

Test: (loss = 0.0445f0, acc = 98.53)

همانگونه که از نتایج پیاده سازی مشخص است، تاثیر افزودن ویژگی‌های تصادفی بیشتر از منظم ساز در رگرسیون ریج است. با پیچیده تر کردن تابع هدف مسئله حداقل مربعات، بطور مثال با افزودن منظم ساز متناسب با نرم فروبنیوسی ماتریس وقوع گراف پیکسل‌ها، می‌توان اطلاعات مربوط به همواری عکس‌ها را نیز به تابع هدف اضافه نمود. همچنین می‌توان تابع هدف حقیقی را اینگونه تغییر داد که به جای استفاده از تابع زیان بر روی داده های اعتبارسنجی، با روش cross validation، ابتدا داده های اموزشی را به k قسمت تقسیم کرده و $k-1$ قسمت را بعنوان آموزش و باقی مانده را بعنوان داده‌ی اعتبارسنج استفاده کرد. تابع هدف حقیقی را می‌توان میانگین توابع زیان در هر مرحله از cross validation در نظر گرفت. مجددا می‌توان تابع زیان برای یک داده از داده‌های اعتبارسنجی، را همان تابع آنتروپی متقابل قرار داد.

منابع

[1] S. Barratt. On the differentiability of the solution to convex optimization problems. arXiv preprint arXiv:1804.05098, 2018.

[2] Shane Barratt, Stephen Boyd: Least Squares Auto-Tuning. arXiv:1904.05460v1, 2019.

[3] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind. Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research, 18:1-43, 2018.

[4] R. Eigenmann and J. Nossek. Gradient based adaptive regularization. In Proc. Neural Networks for Signal Processing, pages 87-94, 1999.

[5] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[6] Parikh, N., and S. Boyd. 2014. \Proximal algorithms." Foundations and Trends R in Optimization 1 (3): 127-239.