

B°۶۶A"۶۶"۹F۶۶"۸F۶۶"۷F۶۶"۶F۶۶"۵F۶۶"۴F۶۶"۳F۶۶"۲F۶۶"۱F۶۶"۰F۶۶"





دانشگاه شهید بهشتی

دانشکده علوم ریاضی

گروه علوم کامپیوتر

تمرین اول داده کاوی

نگارش: زهرا حق شناس

استاد: دکتر هادی فراهانی

۱۴۰۳ مهر

سوال ۱

مقایسه رگرسیون لجستیک، تحلیل تقسیمی خطی (LDA) و ماشین‌های بردار پشتیبان .(SVM) تحت چه شرایطی رگرسیون لجستیک ممکن است از LDA بهتر عمل کند و برعکس؟ علاوه بر این، کدام یک برای داده‌های با ابعاد بالا و ویژگی‌های زیاد مناسب‌تر است: SVM یا، LDA و چرا؟

پاسخ

۱. رگرسیون لجستیک

رگرسیون لجستیک یک روش احتمالاتی است که برای پیش‌بینی دسته‌بندی‌های دودویی استفاده می‌شود. این روش از تابع لجستیک برای محاسبه احتمال قرارگیری داده‌ها در یکی از دو کلاس استفاده می‌کند. ما می‌خواهیم احتمال $Pr(Y|X)$ که به اختصار $p(X)$ نمایش می‌دهیم را مدل کنیم. به عبارت دیگر می‌توان این طور بیان کرد که می‌خواهیم احتمال 50% (X) را به عنوان پیش‌بینی طبقه ۱ در دسته بندی کلاس‌ها به دست آوریم.

مدل لاجستیک

مدل رگرسیون خطی برای مدل سازی این احتمال $p(x) = \beta_0 + \beta_1 x$ مشکلی که داشت این بود که این احتمال اعتبار را در بازه $[0, 1]$ تخمین نمی‌زد. اما ما باید از تابعی استفاده کنیم که همواره خروجی بین 0 و 1 بدهد. تابعی که مناسب این کار است تابع لاجستیک است.

$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

برای تخمین ضرایب این مدل از روشی به نام maximum likelihood استفاده می‌کنیم. این تابع همیشه مقداری بین صفر و یک می‌دهد و به همین خاطر مناسب است.

رابطه لاجیت

به این رابطه *logit* یا لگاریتم بخت گفته می‌شود. این رابطه به صورت زیر است:

$$\log \left(\frac{p(x)}{1 - p(x)} \right) = \beta_0 + \beta_1 x$$

مدل رگرسیون لاجستیک به این صورت است که احتمال وقوع رخدادی با یک متغیر مستقل x مدل‌سازی می‌شود. به‌طور خاص، احتمال را به شکل زیر تعریف می‌کنیم:

$$p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

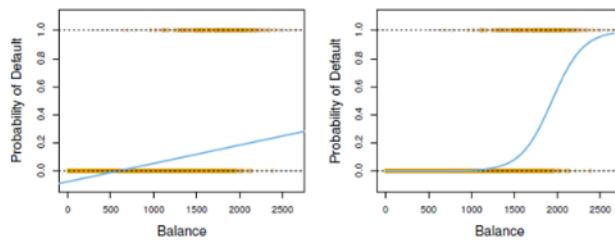
این مدل احتمال را بین صفر و یک نگه می‌دارد. حال می‌توانیم رابطه‌ی شرطی را که به آن odds گفته می‌شود، به فرم لگاریتمی بازنویسی کنیم:

$$\frac{p(x)}{1 - p(x)} = e^{\beta_0 + \beta_1 x}$$

با گرفتن لگاریتم دو طرف داریم:

$$\log \left(\frac{p(x)}{1 - p(x)} \right) = \beta_0 + \beta_1 x$$

این رابطه‌ی logit است. به این ترتیب، بهازای هر واحد افزایش در متغیر ورودی x ، ضریب β_1 تعیین می‌کند که چقدر احتمال رخداد پیش‌بینی شده تغییر می‌کند. بهازای افزایش یک واحد در x ، رابطه‌ی ورودی-خروجی تغییر خواهد کرد. در رگرسیون لاجیستیک رابطه ورودی و خروجی خطی نیست و یک واحد افزایش ورودی منجر به β_1 واحد افزایش در لگاریتم بخت و e^{β_1} واحد افزایش در بخت می‌شود. اما میزان تاثیری که یک واحد افزایش ورودی در $p(x)$ دارد، به میزان خود x بستگی دارد و مقدار ثابتی نیست. برای مقادیر مثبت x ، منجر به افزایش خروجی و برای مقادیر منفی x منجر به کاهش خروجی می‌شود.



مسئله کارت های بانکی را در نظر بگیرید. می خواهیم احتمال $Pr(default = Yes | balance)$ که به اختصار به صورت $p(balance)$ نشان می دهیم، را مدل کنیم. در شکل بالا نمودار سمت راست تابع لاجیستیک به دست آمده برای مسئله کارت های بانکی را نشان می دهد که احتمال بین ۰ و ۱ است در حالی که در شکل سمت چپ مدل رگرسیون مکان احتمال را زیر ۱ یا بالا گرفته است.

تخمین ضرایب رگرسیون لجستیک

ضرایب β_0 و β_1 در مدل $p(x) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$ مجهول هستند و باید با استفاده از داده‌های آموزش تخمین زده شوند. در مدل غیرخطی لجستیک از تابع درستنمایی برای تخمین پارامترها استفاده می‌کنیم. در واقع ما به دنبال پارامترهایی هستیم که وقتی آنها را در مدل جایگذاری کردیم، $(X)^{\hat{p}}$ برای داده‌هایی که کارگزاران پیش‌فرض هستند نزدیک ۱ و برای کسانی که کارگزاران غیر پیش‌فرض هستند نزدیک صفر شود. این مفهوم را می‌توان به عبارت ریاضی زیر فرمول کرد:

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

به این فرمول، تابع درستنمایی می‌گویند و هدف این است که پارامترهای β_1 و β_0 را به گونه‌ای انتخاب کنیم که مقدار تابع درستنمایی بیشینه شود.

رگرسیون لاجستیک چندمتغیره

حال می‌خواهیم به مسئله کلاس بندی دو کلاسه با استفاده از چندین متغیر ورودی بپردازیم، همانند تجمعی که برای رگرسیون داشتیم، اینجا هم مسئله کلاس بندی را به این صورت تعریف می‌کنیم:

$$\log \left(\frac{p(x)}{1 - p(x)} \right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

بنابراین خواهیم داشت:

$$p(x) = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p}}$$

برای تخمین پارامترهای $\beta_0, \beta_1, \dots, \beta_p$ مجدداً از روش بیشینه درستنمایی استفاده می‌کنیم.
ویژگی‌های کلیدی:

- سادگی: مدل به راحتی تقسیم می‌شود.
- احتمالمحور بودن: پیش‌بینی‌ها را بر اساس احتمال وقوع دسته‌بندی انجام می‌دهد.
- کاربرد گسترده: در داده‌هایی که محدودیت نرمالیتی ندارند یا فرض واریانس‌های مشابه بین گروه‌ها برقرار نیست، عملکرد خوبی دارد.

شرایط مناسب برای رگرسیون لجستیک:

- اگر داده‌ها از توزیع نرمال پیروی نکنند، رگرسیون لجستیک به دلیل نداشتن فرض نرمالیتی بهتر عمل می‌کند.
- هنگامی که داده‌های آموزشی کم حجم باشند، پیچیدگی کمتر رگرسیون لجستیک مزیت محسوب می‌شود.

۲. تحلیل تفکیکی خطی (LDA)

LDA یک روش آماری برای یافتن ترکیب خطی از ویژگی‌ها است که بیشترین تفکیک بین کلاس‌ها را ایجاد می‌کند. در این روش به جای آنکه مستقیماً $Pr(Y = k|X = x)$ مدل کنیم، ابتدا احتمال توزیع متغیر ورودی به شرط کلاس X را به دست می‌آوریم و از روی آن با استفاده از قضیه بیز احتمال $(Pr(Y = k|X = x))$ را محاسبه می‌کنیم. چراگاهی از این روش به جای لاجستیک استفاده می‌کنیم؟

- وقتی کلاس‌ها به خوبی از هم تفکیک شده‌اند، روش لاجستیک ناکاراست.
- وقتی تعداد داده‌های آموزشی در هر کلاس بسیار کم است، روش *discriminant linear* بهتر از لاجستیک عمل می‌کند.
- روش *discriminant linear* برای حالت چندکلاسه معروف تر است.

استفاده از قضیه بیز برای کلاس‌بندی

فرض کنید که K کلاس داریم و می‌خواهیم K متغیر پیش‌بینی‌کننده X را به یکی از این کلاس‌ها نسبت دهیم. هدف ما محاسبه احتمال $Pr(Y = k|X = x)$ است. برای محاسبه این احتمال، داده‌های عضو کلاس k را به شکل توزیع متغیر ورودی برای کلاس k مدل می‌کنیم.

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

در اینجا π_k احتمال پیشین کلاس k و $f_k(x) = Pr(X = x|Y = k)$ توزیع متغیر ورودی برای کلاس k است.

حالت تک متغیره

ابتدا فرض می‌کنیم فقط یک متغیر ورودی داریم و فرض می‌کنیم $f_k(x)$ برای تمام کلاس‌ها دارای توزیع نرمال به فرم زیر است:

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

که در آن μ_k و σ_k^2 میانگین و واریانس کلاس k هستند. حالا می‌توانیم $Pr(Y = k|X = x)$ را برای این حالت محاسبه کنیم. اگر فرض کنیم که واریانس تمام کلاس‌ها با هم برابر است یعنی $\sigma_1 = \sigma_2 = \dots = \sigma_k = \sigma$ ، با جایگذاری توزیع نرمال فوق در رابطه $\sum_{l=1}^K \pi_l f_l(x)$ خواهیم داشت:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}$$

حال با لگاریتم گرفتن و ساده کردن برخی جملات به فرمول زیر می‌رسیم:

$$\delta_k(x) = \frac{\mu_k}{\sigma^2} x - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

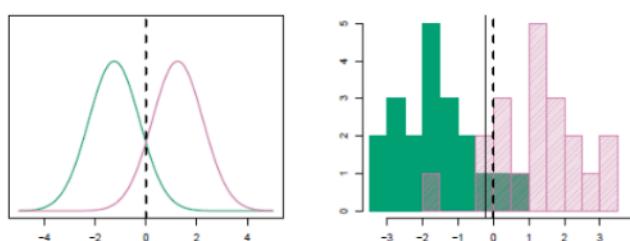
هر داده‌ای را به کلاس k -امی نسبت می‌دهیم که دارای مقدار $\delta_k(x)$ بزرگتری باشد. مثلاً اگر $K = 2$ و $\pi_1 = \pi_2 = 1/2$ باشد، داده را به کلاس ۱ نسبت می‌دهیم اگر:

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$$

بنابراین مرز تصمیم‌گیری نقطه زیر است:

$$x = \frac{\mu_1^2 - \mu_2^2}{2x(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

به عنوان مثال داده‌هایی که در سمت چپ شکل زیر مشاهده می‌کنید از دو توزیع با واریانش‌های یکسان و برابر ۱ و میانگین‌های $\mu_1 = -1/25$ ، $\mu_2 = 1/25$ آمده‌اند. اگر $\pi_1 = \pi_2 = 0.5$ مرز تصمیم، $x = 0$ با خط نقطه چین در شکل نشان داده شده است. بنابراین داده‌های با $x < 0$ به کلاس سبز و داده‌های با $x > 0$ به کلاس بنفش تعلق خواهند گرفت.



اما ما همیشه مقادیر دقیق پارامترهای میانگن و واریانس را نداریم و باید این مقادیر را از روی نمونه‌های آموختشی به صورت زیر تخمین بزنیم:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{k=1}^K \sum_{i:y_i=k} \frac{(x_i - \hat{\mu}_k)^2}{n_k}$$

$$\hat{\pi}_k = \frac{n_k}{n}$$

در اینجا، n تعداد کل نمونه های آموزشی، n_k تعداد کل نمونه های آموزشی که به کلاس k اختصاص داده شده است. داده x به کلاس k نسبت داده می شود اگر مقدار $\delta_k(x)$ بیشتر باشد:

$$\delta_k(x) = \frac{x\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

ویژگی های کلیدی:

- فرض نرمال بودن داده ها: LDA فرض می کند که داده ها از توزیع نرمال چند متغیره پیروی می کنند.
- واریانس برابر بین کلاس ها: فرض بر این است که واریانس در تمامی کلاس ها یکسان است.

شرایط مناسب برای LDA:

- اگر داده ها از توزیع نرمال پیروی کنند و واریانس ها مشابه باشند، LDA به دلیل استفاده بهینه از واریانس و میانگین، عملکرد بهتری نسبت به رگرسیون لجستیک دارد.

۳. ماشین های بردار پشتیبان (SVM)

SVM یک ابزار قوی برای کلاس بندی در مساله های دو کلاسه است که تعمیم از روش کلاس بند بیشترین حاشیه (maximal margin) است. SVM یک الگوریتم یادگیری ناظارت شده است که از فوق صفحه برای تفکیک کلاس ها استفاده می کند و از هسته ها classifier) برای نگاشت داده ها به فضای ویژگی های بالاتر بپره می برد (kernels). در یک فضای p بعدی، یک ابرصفحه یک زیرفضای $1-p$ بعدی است. مثلاً در فضای دو بعدی، ابرصفحه یک زیرفضای یک بعدی (یعنی یک خط) است. در فضای سه بعدی، یک ابرصفحه یک زیرفضای دو بعدی است. در فضاهای بیشتر از ۳ بعدی نیز به همین ترتیب، یک ابرصفحه یک زیرفضای $1-p$ بعدی است. هر ابرصفحه در فضای p بعدی به فرم زیر است:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

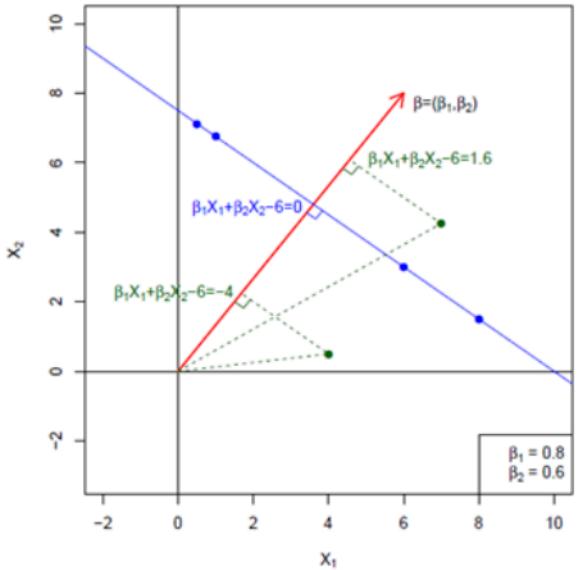
یعنی تمام نقاط $X = (X_1, X_2, \dots, X_p)$ که در معادله فوق صدق می کنند روی این ابرصفحه قرار دارند. در حالت خاص معادله یک ابرصفحه در فضای دو بعدی، به صورت زیر است:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

که معادله یک خط است.

اگر نقطه X به جای صدق کردن در معادله فوق در نامعادله زیر صدق کند

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p > 0$$



این نقطه در یک سمت از ابرصفحه قرار دارد و اگر نقطه X در نامعادله زیر صدق کند این نقطه در سمت دیگر ابرصفحه قرار دارد.

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p < 0$$

بنابراین هر ابرصفحه فضا را به دو قسمت تقسیم می‌کند. در شکل رو برو تصویری از یک ابرصفحه در فضای دو بعدی با معادله زیر را مشاهده می‌کنید:

$$1 + 2X_1 + 3X_2 = 0$$

ناحیه آبی رنگ مجموعه نقاطی است که

$$1 + 2X_1 + 3X_2 > 0$$

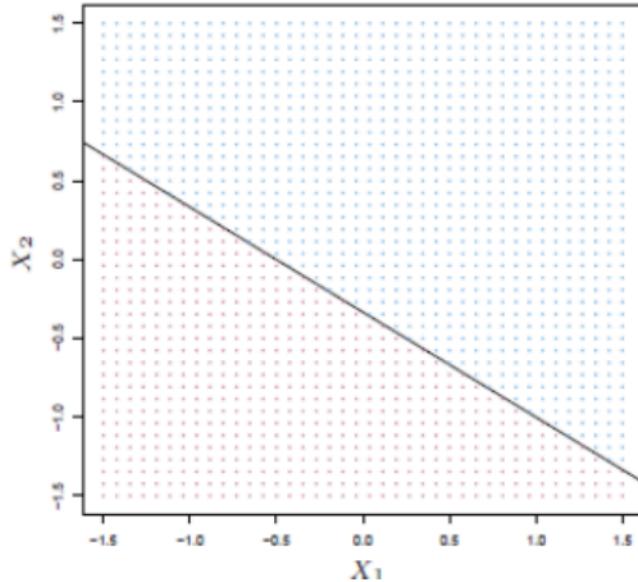
و ناحیه صورتی رنگ مجموعه نقاطی است که

$$1 + 2X_1 + 3X_2 < 0$$

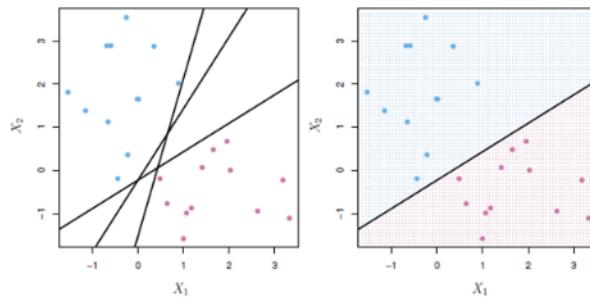
کلاس‌بندی با استفاده از ابرصفحه جداکننده: فرض کنید یک ماتریس $p \times n$ شامل n داده آموزشی p بعدی به ما داده شده است که سطرهای این ماتریس هر کدام مرتبط به یک نمونه آموزشی و به صورت زیر هستند:

$$X_1 = \begin{pmatrix} X_{11} \\ X_{12} \\ \vdots \\ X_{1p} \end{pmatrix}, \quad X_2 = \begin{pmatrix} X_{21} \\ X_{22} \\ \vdots \\ X_{2p} \end{pmatrix}, \dots, \quad X_n = \begin{pmatrix} X_{n1} \\ X_{n2} \\ \vdots \\ X_{np} \end{pmatrix}$$

هر نمونه آموزشی در یک کلاس قرار دارد. کلاس هر نمونه آموزشی را با $\{Y_1, Y_2, \dots, Y_n\} \in \{-1, 1\}$ نشان می‌دهیم که یکی از دو مقدار -1 یا 1 را می‌تواند بگیرد. همچنین یک نمونه تست داریم که $X^* = (X_1^*, \dots, X_p^*)$ را نشان می‌دهیم. هدف ما ساخت یک کلاس‌بند به نحوی است که بتوانیم داده تست را به درستی کلاس‌بندی کنیم. در این فصل کلاس‌بند را با استفاده از ابرصفحه جداکننده می‌سازیم. فرض کنید بتوانیم نقاط آموزشی را با یک ابرصفحه از هم جدا کنیم. مثلاً در تصویر سمت چپ مجموعه‌ای از ابرصفحه‌ها را می‌بینید که توانسته‌اند به خوبی نقاط قرمز ($y = -1$) و آبی ($y = 1$) را از هم جدا کنند.



شکل ۱: تصویر یک ابرصفحه در فضای دو بعدی



شکل ۲: مجموعه نقاط و ابرصفحه‌ها

ابرصفحه جداکننده دارای ویژگی‌های زیر است:

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > 0 \quad \text{اگر } y_i = 1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) < 0 \quad \text{اگر } y_i = -1$$

به طور معادل یعنی همیشه:

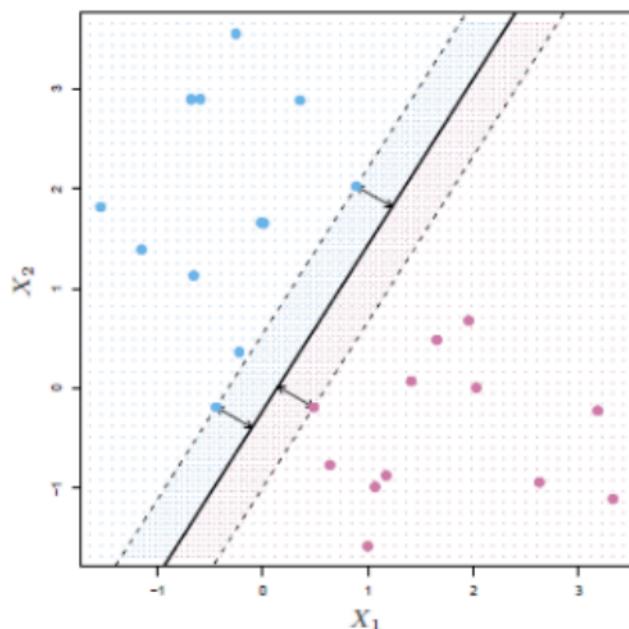
$$y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) > 0$$

اگر چنین ابرصفحه‌ای وجود داشته، می‌توان از آن به عنوان یک کلاس‌بند ساده استفاده کرد و هر داده جدید با توجه به آن که در کدام سمت ابرصفحه قرار می‌گیرد کلاس‌بندی می‌شود. نمونه‌ای از چنین کلاس‌بندی را در تصویر سمت راست شکل بالا مشاهده کنید. بدین معنا که برای داده تست تابع: $f(x^*) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ مثبت بوده آن را به کلاس +1 و اگر منفی بوده به کلاس -1 منسوب می‌کنیم.

همچنین مقدار عددی $f(x^*)$ می‌تواند نمایانگر درجه اطمینان ما از این انتساب باشد. هر چه $f(x^*)$ عددی دورتر از صفر باشد یعنی نقطه دورتر از ابرصفحه است و با اطمینان بیشتری آن را به کلاس نسبت می‌دهیم و هر چه $f(x^*)$ به صفر نزدیک باشد اطمینان کمتر به این انتساب داریم. واضح است که مرز تصمیم‌گیری برای چنین کلاس‌بندی به صورت خطی است.

کلاس‌بند بیشترین حاشیه

به طور کلی، وقتی بتوان داده‌ها را با یک ابرصفحه کلاس‌بندی کرد، بینهایت ابرصفحه با چنین ویژگی‌ای وجود دارند. زیرا با کمی بالا و پایین کردن یا چرخاندن ابرصفحه، می‌توان ابرصفحه‌های جدیدی را بدست آورد که هم‌چنان بتوانند بدون تماس داشتن با داده‌ای، کل داده‌ها را به خوبی کلاس‌بندی کنند. مثلا در تصویر سمت چپ شکل فوق سه تا از این ابرصفحه رسم شده است. حال سوالی که مطرح می‌شود این است که از بین این همه ابرصفحه، کدام را انتخاب کنیم. یکی از انتخاب‌های معقول، ابرصفحه با بیشترین فاصله از داده‌های آموزشی است که به آن ابرصفحه با بیشترین حاشیه یا ابرصفحه بهینه نیز می‌گویند. در واقع می‌توان فاصله عمودی هر نقطه تا ابرصفحه را حساب کرد. به دنبال یافتن ابرصفحه ای هستیم که بیشترین حاشیه را داشته باشد، یعنی بیشترین کمترین فاصله را با نقاط آموزشی دارد. بنابراین می‌توان انتظار داشت که بیشترین فاصله را با نقاط تست هم داشته باشد و از آن به عنوان کلاس‌بند استفاده کرد. گرچه کلاس‌بند با بیشترین حاشیه عمولاً عملکرد خوبی دارد، اما وقتی p بزرگ باشد ممکن است بیش برآذش (overfitting) رخ دهد. شکل روپرو کلاس‌بند بیشترین حاشیه را نشان می‌دهد. در واقع ابرصفحه بیشترین حاشیه، خط وسط پهن ترین نواری است که می‌توان در بین داده‌های دو کلاس جا داد. اگر به شکل روپرو دقت کنید، سه نقطه را می‌بینید که در مرز این نوار قرار گرفته‌اند. جایه‌جا شدن این نقاط منجر به جایه‌جا شدن نوار می‌شود، زیرا در صورتی که این نقاط دورتر یا نزدیکتر شوند پهن ترین نواری که در بین داده‌های دو کلاس قرار می‌گیرد تغییر می‌کند.



شکل ۳: مثال ابرصفحه بیشترین حاشیه

نحوه به دست آوردن کلاس‌بندی بیشترین حاشیه: فرض کنید n داده آموزشی در فضای \mathbb{R}^p داریم به صورت $(x_1, y_1), \dots, (x_n, y_n)$ با برچسب‌های کلاس‌های $\{+1, -1\}$ داده شده است. ابرصفحه بیشترین حاشیه جواب مسئله بهینه‌سازی زیر است:

$$\max_M M \quad \text{to subject} \quad \sum_{j=1}^p \beta_j = 1, \quad y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

شرط آخر (با فرض مثبت بودن M) بیانگر آن است که تمام داده‌های آموزشی باید در کلاس درست کلاس‌بندی شوند (زیرا برای اینکه داده‌ای درست کلاس‌بندی شود باید:

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq 0$$

تا مرز کلاس‌بندی بیشتر از M باشد. شرط اول در واقع محدودیتی اعمال نمی‌کند زیرا تنها بیان می‌کند که بردار نرمال عمود بر ابرصفحه باید یک است و اگر $\beta_0 + \beta_1x_1 + \cdots + \beta_px_p = 0$ آنگاه $k(\beta_0 + \beta_1x_1 + \cdots + \beta_px_p) = 0$ کلاس‌بند بیشتر حاشیه با حل این مسئله بهینه‌سازی بدبست می‌آید.

ویژگی‌های کلیدی:

- استفاده از هسته‌ها: هسته‌ها امکان نگاشت داده‌های غیرخطی به فضای ویژگی‌های بالاتر را فراهم می‌کنند.
- مقاومت در برابر نویز: SVM به دلیل استفاده از نسخه‌های نرم، نسبت به داده‌های پرت مقاوم‌تر است.
- کارایی در داده‌های با ابعاد بالا: بهویژه در مسائل پیچیده با ویژگی‌های زیاد، SVM عملکرد بهتری نسبت به LDA و رگرسیون لجستیک دارد.

مقایسه عملی بین روش‌ها

۱. نوع رابطه بین ویژگی‌ها و برجسب‌ها:
 - رگرسیون لجستیک بر اساس مدل‌های احتمال شرطی عمل می‌کند، اما در صورت وجود روابط پیچیده بین ویژگی‌ها و برجسب‌ها، LDA یا SVM ممکن است بهتر عمل کنند.
 - LDA از ترکیبات خطی برای تفکیک کلاس‌ها استفاده می‌کند و در صورت برقرار بودن فرض نرمال بودن داده‌ها، بسیار مؤثر است.
 - SVM با استفاده از هسته‌ها امکان تفکیک غیرخطی را فراهم می‌کند و این ویژگی بهویژه در داده‌های پیچیده مفید است.
۲. ابعاد داده‌ها:
 - SVM به دلیل توانایی استفاده از هسته‌ها برای نگاشت داده‌ها به فضای ویژگی‌های بالاتر، برای داده‌های با ابعاد بالا مناسب‌تر است.
 - LDA در داده‌ایی که ابعاد زیادی دارند، ممکن است به مشکل بخورد، مگر اینکه از روش‌هایی مانند کاهش ابعاد (PCA) استفاده شود.

۳. حساسیت به نویز:

- SVM بهویژه در نسخه‌های نرم، مقاوم‌تر از رگرسیون لجستیک و LDA است.
- رگرسیون لجستیک به دلیل تخمین احتمال شرطی، نسبت به نویز و داده‌های پرت حساس است، اما کمتر از LDA آسیب می‌بیند.

۴. پیچیدگی محاسباتی:

- SVM به دلیل استفاده از کرنل‌ها و بهینه‌سازی مبتنی بر فاصله، پیچیدگی محاسباتی بیشتری دارد.
- رگرسیون لجستیک و LDA از نظر پیچیدگی محاسباتی سبک‌تر هستند.

مثال‌های عملی

- رگرسیون لجستیک: برای پیش‌بینی اینکه آیا یک ایمیل اسپم است یا خیر، به خوبی عمل می‌کند زیرا داده‌ها معمولاً دارای روابط خطی ساده‌تری هستند.
- LDA: در یک مطالعه پزشکی که گروه‌های بیماران بر اساس ویژگی‌های بالینی نرمال طبقه‌بندی شده‌اند، LDA ممکن است بهترین انتخاب باشد.
- SVM: برای طبقه‌بندی تصویر، SVM به دلیل استفاده از هسته‌های غیرخطی برای تفکیک پیچیده میان دسته‌های تصویر کاربرد بسیار خوبی دارد.

نتیجه‌گیری

- رگرسیون لجستیک برای داده‌های کوچک یا غیرنرمال مناسب‌تر است.
- LDA در داده‌های بزرگ و با فرض‌های نرمالیتی عملکرد بهتری دارد.
- SVM به دلیل استفاده از هسته‌ها و توانایی تفکیک غیرخطی، برای داده‌های با ابعاد بالا و پیچیده بهترین گزینه است.

سوال ۲

مقایسه ماشین بردار پشتیبان (SVM) و رگرسیون بردار پشتیبان (SVR) تفاوت‌های اصلی این دو روش چیست و تحت چه شرایطی هر کدام بهتر عمل می‌کند؟

پاسخ

۱. ماشین بردار پشتیبان (SVM)

ماشین بردار پشتیبان یا SVM یک الگوریتم یادگیری نظارت شده است که برای حل مسائل طبقه‌بندی طراحی شده است. این مدل به دنبال یافتن یک ابرصفحه (hyperplane) است که داده‌ها را به بهترین شکل بین کلاس‌های مختلف تقسیک کند. ویژگی‌ها:

- SVM تلاش می‌کند که بیشترین فاصله (حاشیه) بین کلاس‌ها حفظ شود.
- نقاط نزدیک به این ابرصفحه به عنوان نقاط پشتیبان یا Support Vectors شناخته می‌شوند و نقشی کلیدی در ساخت مدل دارند.
- این مدل معمولاً در مسائلی مانند طبقه‌بندی ایمیل‌های اسپم، تشخیص تصاویر و تشخیص بیماری‌ها استفاده می‌شود.

۲. رگرسیون بردار پشتیبان (SVR)

رگرسیون بردار پشتیبان یا SVR برای حل مسائل رگرسیون و پیش‌بینی مقادیر پیوسته طراحی شده است. این مدل از همان مفاهیم استفاده می‌کند، اما به جای پیدا کردن ابرصفحه برای تقسیک کلاس‌ها، یک حاشیه ϵ تعیین می‌کند که داده‌ها باید در آن قرار بگیرند. ویژگی‌ها:

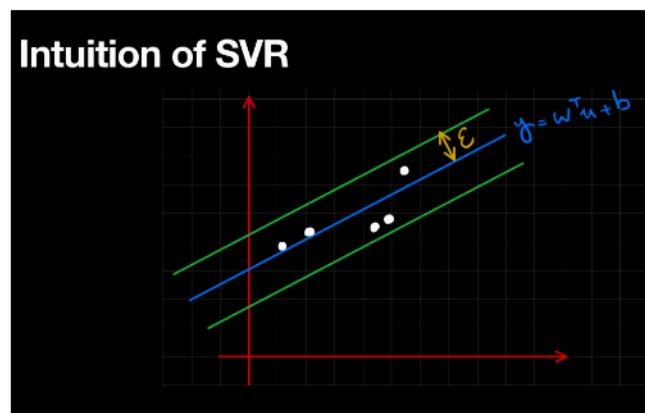
- SVR به دنبال یافتن تابعی است که بتواند داده‌ها را با کمترین خطای پیش‌بینی کند.
- نقاطی که خارج از حاشیه ϵ قرار دارند به عنوان نقاط پشتیبان شناخته می‌شوند و نقش مهمی در تعیین مدل دارند.
- این مدل در مسائلی مانند پیش‌بینی قیمت‌ها و تحلیل بازارهای مالی کاربرد دارد.

۳. فرمول دقیق SVR

Inputs: $\{x_i, y_i\} \rightarrow \text{dataset}$

$$\begin{cases} x_i \in \mathbb{R}^m \\ t_i \in \mathbb{R} \\ t_i \in \{-1, 1\} \end{cases}$$

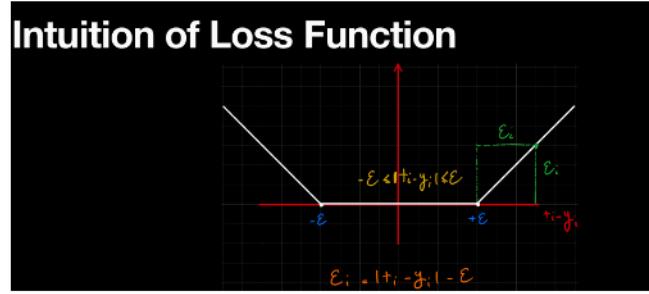
$$t_i \approx y_i = w^T x_i + b$$



Loss Function of SVR

$$L_\xi(t_i, y_i) = \begin{cases} \circ & \text{if } |t_i - y_i| \leq \xi, \\ ? & \text{otherwise.} \end{cases}$$

$$\begin{cases} t_i : \text{Correct Output} \\ y_i : \text{Model Output} \end{cases}$$



$$L_\xi(t_i, y_i) = \begin{cases} 0 & \text{if } |t_i - y_i| \leq \epsilon, \\ |t_i - y_i| - \epsilon & \text{otherwise.} \end{cases}$$

$$\epsilon_i = |t_i - y_i| - \epsilon$$

Operational Risk

$$R_{\text{emp}} = \frac{1}{N} \sum_{i=1}^N L_\xi(t_i, y_i) \longleftrightarrow \min \|w\| \longrightarrow \min \frac{1}{\sqrt{\gamma}} w^T w$$

با قیدهای زیر:

$$-\xi \leq t_i - y_i \leq \xi \implies -\xi - \xi_i^- \leq t_i - y_i \leq \xi + \xi_i^+$$

$$\begin{cases} \xi_i^+ + \xi_i^- = L_\xi(t_i, y_i) \\ \xi_i^+ \xi_i^- = 0 \\ \xi_i^+, \xi_i^- \geq 0 \end{cases}$$

New Problem Statement

$$R_{\text{emp}} = \frac{1}{N} \sum_{i=1}^N L_\xi(t_i, y_i) = \frac{1}{N} \sum_{i=1}^N (\xi_i^+ + \xi_i^-)$$

$$\min \frac{1}{\sqrt{\gamma}} w^T w + C \sum_{i=1}^N (\xi_i^+ + \xi_i^-)$$

با قیدهای زیر:

$$\alpha_i^+ \leftarrow -t_i + y_i + \xi + \xi_i^+ \geq 0$$

$$\alpha_i^- \leftarrow t_i - y_i + \xi + \xi_i^- \geq 0$$

$$\mu_i^+ \longleftarrow \xi_i^+ \geq \circ$$

$$\mu_i^- \longleftarrow \xi_i^- \geq \circ$$

Lagrange Multipliers

$$L_p(\alpha, w, b, \xi^+, \xi^-) = \frac{1}{2} w^T w + c \sum_{i=1}^N (\xi_i^+ + \xi_i^-) - \sum_{i=1}^N \alpha_i^+ (-t_i + y_i + \xi + \xi_i^+) - \sum_{i=1}^N \alpha_i^- (t_i - y_i + \xi + \xi_i^-) - \sum_{i=1}^N \mu_i^+ \xi_i^+ - \sum_{i=1}^N \mu_i^- \xi_i^-$$

Partial Differential

$$\begin{cases} \frac{\partial L_p}{\partial w} = \circ \rightarrow w = \sum_i (\alpha_i^+ - \alpha_i^-) x_i \\ \frac{\partial L_p}{\partial b} = \circ \rightarrow \circ = \sum_i \alpha_i^+ - \alpha_i^- \\ \frac{\partial L_p}{\partial \xi_i^+}, \frac{\partial L_p}{\partial \xi_i^-} = \circ \rightarrow \begin{cases} \alpha_i^+ + \mu_i^+ = c \\ \alpha_i^- + \mu_i^- = c \end{cases} \rightarrow \circ \leq \alpha_i, \mu_i \leq c \end{cases}$$

Dual Problem

$$\begin{aligned} \max L_D &= \frac{1}{2} \sum_i \sum_j (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) x_i^T x_j - \sum_i (\alpha_i^+ - \alpha_i^-) t_i + \xi \sum_i (\alpha_i^+ + \alpha_i^-) \\ &\quad \sum_i (\alpha_i^+ - \alpha_i^-) = \circ \\ &\quad \circ \leq \alpha_i^+ \leq c, \quad \circ \leq \alpha_i^- \leq c \end{aligned}$$

K.K.T Conditions

$$\begin{cases} \alpha_i^+ (-t_i + y_i + \xi + \xi_i^+) = \circ \\ \alpha_i^- (t_i - y_i + \xi + \xi_i^-) = \circ \\ \mu_i^+ \xi_i^+ = (c - \alpha_i^+) \xi_i^+ = \circ \\ \mu_i^- \xi_i^- = (c - \alpha_i^-) \xi_i^- = \circ \end{cases}$$

$$\left\{ \begin{array}{l} \alpha_i^+ = \circ \rightarrow \mu_i^+ = c \rightarrow \xi_i^+ = \circ \rightarrow -t_i + y_i + \xi \geq \circ \rightarrow t_i - y_i \leq \xi \\ \alpha_i^- = \circ \rightarrow \mu_i^- = c \rightarrow \xi_i^- = \circ \rightarrow t_i - y_i + \xi \geq \circ \rightarrow t_i - y_i \geq -\xi \\ \circ < \alpha_i^+ < c \rightarrow \circ < \mu_i^+ < c \rightarrow \xi_i^+ = \circ \rightarrow -t_i + y_i + \xi = \circ \rightarrow t_i - y_i = \xi \\ \circ < \alpha_i^- < c \rightarrow \circ < \mu_i^- < c \rightarrow \xi_i^- = \circ \rightarrow t_i - y_i + \xi = \circ \rightarrow t_i - y_i = -\xi \\ \alpha_i^+ = c \rightarrow \mu_i^+ = \circ \rightarrow \xi_i^+ > \circ \rightarrow t_i - y_i = \xi \\ \alpha_i^- = c \rightarrow \mu_i^- = \circ \rightarrow \xi_i^- > \circ \rightarrow t_i - y_i = -\xi \end{array} \right.$$

$$\left\{ \begin{array}{l} \alpha_i^+ = \alpha_i^- = \circ \\ \circ < \alpha_i^+ < c, \alpha_i^- = \circ \\ \circ < \alpha_i^- < c, \alpha_i^+ = \circ \\ \alpha_i^+ = c, \alpha_i^- = \circ \\ \alpha_i^- = c, \alpha_i^+ = \circ \end{array} \right.$$

Support Vector Regression

$$S = \{i | \circ < \alpha_i^+ < c \text{ or } \circ < \alpha_i^- < c\}$$

$$S = \{i | \circ < \alpha_i^+ + \alpha_i^- < c\}$$

$$\circ < \alpha_i^+ < c, \alpha_i^- = \circ \Rightarrow t_i - y_i = \xi \Rightarrow t_i = w^T x_i - b = \xi \Rightarrow t_i = y_i + \xi$$

$$\circ < \alpha_i^- < c, \alpha_i^+ = \circ \Rightarrow t_i - y_i = -\xi \Rightarrow t_i = y_i - \xi \Rightarrow t_i = w^T x_i + b + \xi$$

$$b = t_i - w^T x_i - \text{sign}(\alpha_i^+ - \alpha_i^-) \xi$$

فرمول دقیق SVR
تابع هدف SVR به حداقل رساندن عبارت زیر است:

$$\min_{\mathbf{w}, b} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*) \right)$$

با توجه به محدودیت ها:

$$y_i - (w^T x_i + b) \leq \epsilon + \xi_i \quad \forall i$$

$$(w^T x_i + b) - y_i \leq \epsilon + \xi_i^* \quad \forall i$$

$$\xi_i \xi_i^* \geq 0$$

w و b پارامترهای مدل هستند. ϵ حاشیه خطای مجاز در مدل است. ξ و ξ^* متغیرهایی هستند که به برخی از نقاط داده اجازه

می دهند خارج از حاشیه قرار گیرند. - C یک پارامتر تنظیمی است که مبادله بین پیچیدگی مدل و سستی مجاز را کنترل می کند. هدف SVR یافتن تابعی است که مقدار هدف را با حداقل انحراف از مقادیر واقعی پیش بینی می کند، در حالی که پیچیدگی مدل را کنترل می کند.

نتیجه‌گیری

SVM برای حل مسائل طبقه‌بندی و SVR برای مسائل رگرسیون استفاده می شود. انتخاب هر کدام از این مدل‌ها بستگی به نوع مسئله و داده‌های موجود دارد.

۱. دستگاه بردار پشتیبانی (SVM):

- هدف: SVM برای حل مسائل طبقه‌بندی طراحی شده است. هدف آن یافتن ابرصفحه‌ای است که نقاط داده را به بهترین نحو به کلاس‌های مجزا با حداقل حاشیه جدا می کند.

- عملیات: SVM بردارهای پشتیبانی (نردیک ترین نقاط داده به ابرصفحه) را شناسایی می کند و سعی می کند حاشیه بین کلاس‌های مختلف را به حداقل برساند.

- برنامه: SVM در کارهایی مانند طبقه‌بندی تصاویر، دسته‌بندی متن و سایر کارهای طبقه‌بندی باینری یا چند کلاسه استفاده می شود.

۲. پشتیبانی از رگرسیون برداری (SVR):

- هدف: SVR توسعه ای از SVM است که برای مشکلات رگرسیون طراحی شده است. هدف آن پیش بینی مقادیر پیوسته و در عین حال حفظ خطا در یک آستانه خاص (ϵ) است. - عملیات: SVR از همان مفهوم بردارهای پشتیبان استفاده می کند، اما به جای یافتن یک ابرصفحه که کلاس‌ها را جدا می کند، تابعی را پیدا می کند که نتایج پیوسته را با حداقل خطا پیش بینی می کند. - کاربرد: SVR در سناریوهایی مانند پیش بینی‌های مالی (به عنوان مثال، پیش بینی قیمت سهام) یا مشکلات رگرسیون در داده‌های علمی استفاده می شود.

سوال ۳

وقتی داده‌ها خطی جدایی پذیر نباشد، باید از کلاس بندی با مرزهای تصمیم‌گیری غیرخطی برای کلاس بندی داده‌ها استفاده کنیم. یکی از راه‌های کلاس بندی چنین داده‌هایی استفاده از توان دوم، سوم یا چندم متغیرهای ورودی است. این کار باعث می شود فضای ویژگی‌ها (feature space) گسترش یابد و یک کلاس بندی خطی در فضای گسترش یافته با مرزهای تصمیم‌گیری غیرخطی در فضای اصلی داشته باشیم. به عنوان مثال فضای ویژگی‌های X_1 و X_2 به فضای ویژگی‌های زیر گسترش می‌یابد:

$$(X_1, X_2, X_1^2, X_2^2, X_1 X_2)$$

در این صورت مرز تصمیم به صورت زیر است و بر حسب متغیرهای جدید خطی است:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

که این منجر به مرزهای تصمیم غیرخطی در فضای اصلی می شود.

یکی دیگر از راه‌هایی که برای دسته‌بندی چنین داده‌هایی استفاده می شود، استفاده از توابع هسته (kernel function) برای گسترش فضاست.

در بخش مربوط به ماشین بردار پشتیبان در مورد جواب مسئله بهینه‌سازی مطلبی گفته نشد. اما لازم است بدانید که جواب آن مسئله بهینه‌سازی مبتنی بر ضرب داخلی داده‌هاست. ضرب داخلی دو داده x_i, x_j به صورت زیر تعریف می شود:

$$\langle x_i, x_j \rangle = \sum_{j=1}^p x_{ij} x_{ij'}$$

به عبارت دقیق‌تر کلاس بند بردار پشتیبان به صورت زیر تعریف می‌شود:

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

برای حل این مسئله بهینه‌سازی، از تکنیک ضرایب لگرانژ استفاده می‌کنیم و آن را به فرم ساده‌تری تبدیل می‌کنیم. ما به دنبال یافتن یک ابرصفحه‌ای بودیم که بیشین فاصله را با نقاط هر کلاس داشته باشد. فاصله نقطه X از ابرصفحه تعیین می‌شود که $f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = 0$. بنابراین فاصله نقطه x_i از صفحه تصمیم به صورت زیر است:

$$\frac{y_i f(x_i)}{\|\beta\|} = \frac{y_i (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}{\|\beta\|}$$

حاشیه برابر است با فاصله عمود نزدیک‌ترین نقطه به صفحه، و ما می‌خواهیم طوری β و β_0 را بیابیم که این فاصله بیشینه شود. چنین پارامترهایی با حل مسئله زیر بدست می‌آیند:

$$\underset{\beta, \beta_0}{\operatorname{argmax}} \left(\frac{1}{\|\beta\|} \min_i [y_i (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})] \right)$$

جواب مستقیم به این مسئله دشوار است و باید این مسئله معادل تبدیل شود که ساده‌تر حل شود. برای این کار، کافی است به این نکته توجه کنیم که اگر مقیاسمان را با تبدیل $\beta \rightarrow k\beta$ عوض کنیم، فاصله نقطه x_i از صفحه تصمیم، یعنی $\frac{y_i f(x_i)}{\|\beta\|}$ عوض نمی‌شود. بنابراین از این ویژگی و آزادی در انتخاب مقیاس می‌توانیم استفاده کنیم و برای نقاطی که نزدیک‌ترین نقاط به ابرصفحه هستند، قرار دهیم: $y_i f(x_i) = 1$ و برای تمام نقاط، قرار دهیم:

$$y_i (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1, \quad i = 1, \dots, N$$

این نامساوی برای نقاطی که روی مرز هستند، به صورت تساوی برقرار است. حداقل یک نقطه وجود دارد که این تساوی صدق کند. بنابراین باید $\frac{1}{\|\beta\|}$ را بیشینه با به طور معادل $\frac{1}{2} \|\beta\|^2$ را کمینه کنیم. در ادامه یک ضریب λ برای راحتی کار اضافه می‌شود. بنابراین مسئله بهینه‌سازی مان را به صورت معادل زیر می‌توانیم بازنویسی کنیم:

$$\underset{\beta, \beta_0}{\operatorname{argmin}} \frac{1}{2} \|\beta\|^2$$

$$\text{s.t. } y_i (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1, \quad i = 1, \dots, N$$

برای حل این مسئله بهینه‌سازی، از تکنیک ضرایب لگرانژ استفاده می‌کنیم و آن را به فرم ساده‌تری تبدیل می‌کنیم. این مسئله به فرم زیر در می‌آید:

$$L(x, \alpha) = x^\top - \alpha(x - b) \quad \text{s.t.} \quad \alpha \geq 0$$

حال برگردیم به مسئله خودمان! مسئله بهینه سازی زیر:

$$\underset{\beta, \beta_0}{\operatorname{argmin}} \frac{1}{2} \|\beta\|^2$$

$$\text{s.t. } y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq 1, \quad i = 1, \dots, n$$

را می‌توان با ضرایب لگرانژ به فرم زیر تبدیل کرد:

$$L(\beta, \beta_0, \alpha) = \left(\frac{1}{2} \|\beta\|^2 - \sum_i \alpha_i [y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - 1] \right)$$

با مشتق گرفتن این عبارت بر حسب مقادیر بهینه زیر می‌رسیم:

$$\frac{\partial L}{\partial \beta} = \beta - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad \beta = \sum_i \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial \beta_0} = - \sum_i \alpha_i y_i = 0$$

با جایگذاری مقادیر بهینه در مسئله فوق، به عبارت زیر می‌رسیم:

$$\hat{L}(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

با شروط:

$$\alpha_i \geq 0 \quad \text{for } i = 1, \dots, N$$

$$\sum_i \alpha_i y_i = 0$$

که با توجه به ضرب داخلی‌های تابع هدف، معادله به فرم زیر است:

$$f(x) = \sum_{i=1}^n \alpha_i y_i \langle x, x_i \rangle + \beta_0$$

برای محاسبه ضرایب β_0 و α_i نیاز به محاسبه $\binom{n}{2}$ ضرب داخلی (بین هر دو جفت داده) داریم. برای یک داده جدید x ضرب داخلی آن را با تک تک داده‌های قبلی محاسبه کرده و هر یک را در α_i ضرب می‌کنیم و سپس در مورد داده‌ی جدید تصمیم می‌گیریم. با توجه به آنچه قبلاً گفته‌یم، کلاس بند تنها تحت تأثیر بردار پشتیبان است؛ می‌توان نتیجه گرفت که ضریب α_i تنها برای بردارهای پشتیبان ناصرف است.

بنابراین، اگر S مجموعه تمام اندیس‌های نقاط پشتیبان باشد، کلاس‌بند تابعی به فرم زیر دارد:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$$

سوال ۴

چهار هسته اصلی مورد استفاده در ماشین‌های بردار پشتیبان (SVM) را توضیح داده و مقایسه کنید، یعنی هسته‌های خطی، چندجمله‌ای، RBF و سیگموید. همچنین، اهمیت ثابت ۱ در هسته چندجمله‌ای را توضیح داده و تاثیر حذف یا تغییر این ثابت را بیان کنید.

پاسخ

مقایسه‌ی هسته‌های اصلی در ماشین بردار پشتیبان (SVM)

۱. هسته‌ی خطی (Linear Kernel)

در مدل ماشین بردار پشتیبان (SVM) و همچنین در رگرسیون بردار پشتیبان (SVR)، از توابع هسته‌ای (Kernel) استفاده می‌کیم تا بتوانیم داده‌ها را به فضایی با ابعاد بالاتر نگاشت کنیم و به این ترتیب، مرازهای تصمیم‌گیری را به صورت غیرخطی تنظیم کنیم. یکی از توابع هسته‌ای پرکاربرد، هسته چند جمله‌ای (Polynomial Kernel) است که به صورت زیر تعریف می‌شود:

هسته‌ی خطی ساده‌ترین نوع هسته است و به شکل زیر تعریف می‌شود:

$$K(x_i, x_j) = x_i^T x_j$$

این هسته برای داده‌ایی که به صورت خطی قابل جداسازی هستند، مناسب است.

۲. هسته‌ی چند جمله‌ای (Polynomial Kernel)

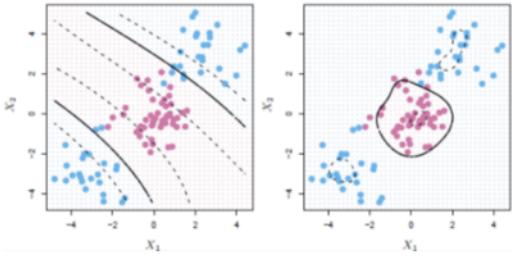
$$K(x_i, x_j) = \left(1 + \sum_{p=1}^d x_{ij} x'_{ij} \right)^d$$

- x_i و x_j بردارهای ورودی هستند.

- درجه‌ی چندجمله‌ای است.

هسته‌ی چند جمله‌ای برای افزایش قدرت تفکیک از طریق افزودن درجات چند جمله‌ای استفاده می‌شود. که در آن d درجه‌ی چند جمله‌ای است و میزان پیچیدگی مدل را تعیین می‌کند. این تابع هسته برای مقادیر مختلف $1 < d$ مرازهای پیچیده‌تری را نسبت به مدل خطی ارائه می‌دهد. در شکل زیر مشاهده می‌شود که چگونه مرازهای تصمیم‌گیری غیرخطی در فضای ویژگی ایجاد می‌شود.

استفاده از توابع هسته چند جمله‌ای با درجه $1 < d$ در کلاس بندهای بردار پشتیبان می‌تواند به انعطاف مرازهای تصمیم‌گیری کمک کند. گویا ما داده‌های را به فضای d بعدی بردۀ این و در آنجا کلاس بندی خطی تعریف کرده ایم که در فضای اصلی مراز تصمیم‌گیری غیرخطی دارد. نمونه‌ای از این کلاس بند را می‌توانید در تصویر سمت چپ شکل بالا است.



اهمیت ثابت ۱ در هسته‌ی چند جمله‌ای

در هسته‌ی چند جمله‌ای، ثابت ۱ برای اطمینان از این است که داده‌های با مقادیر کوچک یا منفی هم در فرآیند یادگیری لحاظ شوند استفاده می‌شود. حذف این ثابت ممکن است تأثیر منفی بر دقت مدل داشته باشد.

اهمیت ثابت ۱ در این فرمول:

اضافه کردن ثابت ۱ به ضرب داخلی $x_i^T x_j$ به جلوگیری از حذف داده‌ها در زمانی که بردارهای ورودی x_i و x_j به هم عمود هستند، کمک می‌کند. به عبارت دیگر، اگر این ثابت اضافه نشود و بردارهای ورودی x_i و x_j به هم عمود باشند، نتیجه ضرب داخلی برابر با صفر خواهد شد و در نتیجه کل مقدار تابع هسته برابر با صفر خواهد بود که می‌تواند اطلاعات مفیدی را در طبقه‌بندی از دست بدهد. اضافه کردن این ثابت به جلوگیری از از دست دادن اطلاعات در موارد خاص کمک می‌کند.

اثر حذف یا تغییر ثابت ۱:

- اگر ثابت ۱ حذف شود، در حالتی که $x_i^T x_j$ کوچک باشد، مقدار هسته نیز بسیار کوچک خواهد شد، که می‌تواند باعث عدم توانایی مدل در یادگیری بهینه شود. اگر این ثابت حذف شود، ممکن است برای داده‌ایی که به یکدیگر نزدیک نیستند، هسته مقدار خیلی کمی بگیرد.
- اگر مقدار این ثابت تغییر یابد (مثلاً بزرگتر شود)، مدل می‌تواند روابط غیرخطی قوی‌تری را بین داده‌ها یاد بگیرد اما در عین حال ممکن است منجر به بیش‌بازش (*overfitting*) شود.

۳. هسته‌ی گاوی (RBF Kernel)

هسته‌ی گاوی (Radial Basis Function - RBF) نیز یکی دیگر از توابع هسته‌ای پرکاربرد است که به صورت زیر تعریف می‌شود:

$$K(x_i, x_j) = \exp\left(-\gamma \sum_{j=1}^p (x_{ij} - x'_{ij})^2\right)$$

این تابع هسته برای داده‌ایی که به سختی در فضای اولیه جدایزیر هستند، عملکرد خوبی دارد و مزهای تصمیم‌گیری غیرخطی را به صورت انعطاف‌پذیرتری ایجاد می‌کند.

در نهایت، با استفاده از این توابع هسته‌ای، می‌توان داده‌های پیچیده را به شکلی بهتر دسته‌بندی کرد و مدل‌هایی با دقت بالاتر ایجاد نمود. که در آن γ پارامتری است که تأثیر فاصله‌ی بین نقاط داده را تنظیم می‌کند. نمونه‌ای از استفاده این تابع هسته در ماشین بردار پشتیبان را می‌توان در تصویر سمت راست شکل فوق ملاحظه کرد. این تابع هسته معمولاً عملکرد خوبی در جداسازی داده‌ها به دو کلاس دارد. علت این عملکرد خوب آن است که مثلاً فرض کنید می‌خواهیم یک داده تست $x^* = (x_1^*, \dots, x_p^*)$ را کلاس‌بندی کنیم. اگر داده تست از داده x_i دور باشد، فاصله اقلیدسی $\sum_{j=1}^p (x_{ij} - x_j^*)^2$ زیاد و بنابراین $K(x_i, x^*)$ کم خواهد شد. پس نقش x_i در تابع

$$f(x^*) = \beta_0 + \sum_{i \in S} \alpha_i K(x_i, x^*)$$

کم خواهد بود. بنابراین تابع هسته شعاعی به صورت محلی عمل می‌کند. به این معنا که هر داده‌ای که به داده تستمان نزدیک‌تر باشد، تأثیر بیشتری در تصمیم‌گیری ما دارد.

۴. هسته سیگموئید (Sigmoid Kernel)

هسته سیگموئید یکی از توابع هسته‌ای است که در ماشین‌های بردار پشتیبان (SVM) به کار می‌رود. این هسته بیشتر زمانی استفاده می‌شود که داده‌ها دارای مزه‌های تصمیم‌گیری غیرخطی باشند. هسته سیگموئید بر اساس تابع غیرخطی سیگموئید ($tanh$) عمل می‌کند و از آن در شبکه‌های عصبی مصنوعی نیز استفاده می‌شود. این هسته مشابه نورون‌های مصنوعی در شبکه‌های عصبی عمل می‌کند. فرمول ریاضی این تابع به صورت زیر است:

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$$

که در این فرمول:

- γ پارامتری است که باید تنظیم شود.

- r یک ثابت است که بایاس تابع را تعیین می‌کند.

این تابع در بعضی مواقع عملکرد مشابه با شبکه‌های عصبی را در یادگیری ماشین ارائه می‌دهد و می‌تواند برای داده‌هایی که دارای رابطه پیچیده‌ای بین ویژگی‌ها هستند، مناسب باشد.

سوال ۵

مقایسه و مقابله الگوریتم‌های CatBoost، LightGBM، XGBoost با تمرکز بر ویژگی‌های منحصر به فرد آن‌ها، بهینه‌سازی‌ها و بهبود عملکردها، بهویژه در پردازش ویژگی‌های دسته‌ای و داده‌های بزرگ. همچنین، توضیح دهید که چگونه این روش‌ها بر درخت‌های تصمیم سنتی بهبود یافته‌اند و چه بهبودهای کلیدی نسبت به درخت‌های تصمیم ارائه می‌دهند؟

پاسخ

XGBoost، LightGBM CatBoost، مقایسه

الگوریتم‌های LightGBM CatBoost، XGBoost و Gradient Boosting (Градиентный бустинг) همگی از مدل‌های گرادیان بوستینگ (Gradient Boosting) استفاده می‌کنند که در مقایسه با درخت‌های تصمیم سنتی بهبودهای قابل توجهی دارند. در ادامه، ویژگی‌های منحصر به فرد هر کدام از این الگوریتم‌ها به تفصیل آمده است.

(Gradient Boosting) مدل‌های گرادیان بوستینگ

مدل‌های گرادیان بوستینگ (Gradient Boosting) یکی از تکنیک‌های یادگیری ماشین با نظارت هستند که برای حل مسائل رگرسیون و طبقه‌بندی به کار می‌روند. ایده اصلی این روش بر پایه‌ی ساخت یک مدل قوی از ترکیب چندین مدل ضعیف است. در این تکنیک، مدل‌ها به صورت سریالی (Sequential) ساخته می‌شوند و هر مدل بعدی تلاش می‌کند تا خطاهای مدل قبلی را کاهش دهد.

مراحل اصلی گرادیان بوستینگ:

۱. آموزش مدل اولیه: ابتدا یک مدل ساده یا ضعیف (معمولًاً یک درخت تصمیم کوچک) آموزش داده می‌شود. این مدل ممکن است عملکرد قابل توجهی نداشته باشد.
۲. محاسبه خطای پس از آموزش مدل اولیه، خطاهای آن (که می‌توانند به عنوان باقیمانده‌ها تعریف شوند) محاسبه می‌شوند. این خطاهای نشان می‌دهند که مدل در پیش‌بینی داده‌های آموزشی تا چه حد دقیق نبوده است.
۳. آموزش مدل بعدی: در مرحله بعد، یک مدل جدید آموزش داده می‌شود که هدف آن کاهش خطاهای مدل قبلی است. این مدل جدید روی باقیمانده‌های مدل قبلی تمرکز می‌کند تا آن‌ها را اصلاح کند.
۴. تکرار فرآیند: این فرآیند تکرار می‌شود و هر مدل جدید خطاهای باقیمانده مدل قبلی را کاهش می‌دهد. در نهایت، خروجی نهایی با ترکیب وزن‌دار تمامی مدل‌ها به دست می‌آید.

ویژگی‌های کلیدی:

- کاهش خطای گرادیان بوستینگ به طور مداوم خطاهای پیش‌بینی را با هر مدل جدید کاهش می‌دهد.
- کارایی بالا: به ویژه برای مجموعه داده‌های بزرگ و پیچیده بسیار کارا است.
- انعطاف‌پذیری: می‌تواند برای هر دو مسئله رگرسیون و طبقه‌بندی به کار رود.

این روش نسبت به درخت‌های تصمیم سنتی کارآمدتر است و نتایج دقیق‌تری به دست می‌دهد، زیرا خطاهای مدل‌های ضعیفتر به تدریج تصحیح می‌شوند.

الگوریتم CatBoost در یادگیری ماشین

CatBoost یکی از الگوریتم‌های پیشرفتی گرادیان بوستینگ است که توسط Yandex توسعه داده شده و برای کار با داده‌های دسته‌بندی شده (categorical data) هینه‌سازی شده است. یکی از چالش‌های اصلی در پردازش داده‌های دسته‌بندی، تبدیل آنها به یک فرم قابل استفاده برای مدل‌های یادگیری ماشین است. CatBoost این مشکل را با تبدیل‌های کدگذاری دسته‌بندی هوشمندانه حل می‌کند و نیازی به تکنیک‌های معمول مانند *one-hot encoding* ندارد.

ویژگی‌های کلیدی CatBoost

- پردازش ویژگی‌های دسته‌ای: CatBoost به طور ویژه‌ای برای پردازش ویژگی‌های دسته‌ای (Categorical Features) بهینه شده است. به جای تبدیل دسته‌ها به اعداد پیوسته یا استفاده از کدگذاری *One-Hot Encoding*، از تکنیک‌های آماری برای تبدیل دسته‌ها استفاده می‌کند که موجب کاهش بیش‌برازش (Overfitting) می‌شود.
- سازگاری با داده‌های غیرمتعادل: CatBoost در برابر داده‌های غیرمتعادل مقاوم‌تر است و به طور بهینه با داده‌هایی که دارای مقادیر مختلف دسته‌ها هستند کار می‌کند.
- کاهش overfitting: با استفاده از **Boosting Ordered**، که در آن ترتیب داده‌ها حفظ می‌شود، خطر overfitting کاهش می‌یابد.
- کارایی بالا: این الگوریتم از تکنیک‌های بهینه‌سازی حافظه و پردازش موازی استفاده می‌کند تا عملکرد بهتری نسبت به روش‌های مشابه داشته باشد.

CatBoost فرمول‌های ریاضی

فرمول پایه‌ای CatBoost مشابه دیگر الگوریتم‌های گرادیان بوستینگ است. هدف الگوریتم بهینه‌سازی یک تابع ضرر (loss) function است که به صورت تجمعی در هر مرحله بهبود می‌یابد:

$$L(y_i, f(x_i)) = \sum_{i=1}^n l(y_i, f(x_i))$$

در اینجا:

- y_i برچسب هدف است.
- $f(x_i)$ پیش‌بینی مدل در مرحله‌ی فعلی است.
- l تابع ضرر است که می‌تواند از انواع مختلفی مانند \log loss $RMSE$ یا $loss$ باشد.

مراحل بهروزرسانی مدل

در هر مرحله، مدل بهروزرسانی می‌شود تا باقی مانده‌های مرحله قبلی به حداقل برسند. به عبارت دیگر، مدل‌های جدید تلاش می‌کنند تا خطاهای مدل قبلی را اصلاح کنند. برای هر مرحله t داریم:

$$f_{t+1}(x) = f_t(x) + \eta \cdot \sum_{i=1}^n g_i(x_i)$$

در اینجا:

- $f_t(x)$ مدل فعلی است.
- $g_i(x_i)$ گرادیان تابع ضرر نسبت به مدل فعلی است.
- η نرخ یادگیری است که تعیین می‌کند مدل جدید چقدر بر اساس خطاهای قبلی تنظیم شود.

CatBoost همچنین از permutations random برای پردازش داده‌های دسته‌بندی شده و کاهش خطاهای آماری استفاده می‌کند که این موضوع آن را برای داده‌های پیچیده بسیار مناسب می‌کند.

الگوریتم LightGBM در یادگیری ماشین

یک الگوریتم پیشرفته در زمینه گرادیان بوستینگ است که توسط Microsoft توسعه یافته و برای کار با مجموعه داده‌های بزرگ و پیچیده بسیار بهینه‌سازی شده است. این الگوریتم ویژگی‌هایی نظیر مقیاس‌پذیری بالا، سرعت اجرا و دقت مناسب دارد. LightGBM به طور خاص برای داده‌های با تعداد زیادی ویژگی و نمونه طراحی شده است و می‌تواند با تکنیک‌های مختلفی مانند نمونه‌گیری مبتنی بر histogram به خوبی با مجموعه داده‌های بزرگ کار کند.

ویژگی‌های کلیدی LightGBM

• پردازش سریع و بهینه داده‌های بزرگ: LightGBM با استفاده از استراتژی Leaf-Wise Tree Growth به جای Level-Wise، رشد درخت‌ها را بهینه کرده و باعث می‌شود تا در داده‌های بزرگ و پیچیده سرعت آموزش افزایش یابد.

- تقسیم‌بندی مبتنی بر هیستوگرام: LightGBM از الگوریتم Histogram-Based Decision Trees برای تقسیم داده‌ها استفاده می‌کند که با کاهش فضای جستجو و دسته‌بندی داده‌ها، سرعت آموزش را بیشتر و مصرف حافظه را بهینه می‌کند.
- حمایت از ویژگی‌های دسته‌بندی شده: این الگوریتم به صورت مستقیم از ویژگی‌های دسته‌ای پشتیبانی کرده و نیازی به کدگذاری پیچیده مثل Encoding One-Hot را حذف می‌کند.
- پردازش موازی: LightGBM قابلیت پردازش موازی را دارد که باعث می‌شود تا از تمام منابع پردازشی سیستم برای تسريع در آموزش مدل استفاده کند.
- تقسیم عمیق‌تر و بهینه درخت‌ها: برخلاف روش‌های دیگر که در هر مرحله کل درخت ساخته می‌شود، LightGBM درخت‌ها را به صورت عمودی و عمیق‌تر می‌سازد که بهبود عملکرد بهتری در داده‌های پیچیده ارائه می‌دهد.

فرمول‌های ریاضی LightGBM

الگوریتم LightGBM از گرادیان بوستینگ پیروی می‌کند که هدف آن کاهش خطای تابع ضرر در هر مرحله است. فرمول اصلی برای بهروزرسانی مدل به این صورت است:

$$f_{t+1}(x) = f_t(x) + \eta \cdot \sum_{i=1}^n g_i(x_i)$$

که در آن:

- $f_t(x)$ مدل فعلی است.
- $g_i(x_i)$ گرادیان تابع ضرر نسبت به مدل فعلی است.
- η نرخ یادگیری است که مقدار تغییرات مدل در هر مرحله را تعیین می‌کند.

در LightGBM، تقسیم داده‌ها برای یافتن بهترین *split* با بیشینه‌سازی **gain** انجام می‌شود. این *gain* به صورت زیر تعریف می‌شود:

$$Gain = \frac{1}{2} \left(\frac{G_L^*}{H_L + \lambda} + \frac{G_R^*}{H_R + \lambda} - \frac{(G_L + G_R)^*}{H_L + H_R + \lambda} \right) - \gamma$$

که در آن:

- G_R و G_L مجموع گرادیان‌های سمت چپ و راست تقسیم هستند.
- H_R و H_L مجموع هسیان‌های سمت چپ و راست تقسیم هستند.
- λ یک پارامتر منظم‌کننده است که از *overfitting* جلوگیری می‌کند.
- γ حداقل سود لازم برای ایجاد یک تقسیم جدید است.

نتیجه‌گیری

یک انتخاب عالی برای داده‌های بزرگ و پیچیده است که با استفاده از تکنیک‌های بهینه‌سازی مانند تقسیم هیستوگرامی، پردازش موازی، و استفاده از روش‌های خاص برای داده‌های دسته‌بندی شده، بهبودهای چشمگیری در کارایی و سرعت ارائه می‌دهد.

الگوریتم XGBoost در یادگیری ماشین

که مخفف Extreme Gradient Boosting است، یک روش قدرتمند برای یادگیری تقویتی گرادیان است که با هدف بهبود سرعت و دقت نسبت به مدل‌های گرادیان بوستینگ معمولی توسعه یافته است.

ویژگی‌های کلیدی XGBoost

- منظم‌سازی برای جلوگیری از بیش‌پرازش (Overfitting): XGBoost از منظم‌سازی L_1 و L_2 استفاده می‌کند که موجب جلوگیری از بیش‌پرازش و افزایش تعمیم‌پذیری مدل در مواجهه با داده‌های جدید می‌شود.
- تقسیم دقیق برگ‌ها: XGBoost از روش تقسیم برگ‌ها به صورت دقیق و عمیق‌تر استفاده می‌کند که باعث بهینه‌سازی عملکرد مدل و کاهش خطای آن می‌شود.
- پردازش موازی: این الگوریتم از پردازش موازی استفاده می‌کند که باعث افزایش سرعت آموزش مدل می‌شود و امکان کار با داده‌های بزرگ را فراهم می‌آورد.
- پشتیبانی از حافظه محدود: XGBoost به طور خاص برای کار با داده‌های بزرگ و محدودیت‌های حافظه طراحی شده است. مصرف بهینه حافظه و کاهش نیاز به رم، امکان پردازش سریع داده‌های حجمی را فراهم می‌کند.
- تعامل با حافظه: این الگوریتم با مصرف بهینه حافظه، باعث می‌شود که بتواند با داده‌های بسیار بزرگ به خوبی کار کند و در شرایطی که حافظه محدود است، عملکرد بهتری داشته باشد.

فرمول ریاضی

فرمول اصلی تابع هدف در XGBoost به شکل زیر است:

$$\mathcal{L}(\phi) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

که در آن:

- $l(y_i, \hat{y}_i)$ تابع خطای خطا است (مثل تابع مربع خطای یا تابع لاجیت).
- $\Omega(f_k)$ تابع منظمی است که پیچیدگی مدل را کنترل می‌کند.
- K تعداد درخت‌های بوست شده است.

همچنین منظم‌سازی در XGBoost به شکل زیر است:

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_j w_j^2$$

که در آن:

- T تعداد برگ‌های درخت است.

• وزن‌های مربوط به برگ‌ها هستند.

• γ و λ پارامترهای منظم‌سازی هستند.

نتیجه‌گیری

XGBoost به دلیل استفاده از روش‌های بهینه‌سازی محاسباتی و قابلیت کنترل پیچیدگی مدل، یک انتخاب بسیار محبوب در یادگیری ماشین برای داده‌های بزرگ و پیچیده است.

درخت‌های تصمیم سنتی و بهبودهای الگوریتم‌های گرادیان بوستینگ

درخت‌های تصمیم سنتی از روش‌های ساده‌تری برای تقسیم داده‌ها استفاده می‌کنند. درخت‌های تصمیم سنتی، که از الگوریتم‌هایی مثل CART (Classification and Regression Trees) استفاده می‌کنند، دارای مزایای بسیاری از جمله سادگی، تفسیرپذیری و سرعت در آموزش هستند. با این حال، این مدل‌ها در معرض مشکلاتی مانند بیش‌برازش (overfitting) قرار دارند و به تنها‌بی دقت پایینی دارند. الگوریتم‌های گرادیان بوستینگ مانند XGBoost، LightGBM و CatBoost با ترکیب چندین درخت ضعیف (درخت‌های تصمیم) و استفاده از تقویت گرادیان، عملکرد مدل‌ها را بهبود می‌دهند. در ادامه بهبودهای کلیدی نسبت به درخت‌های تصمیم سنتی آورده شده است:

• **تقویت مدل‌های ضعیف:** به جای استفاده از یک درخت بزرگ و پیچیده، گرادیان بوستینگ از مجموعه‌ای از درخت‌های تصمیم کوچک استفاده می‌کند که هر کدام خطاهای مدل قبلی را اصلاح می‌کنند. این فرآیند به بهبود تعمیم‌پذیری کمک می‌کند.

• **بهبود دقت:** با ترکیب چندین درخت، مدل‌های گرادیان بوستینگ توانایی بیشتری در شناسایی الگوهای پیچیده در داده‌ها دارند، در حالی که مدل‌های درخت تصمیم به تنها‌بی ممکن است الگوها را به درستی شناسایی نکنند.

• **کاهش بیش‌برازش:** الگوریتم‌های پیشرفته (XGBoost, LightGBM, CatBoost) از منظم‌سازی Gradient Boosting (مثل L_1 و L_2) استفاده می‌کنند که به کاهش پیچیدگی بیش از حد مدل‌ها و جلوگیری از بیش‌برازش کمک می‌کند.

• **پردازش موازی:** یکی از مشکلات درخت‌های سنتی، زمان آموزش طولانی برای داده‌های بزرگ است. اما الگوریتم‌هایی مثل XGBoost و LightGBM با استفاده از پردازش موازی این مشکل را حل کرده‌اند و این افزایش سرعت محاسبات استفاده می‌کنند.

• **مدیریت داده‌های دسته‌ای:** برخلاف درخت‌های تصمیم که به کدگذاری دستی نیاز دارند، CatBoost به‌طور طبیعی ویژگی‌های دسته‌ای را مدیریت می‌کند.

فرمول ریاضی در بهبود مدل‌های گرادیان بوستینگ نسبت به درخت‌های تصمیم سنتی

در درخت‌های تصمیم سنتی، تقسیم هرگره با استفاده از معیارهایی مانند Gini impurity Entropy یا انجام می‌شود.

۱. معیار Gini

معیار Gini که برای اندازه‌گیری ناهمگنی داده‌ها استفاده می‌شود، به صورت زیر تعریف می‌شود:

$$Gini = 1 - \sum_{i=1}^C p_i^2$$

که در آن:

- تعداد کلاس‌ها است.

- احتمال تعلق نمونه به کلاس i است.

۲. معیار Entropy

معیار Entropy برای سنجش عدم قطعیت یا بی‌نظمی داده‌ها استفاده می‌شود و به صورت زیر محاسبه می‌شود:

$$Entropy = - \sum_{i=1}^C p_i \log_2(p_i)$$

که در آن:

- تعداد کلاس‌ها است.

- احتمال تعلق نمونه به کلاس i است.

۳. کاهش اطلاعات (Information Gain)

برای تعیین بهترین تقسیم، کاهش اطلاعات (Gain Information) محاسبه می‌شود. این معیار نشان‌دهنده کاهش در آنتروپی قبل و بعد از تقسیم داده‌ها است و به صورت زیر تعریف می‌شود:

$$IG(T, A) = Entropy(T) - \sum_{v \in Values(A)} \frac{|T_v|}{|T|} Entropy(T_v)$$

که در آن:

- مجموعه داده اصلی است.

- A ویژگی مورد بررسی برای تقسیم است.

- T_v زیرمجموعه‌ای از داده‌ها است که $A = v$ است.

- $|T|$ تعداد کل نمونه‌ها در مجموعه T است.

- $|T_v|$ تعداد نمونه‌ها در زیرمجموعه T_v است.

۴. معیار CART برای رگرسیون

در درخت‌های تصمیم برای رگرسیون، معیار خطای مربعی (Squared Error) به عنوان معیار تقسیم استفاده می‌شود که به صورت زیر محاسبه می‌گردد:

$$SE = \sum_{i=1}^n (y_i - \hat{y})^2$$

که در آن:

- y_i مقدار واقعی است.
- \hat{y} مقدار پیش‌بینی شده است.
- n تعداد نمونه‌ها است.

در گرادیان بوستینگ، مدل‌ها به صورت تکراری با بهروزرسانی گرادیان تابع ضرر ($L(y_i, f(x_i))$) آموزش داده می‌شوند. فرمول تابع ضرر به شکل زیر است:

$$\mathcal{L}(\phi) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

که در آن:

- $l(y_i, \hat{y}_i)$ تابع ضرر (مثلاً خطای مربعی) است.
- $\Omega(f_k)$ تابع منظمی برای جلوگیری از پیچیدگی بیش از حد مدل است.

در اینجا، هدف الگوریتم یافتن بهینه‌سازی تابع ضرر است که در هر مرحله خطاهای مدل قبلی را اصلاح کند. برای مثال در **XGBoost** از روش زیر برای محاسبه سود هر تقسیم (*gain*) استفاده می‌شود:

$$Gain = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma$$

که در آن:

- G_L و G_R مجموع گرادیان‌های سمت چپ و راست هستند.
- H_L و H_R مجموع هسیان‌های سمت چپ و راست هستند.
- λ پارامتر منظم‌سازی است که به جلوگیری از بیش‌برازش کمک می‌کند.
- γ حداقل سود لازم برای ایجاد یک تقسیم جدید است.

نتیجه‌گیری

در مقایسه با درخت‌های تصمیم سنتی، الگوریتم‌های گرادیان بوستینگ نظری **XGBoost** و **LightGBM** با ترکیب مدل‌های ضعیف، استفاده از منظم‌سازی، و بهره‌گیری از پردازش موازی توانسته‌اند کارایی و دقت بسیار بیشتری را در مسائل پیچیده یادگیری ماشین ارائه دهند.

سوال ۶

استراتژی‌های رایج برای جلوگیری از بیش‌بازش در درخت‌های تصمیم را توضیح دهید. این روش‌ها چگونه به بهبود تعمیم مدل کمک می‌کنند؟

پاسخ

بیش‌بازش (**Overfitting**) در درخت‌های تصمیم زمانی رخ می‌دهد که مدل به قدری پیچیده شود که نویز و ناهنجاری‌های داده‌های آموزشی را نیز یاد بگیرد. برای جلوگیری از این مشکل، چندین استراتژی متداول وجود دارد که باعث بهبود تعمیم مدل می‌شوند. زمانی که درخت بزرگی تولید شود که بر روی نمونه‌های آموزشی، جواب خوبی برگرداند اما بر داده‌های نمونه‌های تست، کارایی پایینی داشته باشد این امر به دلیل پیچیده شدن درخت (عمق زیاد و استفاده از انشعاب‌های متعدد) است. یک درخت کوچکتر با تعداد انشعاب‌های کمتر ممکن است مقدار کمی بایاس داشته باشد، اما به شدت واریانس کمتری بر روی داده‌های تست خواهد داشت.

۱. هرس درخت (Pruning)

هرس درخت به دو صورت هرس پیش‌هنگام (Post-pruning) و هرس پس‌هنگام (Pre-pruning) انجام می‌شود:

- **هرس پیش‌هنگام:** در این روش، درخت قبل از اینکه به شکل کامل رشد کند متوقف می‌شود، و گره‌هایی که دارای اطلاعات کمی هستند حذف می‌شوند.
- **هرس پس‌هنگام:** این روش بعد از اینکه درخت به طور کامل رشد کرد اعمال می‌شود و گره‌هایی که بهبود زیادی در پیش‌بینی ندارند حذف می‌شوند.

هرس درخت کمک می‌کند تا مدل ساده‌تر شود و از بیش‌بازش جلوگیری می‌کند.

یک ایده برای هرس کدن درخت این است که از بالا به پایین حرکت کنیم و هر انشعابی که RSS نسبت به انشعاب قبلی مقدار کمی (کمتر از یک آستانه) کاهش یافته بود، در شاخه‌های آن را از درخت حذف کنیم. این استراتژی چندان مناسب نیست زیرا نوعی نگرش (short sighting) است. چون ممکن است یک انشعاب چندان خوب نباشد اما در شاخه‌های آن شامل انشعاب‌های خیلی خوبی باشد. یک استراتژی بهتر می‌تواند تولید یک درخت T بزرگ و بعد هرس کردن آن باشد. به طور شهودی ما به دنبال زیر درختی هستیم که دارای کمترین نرخ خطأ بر روی داده‌های تست باشد. که نرخ خطای تست را می‌توانیم با cross-validation error تخمین بزنیم. چک کردن تمام زیر درخت‌های یک درخت و محاسبه cross-validation error برای همه آنها زمان بر است. در عوض، ما از روش زیر برای هرس کردن درخت استفاده می‌کنیم:

دنباله‌ای از درخت‌ها که با پارامتر نامنفی α اندیس گذاری شده اند را در نظر می‌گیریم. به ازای هر مقدار که به ازای هر مقدار α زیر درخت $T \subset T$ ای وجود دارد که مقدار تابع هزینه هرس زیر به ازای آن کمینه می‌شود:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

که $|T|$ تعداد برگ‌ها در زیر درخت T است، پارامتر α ، تعدیل کننده‌ای بین میزان پیچیدگی درخت و میزان وابستگی آن به داده‌های آموزشی است. وقتی $\alpha = 0$ زیر درخت T برابر همان درخت اولیه است و هر چه مقدار α افزایش یابد، هرس بیشتری اعمال می‌شود. هر چه درخت بزرگ‌تر می‌شود میزان جریمه تابع فوق به ازای برگ‌ها، بیشتر می‌شود و بنابراین تابع فوق به ازای زیردرخت‌های کوچکتری کمینه می‌شود. خلاصه این روش در الگوریتم زیر آمده است:

الگوریتم

۱. با استفاده از روش تقسیم‌سازی دوگانه بازگشتی (Recursive binary splitting)، درخت بزرگی چون T بر حسب نمونه‌های آموزشی با شرط توقف چون وجود کمتر از ۵ نمونه در هر دسته بسازیم.
۲. تابع هزینه هرس (Cost complexity pruning) را برای درخت محاسبه کنیم تا دنباله‌ای از درخت‌ها بر حسب مقادیر α بدست آوریم.
۳. با استفاده از k -fold cross validation بهترین مقدار α را محاسبه کنیم.
 - a. مراحل ۱ و ۲ را بر روی $\frac{k-1}{k}$ نمونه‌های آموزشی (با حذف k -امین بخش) محاسبه کنیم.
 - b. میانگین مجوزدهی خطای را بر روی دسته k ام بدست آورده و این اعداد را میانگین بگیریم و α ای را انتخاب کنیم که کمترین میانگین خطای (MSE) را داشته باشد.
۴. به مرحله ۲ برگشته و زیر درخت متناظر با α انتخاب شده را به عنوان خروجی برگردانیم.
ایده روانتر الگوریتم فوق بصورت زیر است:

۱. با استفاده از روش تقسیم سازی دوگانه بازگشتی (Recursive binary splitting)، درخت بزرگی چون T بر حسب نمونه‌های آموزشی با شرط توقفی چون وجود کمتر از ۵ نمونه در هر دسته بسازیم.

۲. را بکار می‌بریم تا یک α بینه را بدست آوریم. به ازای هر α ، دلخواه داده‌های آموزشی را به K – fold cross validation قسمت تقسیم می‌کنیم. روی هر $\frac{k-1}{k}$ بخش از نمونه‌های آموزشی مانند مرحله ۱، یک درخت بزرگ T بسازیم. سپس هر نمونه از نمونه‌های در دست T را ارزیابی کنیم تا شرایط توقفی چون وجود کمتر از ۵ نمونه در هر دسته بسازیم. در ادامه تابع هزینه هرس برای درخت به این مقدار زیر کمینه شود:

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

۳. بر روی هر یک از $\frac{k}{k}$ باقی مانده‌های داده‌های آموزشی (یعنی k -fold CV validation set) خطای MSE را بدست آوریم. بین ترتیب خطاهای زیر بدست می‌آید:

$$MSE_1, MSE_2, \dots, MSE_k$$

حال خطای MSE_α منتظر از α که میانگین این خطاهای می‌باشد به صورت زیر بدست می‌آید:

$$MSE_{\alpha} = \frac{MSE_1 + \dots + MSE_k}{K}$$

۴. یک α ای را انتخاب کرده که MSE_{α} وابسته به کمیته باشد.

۵. به ازای این α ، زیردرخت T از T ساخته شده در مرحله (۱) را بدست آورده که تابع هزینه هرس (Cost complexity pruning) برای آن کمیته باشد.

۲. تعیین عمق ماکسیمم درخت (Max Depth)

یکی از روش‌های رایج برای کنترل پیچیدگی درخت‌های تصمیم، تعیین یک عمق ماکسیمم برای درخت است. با محدود کردن عمق درخت، از ایجاد شاخه‌های پیچیده و بلند که منجر به یادگیری بیش از حد داده‌های آموزشی می‌شود، جلوگیری می‌گردد.

۳. استفاده از حداقل نمونه‌ها برای تقسیم (Min Samples Split)

این روش تعیین می‌کند که یک گره تنها در صورتی تقسیم شود که حداقل تعداد مشخصی از نمونه‌ها در آن گره وجود داشته باشد. این کار باعث جلوگیری از ایجاد گره‌هایی با تعداد نمونه‌های کم می‌شود که می‌توانند باعث بیش‌برازش شوند.

۴. استفاده از حداقل نمونه‌ها برای برگ (Min Samples Leaf)

این پارامتر تعیین می‌کند که هر برگ باید حداقل دارای تعداد مشخصی از نمونه‌ها باشد. این کار باعث می‌شود تا برگ‌ها به‌گونه‌ای شکل بگیرند که از یادگیری ناهنجاری‌ها و نویز جلوگیری شود.

۵. اعتبارسنجی متقابل (Cross-validation)

اعتبارسنجی متقابل به تقسیم داده به چند بخش کمک می‌کند و از مدل روی بخش‌های مختلف استفاده می‌کند. این کار باعث می‌شود که مدل توانایی تعمیم بهتری داشته باشد و از تمرکز بیش از حد بر روی یک مجموعه خاص داده جلوگیری شود.

۶. منظم‌سازی (Regularization)

با استفاده از تکنیک‌های منظم‌سازی، مانند افزودن یک پارامتر جرمیه به تابع هزینه، مدل از پیچیدگی بیش از حد دوری می‌کند. این کار باعث می‌شود که مدل پارامترهایی را که کمک چندانی به دقت مدل نمی‌کنند، کنار بگذارد.

چگونه این روش‌ها به بهبود تعمیم کمک می‌کنند؟

این استراتژی‌ها به طور کلی با کاهش پیچیدگی مدل، جلوگیری از یادگیری جزئیات غیرضروری و نویز در داده‌های آموزشی، به بهبود تعمیم مدل کمک می‌کنند. به عبارت دیگر، مدل با استفاده از این روش‌ها بهتر می‌تواند الگوهای اصلی در داده‌ها را شناسایی کرده و بر روی داده‌های جدید عملکرد بهتری داشته باشد. در نتیجه، مدل نه تنها روی داده‌های آموزشی عملکرد خوبی دارد، بلکه روی داده‌های نادیده نیز دقت بالاتری را نشان می‌دهد.

سوال ۷

معیارهای مورد استفاده برای اندازگیری خلوص یک تقسیم در درخت‌های تصمیم، مانند شاخص جینی (Gini Index)، آنتروپی (Entropy) و نسبت کسب اطلاعات (Information Gain) را توضیح دهد. فرمول‌های آنها را درج کنید و بحث کنید که در کدام شرایط هر معیار ترجیحاً استفاده می‌شود.

پاسخ

آنتروپی(خلوص)

موضوع آنتروپی اولین بار توسط شanon معرفی شد فرض کنیم یک متغیر داریم $X = (x_1, \dots, x_m)$ که مقادیر (p_1, \dots, p_m) را دارد.

$$H(X) = -\sum_{i=1}^m p_i \log p_i$$

به خاطر اینکه در حوزه اطلاعات است به صورت بیت، اطلاعاتی که می‌دهد را نشان می‌دهیم چون \log از لحاظ ریاضی تعریف نشده بنابراین $\lim_{x \rightarrow 0} \log x$ به سمت بی‌نهایت میل می‌کند نشان می‌دهیم. وقتی مقدار آنتروپی برابر با صفر است یعنی بالاترین خلوص را دارد. آنتروپی معمولاً در شرایطی استفاده می‌شود که تعادل بیشتری بین کلاس‌ها وجود داشته باشد و نیاز به محاسبه دقیق عدم قطعیت داریم.

ID³

یک معیار شکستن ویژگی است و ویژگی را در ریشه درخت قرار می‌دهیم که بالاترین میزان information gain دارد.

گام اول:

اطلاعات مورد انتظاری که برای دسته بندی یک نمونه موجود در در D لازم است از فرمول زیر به دست می‌آید:

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$$

که در آن p_i احتمال تعلق یک نمونه در D به کلاس C_i است و با کمک عبارت $|C_{i,d}| / |D|$ تخمین زده می‌شود چون اطلاعات به صورت بیتی کدگذاری می‌شود از یکتابع لگاریتمی با پایه دو استفاده می‌شود فرمول بالا تنها میانگین مقدار اطلاعات لازم برای شناسایی برچسب کلاس یک نمونه در D است. در این نقطه اطلاعات در دسترس تنها بر اساس نسبت و سهم نمونه‌های هر دسته است این فرمول همچنین به عنوان آنتروپی شناخته می‌شود.

گام دوم:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

اطلاعات مورد انتظاری که برای دسته بندی یک مجموعه از نمونه‌ها بر اساس بخش‌بندی با صفت خاصه لازم است را نشان میدهد.

$$Gain(A) = Info(D) - Info_A(D)$$

این سنجه به صورت تفاضل میان اطلاعات مورد نیاز اولیه که تنها بر اساس نسبت دسته ها تعریف شده اند و نیاز جدید که پس از بخش بنده بر اساس صفت خاصه به دست آمده است تعریف می شود . این معیار زمانی ترجیح داده می شود که بخواهیم یک ویژگی را برای تقسیم انتخاب کنیم که بیشترین کاهش عدم قطعیت را ایجاد کند. این متريک معمولاً در الگوريتم ID³ به کار می رود اما ممکن است در ویژگی هایي با تعداد دسته های زياد سوگيري ایجاد کند.

اشکالات اساسی ID³

سنجه information gain متمایل به آزمون هاي با تعداد زيادي خروجي مي باشد يعني فرض کنيم يك IDnumber داشته باشيم که هر کدام يك هستند که آنتروپي صفر مي شود و در نتيجه $Info_A(D)$ صفر مي شود information gain وقتی که ویژگي ها به يك تعداد زيادي از مقادير متمایل شوند ID³ خوب عمل نمیکند و باياس ميشود

C4.5 ۱۰

بعد از ID³ معرفی شد و يك سنجه معیاري را به نام GainRatio برای غلبه بر اين مشکل به کار برد که به آن نرمال کردن information گويند همان کاري که در ID³ انجام داديم انجام ميدهيم اما يك شاخص $GainRatio$, $splitInfo_A(D)$ بدست می آوريم که از همان $Gain$ قبلی تقسیم $SplitInfo$ است با محاسبه مقادير اين سنجه برای ویژگي هاي متفاوت آن ویژگي که بالاترین مقدار را دارد در ریشه درخت قرار می گيرد و به عنوان صفت خاصه شکست انتخاب مي شود.

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$GainRatio(A) = Gain(A)/SplitInfo(A)$$

میزان اطلاعات لازم برای توزیع نمونه ها بین دسته های مختلف يك ویژگی را نشان می دهد. Gain Ratio زمانی $SplitInfo_A(D)$ استفاده می شود که بخواهیم از انتخاب ویژگی هاي با دسته های زياد جلوگيري کنیم.

مشکل ۴.۵

اگر SplitInfo برابر صفر شود فرمول GainRatio ناپايدار می شود برای رفع اين مشکل يك روش دیگر به نام CART معرفی می شود.

CART ۲۰

در الگوريتم CART از سنجه Giniindex استفاده می شود اين سنجه ناچالصی تاپل های آموزشی D را به صورت زير محاسبه می نماید:

$$Gini(D) = 1 - \sum_{j=1}^n p_j^2$$

شاخص جینی زمانی ترجیح داده می‌شود که نیاز به تقسیم‌های دوگانه در مجموعه داده‌ها باشد و همچنین در حالتی که الگوریتم CART به کار رود.

سنجه Gini index برای هرگره درخت دوشاخه را برسی می‌کند و به عبارت دیگر برای هر صفت خاصه یک شکست یا بخش بندی دوتایی را برسی می‌کند اگر صفت خاصه A گستته و a_1, \dots, a_v مقدار مجزا در مجموعه داده‌های D می‌باشد جهت تعیین بهترین شکست دودویی برای صفت خاصه A به شکل $S_A \in A$ خواهد بود برای یک تاپل هنگامی این شرط برآورده می‌شود که مقدار صفت خاصه A دارای index Gini مقدار باشد بنابراین \sum زیر مجموعه ممکن وجود دارد. بنابراین به 2^v روش ممکن این بخش بندی انجام می‌شود. سنجه Gini برای هرگره درخت دوشاخه را برسی می‌کند و به عبارت دیگر برای هر صفت خاصه یک شکست یا بخش بندی دوتایی را برسی می‌کند اگر صفت خاصه A گستته و a_1, \dots, a_v مقدار مجزا در مجموعه داده‌های D انجام می‌دهیم. برای هر یک از حالت‌های شکست یک مجموعه وزنی ناخالصی محاسبه می‌شود برای مثال چنانچه یک تقسیم دودویی بر روی صفت خاصه‌ی A مجموعه داده‌های D₁, D₂ را به دو مجموعه D_1, D_2 بخش بندی کند مقدار Gini index مجموعه داده‌های برای D₁ چنین افزایی برابر است با

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

کاهش ناخالصی که با شکست دودویی بر روی صفت خاصه A انجام می‌شود برابر است با

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

صفت خاصه‌ای که این کاهش ناخالصی را به حداقل خود برساند (یا حداقل مقدار Gini index) به عنوان صفت خاصه‌ی شکست انتخاب می‌شود

نتیجه‌گیری

هر یک از این معیارها در شرایط مختلف قابل استفاده هستند:

- آنتروپی و کسب اطلاعات information gain: زمانی مناسب هستند که تعادل بیشتری در داده‌ها وجود داشته باشد. این معیارها می‌توانند برای شناسایی بهترین ویژگی برای تقسیم داده‌ها بر اساس بیشترین کاهش در عدم قطعیت استفاده شوند.
- نسبت کسب اطلاعات (Gain Ratio): برای جلوگیری از سوگیری از سوگیری نسبت به ویژگی‌هایی که دارای مقادیر زیادی هستند به کار می‌رود. این معیار معمولاً در الگوریتم C4.5 به کار می‌رود.
- شاخص جینی (Gini Index): این شاخص معمولاً در شرایطی استفاده می‌شود که نیاز به تقسیم‌های دوگانه داریم و در الگوریتم CART به کار گرفته می‌شود. شاخص جینی برای محاسبه ناخالصی مجموعه داده‌ها استفاده می‌شود.

سوال ۸

مفهوم توازن بایاس-واریانس را توضیح دهید، فرمول دقیق آن را ارائه دهید و همچنین مفاهیم overfitting (بیش‌پرازش) و underfitting (کم‌پرازش) را در ارتباط با بایاس و واریانس مورد بحث قرار دهید.

توازن بایاس-واریانس

توازن بایاس-واریانس یک اصل مهم در یادگیری ماشین است که نشان می‌دهد افزایش پیچیدگی مدل‌ها چگونه می‌تواند بر دقت و تعمیم‌پذیری آن‌ها تأثیر بگذارد. در مدل‌های ساده، بایاس بالا است، زیرا مدل نمی‌تواند الگوهای پیچیده را یاد بگیرد. در مقابل، مدل‌های بسیار پیچیده واریانس بالایی دارند، زیرا به جزئیات بیش از حد داده‌های آموزش وابسته می‌شوند و روی داده‌های جدید عملکرد ضعیفی دارند. هدف اصلی در الگوریتم‌های یادگیری ماشین نظارت شده کاهش خطای مدل است. آیا تنها معیار برای اعتبارسنجی دقت است؟ خیر، داشتن اطلاعات دقیق در مورد این خطاهای نه تنها به ساخت مدل‌های دقیق کمک می‌کند، بلکه می‌تواند وضعیت *overfitting* (بیش‌برازش) و *underfitting* (کم‌برازش) را نیز برطرف کند. *bias* از خطای آموزشی و *variance* از خطای تست ناشی می‌شود.

خطای *Irreducible* و *Reducible*

خطای Irreducible: این خطاهای صرف نظر از الگوریتم مورد استفاده قابل کاهش نیستند و معمولاً به دلیل متغیرهای غیرمنتظره که مستقیماً بر خروجی تأثیر می‌گذارند، به وجود می‌آیند.

خطای Reducible: این خطاهای را می‌توان کاهش داد و به دو جزء *bias* و *variance* تقسیم می‌شوند.

چیست *Bias*؟

به فاصله بین مقادیر پیش‌بینی شده و واقعی اشاره دارد. اگر مقادیر پیش‌بینی شده به طور مداوم از مقادیر واقعی فاصله زیادی داشته باشند، این وضعیت به *bias* بالا معروف است و منجر به *underfitting* می‌شود.

$$\text{Bias} = \mathbb{E}[\hat{f}(x) - f(x)]$$

اینجا \hat{f} مدل پیش‌بینی شده و f خروجی واقعی است.

بایاس بالا:

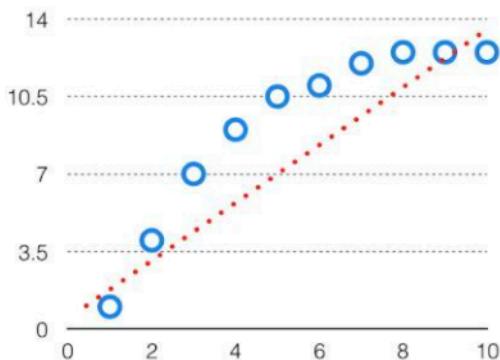
- معنی: بایاس بالا به معنای سادگی بیش از حد مدل است. مدل نمی‌تواند الگوهای پیچیده داده‌ها را به خوبی یاد بگیرد و در نتیجه نتایج آن فاصله زیادی با واقعیت دارند.
- مزایا: مدل‌های با بایاس بالا سریع‌تر اجرا می‌شوند و از لحاظ محاسباتی سبک‌تر هستند.
- معایب: دقت پایین و عدم توانایی در یادگیری الگوهای پیچیده، منجر به *underfitting* (کم‌برازش) می‌شود. در این حالت مدل نمی‌تواند به خوبی تعمیم یابد و نتایج روی داده‌های جدید ضعیف خواهد بود.

بایاس پایین:

- معنی: بایاس پایین به معنای این است که مدل بسیار دقیق است و به خوبی می‌تواند داده‌های آموزشی را پیش‌بینی کند.
- مزایا: مدل توانایی بسیار خوبی در یادگیری الگوهای پیچیده دارد و می‌تواند دقت بالایی در پیش‌بینی‌ها داشته باشد.

- معایب: بایاس پایین می‌تواند به overfitting (بیش‌پرازش) منجر شود، یعنی مدل به جزئیات داده‌های آموزشی بیش از حد وابسته می‌شود و عملکرد ضعیفی روی داده‌های جدید خواهد داشت.

شکل ۴: Model the in Bias High



Variance چیست؟

به میزان پراکندگی مقادیر پیش‌بینی شده از داده‌های واقعی اشاره دارد. اگر مدل بر روی داده‌های آموزشی عملکرد خوبی داشته باشد اما با داده‌های جدید (تست) شکست پخورد، نشان‌دهنده overfitting variance باشد که به variance منجر می‌شود.

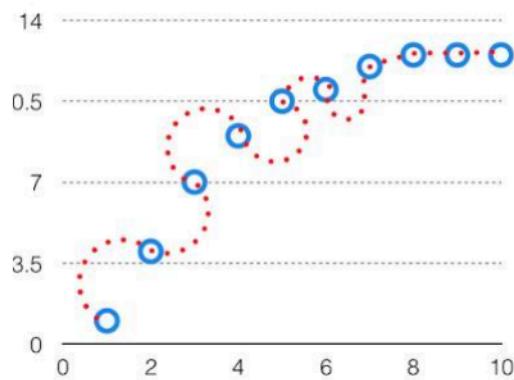
$$\text{(Variance)} = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$$

واریانس بالا:

- معنی: واریانس بالا به این معناست که مدل به تغییرات کوچک در داده‌های آموزشی بسیار حساس است. این مدل‌ها روی داده‌های آموزشی عملکرد خوبی دارند اما روی داده‌های جدید نتایج ناپایدار تولید می‌کنند.
- مزایا: مدل‌های با واریانس بالا دقیق‌تر با این تغییرات را پیش‌بینی می‌کنند.
- معایب: منجر به overfitting می‌شود، به این معنا که مدل نمی‌تواند به خوبی تعمیم یابد و در مواجهه با داده‌های جدید نتایج غیرقابل اعتمادی تولید می‌کند.

واریانس پایین:

- معنی: واریانس پایین به این معناست که مدل نسبت به تغییرات داده‌های آموزشی حساسیت کمتری دارد و نوسانات کمتری در پیش‌بینی‌ها دارد.
- مزایا: مدل تعمیم‌پذیری بهتری دارد و نتایج پایدارتر و مطمئن‌تری روی داده‌های جدید تولید می‌کند.
- معایب: واریانس پایین می‌تواند منجر به underfitting شود؛ در این حالت مدل ممکن است نتواند پیچیدگی‌های موجود در داده‌ها را پاد بگیرد و دقیق‌تر کاهش نماید.



شکل ۵: Model the in Variance High

تعادل بایاس(سوگیری)-واریانس (Bias-Variance Tradeoff)

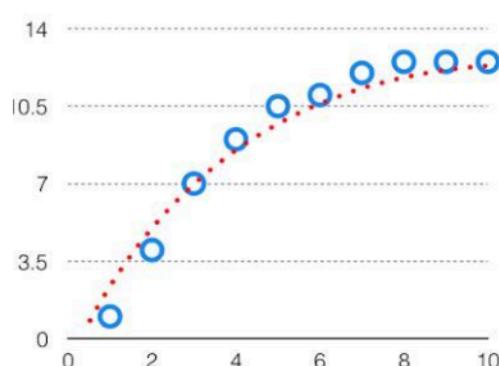
اگر الگوریتم بیش از حد ساده باشد (مثلاً فرضیه با معادله خطی)، مدل دارای سوگیری بالا و واریانس کم خواهد بود که منجر به خطا می‌شود.

از طرف دیگر، اگر الگوریتم بسیار پیچیده باشد (مثلاً فرضیه با درجات بالا)، مدل دارای واریانس بالا و سوگیری کم خواهد بود. به طور ایدهآل، تعادلی بین این دو وضعیت وجود دارد که به تعادل سوگیری-واریانس معروف است. به این ترتیب، بهترین مدل آن است که هم سوگیری و هم واریانس را به حداقل برساند.

فرمول خطای کل به شکل زیر تعریف می‌شود:

$$\text{Error Total} = \text{Bias}^2 + \text{Variance} + \text{Error Irreducible}$$

این فرمول به ما کمک می‌کند تا خطای کل مدل را بهینه کنیم و مدلی با کمترین خطای آموزشی و هم در داده‌های آزمایشی به دست آوریم.



بایاس و واریانس دو مفهوم کلیدی در یادگیری ماشین هستند که با توازن میان سادگی و پیچیدگی مدل سروکار دارند.

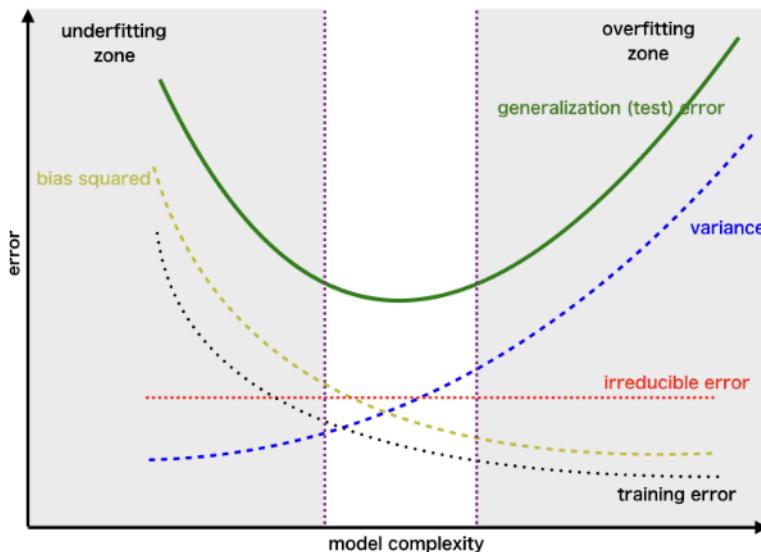
تأثیر *Variance* و *Bias* بر مدل:

مدل با *bias* بالا و *variance* پایین معمولاً دچار *underfitting* می‌شود.

مدل با *bias* پایین و *variance* بالا معمولاً دچار *overfitting* می‌شود.

Trade-off Bias-Variance

برای دستیابی به مدل بھینه باید تعادلی میان *bias* و *variance* برقرار شود تا مدل نه دچار *underfitting* و نه *overfitting* شود.



شکل ۶: Error Total of Value Least the for Region

از این به عنوان بهترین نقطه انتخاب شده برای آموزش الگوریتم یاد می شود که خطای کمی در آموزش و همچنین آزمایش داده ها می دهد.

فرمول دقیق توازن بایاس-واریانس:

فرمول کلی خطای مدل به صورت زیر است:

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Error Irreducible}$$

• **Bias:** نشان دهنده سادگی بیش از حد مدل و فاصله آن از حقیقت است.

• **Variance:** حساسیت مدل به داده های آموزشی و تغییرات کوچک در آنها را نشان می دهد.

• **Error: Irreducible** خطای ذاتی داده ها که قابل کاهش نیست.

:Underfitting و Overfitting

• **Overfitting (بیشبرازش):** وقتی مدل بسیار پیچیده است، واریانس بالا ولی بایاس کم می شود. این حالت منجر به تطبیق بیش از حد مدل با داده های آموزشی و عدم تعمیم پذیری به داده های جدید می شود.

• **Underfitting (کم برازش):** وقتی مدل بسیار ساده است، بایاس بالا و واریانس پایین است. مدل توانایی یادگیری الگوهای داده ها را ندارد و نمی تواند به خوبی عمل کند.

نتیجه‌گیری:

توازن بین بایاس و واریانس بسیار مهم است. مدل‌هایی که بایاس و واریانس متعادلی دارند، می‌توانند هم به خوبی یاد بگیرند و هم در مواجهه با داده‌های جدید عملکرد مطلوبی ارائه دهند. بنابراین، هدف اصلی در یادگیری ماشین، یافتن این توازن برای بهبود تعیین‌پذیری و دقت مدل است.

سوال ۹

توضیح دهید که چگونه اعتبارسنجی متقابل K-fold برای ارزیابی مدل استفاده می‌شود، از جمله ترتیب صحیح پیاده‌سازی آن در مورد مقیاس‌بندی داده‌ها و تقسیم‌بندی. همچنین، تاثیر انتخاب مقادیر مختلف "k" در اعتبارسنجی K-fold بر توازن خطای Bias-Variance (Tradeoff) در حین ارزیابی مدل را مورد بحث قرار دهید.

پاسخ

نرخ خطای تست، میزان خطای مدل بر روی داده‌های تست و نرخ خطای آموزشی، میزان خطای مدل بر روی داده‌های آموزشی است. عموماً نرخ خطای آموزشی بسیار کمتر از نرخ خطای تست است. نرخ خطای تست، یک معیار مناسب برای ارزیابی عملکرد مدل است. اما معمولاً مجموعه تست در دسترس نیست و باید به روش‌های دیگری مدل را ارزیابی کنیم. در اینجا ما نرخ خطای تست را با استفاده از نرخ خطای آموزشی تخمین می‌زنیم. هر بار یک زیرمجموعه از داده‌های آموزشی را کنار می‌گذاریم، مدل را با استفاده از بقیه داده‌ها آموزش می‌دهیم و از داده‌های کنار گذاشته شده به عنوان داده‌های تست استفاده می‌کنیم.

(validation set)

در این روش، مجموعه داده‌های آموزشی را به صورت تصادفی، به دو بخش آموزشی و ارزیابی تقسیم می‌کنیم. در شکل زیر یک تقسیم بندی داده‌ها را مشاهده می‌کنید. مجموعه داده‌های آموزشی اعداد ۱ تا n هستند. آنها را به طور تصادفی به دو بخش آبی (داده‌های آموزشی) و نارنجی (داده‌های ارزیابی) تقسیم می‌کنیم.. مدل را بر روی داده‌های آموزشی بدست آورده و خطای (MSE) را بر روی داده‌های ارزیابی حساب می‌کنیم. اما تقسیم‌بندی داده‌ها به دو دسته آموزشی و ارزیابی، به صورت تصادفی بود و اگر بخواهیم مجدداً داده‌ها را دسته‌بندی کنیم،



تقسیم‌بندی متفاوتی خواهیم داشت.

روش ارزیابی دو عیب دارد:

۱. براساس اینکه چه داده‌هایی در مجموعه آموزشی و چه داده‌هایی در مجموعه ارزیابی قرار بگیرند، خطای ارزیابی متغیر خواهد بود.
۲. از آنجایی که در این روش تنها زیرمجموعه‌ای از داده‌های آموزشی برای آموزش مدل استفاده می‌شوند و هر چه تعداد داده‌های آموزش کمتر باشد، دقت مدل کمتر است؛ این روش می‌تواند خطای ارزیابی بسیار بیشتری نسبت به مدلی داشته باشد که از کل داده‌های آموزش اولیه برای آموزش مدل استفاده می‌کند. در بخش‌های بعدی، با اعمال تغییراتی، این روش را اصلاح می‌کنیم.

روش Leave-One-Out-Cross-Validation (LOOCV)

این روش بسیار مشابه روش قبل است؛ اما قصد دارد با ایجاد تغییراتی ایرادات روش قبل را برطرف کند. در این روش تنها داده آموزشی (x_1, y_1) به عنوان داده ارزیابی کنار گذاشته می‌شود و مدل بر اساس $n-1$ داده آموزشی دیگر یعنی $\{(x_2, y_2), \dots, (x_n, y_n)\}$ بدست می‌آید.

سپس خروجی برای تنها داده ارزیابی یعنی x_1 پیش‌بینی می‌شود و \hat{y}_1 بدست می‌آید. حال مقدار میزان خطأ بر روی تنها داده ارزیابی محاسبه می‌شود که مقدار این خطأ برابر است با^۲ $MSE_1 = (y_1 - \hat{y}_1)^2$ این خطأ، تخمینی از نرخ خطای تست است. اما از آنجایی که این خطأ بسیار وابسته به تک داده ارزیابی است، این مراحل را به ازای تک تک داده‌های آموزش انجام می‌دهیم. هر بار یکی از داده‌های آموزشی را به عنوان داده ارزیابی کنار گذاشت، مدل را بر روی $n-1$ داده دیگر بدست آورده و خطای مدل را بر روی تک داده ارزیابی حساب می‌کنیم و به این ترتیب مقادیر $MSE_1..MSE_{n-1}..MSE_n$ محاسبه می‌شوند. در شکل زیر نمایی کلی از این روش را می‌بینید. (هربار داده نارنجی رنگ داده ارزیابی است).



نهایتاً روش LOOCV خطای زیر را به عنوان تخمینی از نرخ خطای تست گزارش می‌کند:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n MSE_i$$

روش LOOCV مزایای زیادی نسبت به روش ارزیابی بخش قبل دارد. این روش اریب bias کمتری دارد. زیرا به جای آنکه مدل را بر اساس بخشی از داده‌های آموزشی بدست آورده، تقریباً بر اساس تمام داده‌های آموزشی (همه به جز یکی) بدست می‌آورد. بنابراین به بخشی از داده‌های آموزشی وابسته نیست و ناریب‌تر است. همچنین میزان خطای ارزیابی که بدست می‌آورد بیشتر از نرخ خطای تست نیست چون تقریباً از تمام داده‌های آموزشی استفاده می‌کند. علاوه بر این، خطایی که با این روش بدست می‌آید، پس از اعمال مجدد روش، تغییر نمی‌کند چون هیچ بخشی از آن تصادفی نیست.

این روش به لحاظ محاسباتی بسیار زمانبر است، مخصوصاً اگر n بزرگ باشد. زیرا باید n بار، مدل آموزش دهد. با روش کمترین مربعات خطأ و رگرسیون چندجمله‌ای می‌توان میزان محاسبات LOOCV را در حد شگفت‌آوری پایین آورد و تنها با یک بار آموزش مدل و استفاده از فرمول زیر، خطای ارزیابی را حساب کرد.

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

در این فرمول \hat{y}_i خروجی پیش‌بینی شده برای داده اولیه i و h_i میزان تاثیرگذاری داده i ام بر مدل است. مقدار h_i بین $\frac{1}{n}$ تا ۱ است. بنابراین میزان باقیمانده هر داده به اندازه میزان تاثیرگذاری آن بر مدل، در محاسبه خطأ اثرگذار است. این فرمول یکی از فرمول‌های زیبای آمار است که در حالت کلی برقرار نیست و در بسیاری از موارد، مجبور به محاسبه n مدل هستیم.

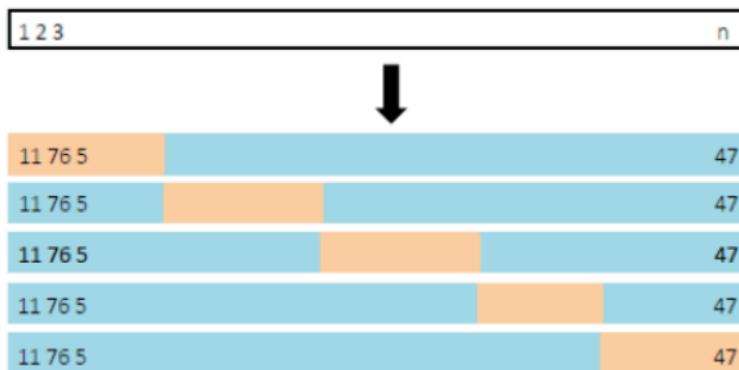
k-fold Cross-Validation

اعتبارسنجی متقابل K-fold یک تکنیک آماری است که برای ارزیابی عملکرد مدل‌های یادگیری ماشین استفاده می‌شود. در این روش، داده‌ها به k بخش مساوی تقسیم می‌شوند و مدل k بار آموزش داده می‌شود؛ اولین بخش را به عنوان داده ارزیابی کنار گذاشته و مدل را بر اساس $1-k$ بخش دیگر بدست می‌آوریم و خطأ MSE_1 را روی داده‌های ارزیابی محاسبه می‌کنیم. این روند را k بار تکرار می‌کنیم. هر

بار یکی از دسته‌ها را به عنوان داده ارزیابی کنار گذاشته و پس از بدست آوردن مدل، خطا را بر روی داده ارزیابی حساب می‌کینم. به این ترتیب، خطاهای $MSE_1..MSE_k$ محاسبه می‌شوند. نهایتاً خطای زیر، به عنوان تخمینی از نزخ خطای تست گزارش می‌شود:

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i$$

در شکل زیر نمای کلی *foldcross-validation* – ۵ را مشاهده می‌کنید. داده‌ها به ۵ قسمت تقسیم می‌شوند و هر بار یک دسته نارنجی رنگ به عنوان داده ارزیابی کنار گذاشته می‌شود. این روند ۵ بار تکرار می‌شود.



روش LOOCV حالت خاصی از k -fold CV است که در آن $n = k$ است. در اغلب موارد از 5-fold CV استفاده می‌کنند. مهم‌ترین علت این امر این است که در مواردی که n خیلی بزرگ باشد، LOOCV بسیار زمان بر است و نیاز به بدست آوردن n مدل دارد اما 5-fold CV تنها نیاز به محاسبه ۵ مدل دارد.

وقتی از cross-validation استفاده می‌کنیم، گاهی تنها به دنبال یافتن تخمینی از خطای نیست و می‌خواهیم از بین مدل‌های با ویژگی‌های متفاوت (مثلاً درجات چندجمله‌ای متفاوت) مدل بهینه را انتخاب کنیم. در این حالت، در واقع به دنبال یافتن نقطه مینیمم در نمودارهای پایین هستیم.

ترتیب صحیح مقیاس‌بندی داده‌ها و تقسیم‌بندی

برای جلوگیری از ایجاد بایاس در ارزیابی مدل، ابتدا باید داده‌ها به مجموعه‌های آموزش و تست تقسیم شوند و سپس عملیات مقیاس‌بندی (مثل نرمال‌سازی) انجام گیرد. اگر قبل از تقسیم داده‌ها عملیات مقیاس‌بندی انجام شود، ممکن است اطلاعاتی از مجموعه تست به مجموعه آموزش نشست کند و ارزیابی مدل به طور غیرواقعی بهدود یابد.

تعريف بایاس و واریانس

بایاس نشان‌دهنده این است که چقدر مدل از داده‌های واقعی دور است؛ یعنی تا چه حد مدل ساده‌سازی شده و نتایج به دور از دقت واقعی هستند. مدل‌هایی با بایاس بالا ممکن است نتایج ساده‌ای ایجاد کنند که توانایی تعمیم مدل به داده‌های جدید را کاهش می‌دهد. واریانس نشان‌دهنده حساسیت مدل به تغییرات کوچک در داده‌های آموزش است. مدل‌هایی با واریانس بالا ممکن است دقت بسیار خوبی در داده‌های آموزش داشته باشند اما روی داده‌های جدید نتایج ناپایداری تولید کنند.

فرمول بایاس-واریانس

توازن بایاس-واریانس به صورت زیر در مدل‌ها تعریف می‌شود:

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Error Irreducible}$$

- بایاس: خطای ناشی از سادگی بیش از حد مدل که باعث می‌شود پیش‌بینی‌ها نتوانند به نتایج واقعی نزدیک شوند. - واریانس: خطای ناشی از پیچیدگی زیاد مدل و حساسیت آن به داده‌های آموزش.

توازن بایاس-واریانس در ارزیابی CV k-fold

قبلاً گفته شد که CV به لحاظ محاسباتی به LOOCV ارجحیت دارد. مزیت دیگر CV به LOOCV این است که CV تخمین دقیق‌تری از نرخ خطای تست ارائه می‌دهد و این به دلیل توازن بایاس-واریانس است. همان‌طور که گفته شد، روش ارزیابی ممکن است نرخ خطای تست را بیش‌تر تخمین بزند، چون از تمام داده‌های آموزشی برای آموزش استفاده نمی‌کند و بنابراین دارای بایاس زیادی است. بر عکس، روش LOOCV به دلیل استفاده از تقریباً تمام داده‌ها برای آموزش، تخمین ناریبی از خطای تست می‌دهد. با این منطق می‌توان نتیجه‌گرفت که CV با $k=1$ یا $k=n$ بایاس متوسطی دارد زیرا تعداد داده‌هایی که برای آموزش استفاده می‌کند $\frac{n(k-1)}{n}$ داده‌است که کمتر از داده‌های استفاده شده در LOOCV و بیشتر از داده‌های روش ارزیابی است. بنابراین اگر تنها هدف ما کاهش بایاس باشد، روش LOOCV ارجح است. اما بایاس تنها منبع ایجاد خطای تخمین نیست و باید به واریانس تخمین هم توجه کنیم. روش LOOCV دارای واریانس بیشتری نسبت به روش CV است زیرا در روش LOOCV میانگین خروجی n مدل را محاسبه می‌کنیم. با توجه به اینکه این n مدل دارای همپوشانی بسیار زیاد در داده‌های آموزشی هستند، خروجی‌هایشان بسیار به هم وابسته است. این وابستگی بین خروجی‌ها در روش CV کمتر است زیرا در روش CV بین داده‌های آموزشی هر k مدل، همپوشانی کمتری وجود دارد. از آنجایی که میانگین مقادیری که بسیار به هم وابسته هستند، دارای واریانس زیادی است، بنابراین تخمین روش LOOCV دارای واریانس زیادی است. به طور خلاصه، انتخاب k در CV میزان بایاس و واریانس را تغییر می‌دهد. انتخاب $k=1$ یا $k=n$ انتخاب‌های معقولی هستند زیرا نه بایاس بالایی دارند و نه واریانس بالا.

تأثیر انتخاب مقادیر مختلف "k" بر توازن بایاس-واریانس

انتخاب مقدار k تاثیر زیادی بر توازن بایاس-واریانس در مدل دارد:

- مقادیر کوچک k (مثل $k=5$): معمولاً منجر به بایاس بالاتر و واریانس کمتر می‌شوند. در این حالت، مدل به طور نسبتاً ساده‌تری ارزیابی می‌شود، اما ارزیابی نهایی ممکن است به اندازه کافی دقیق نباشد. - مقادیر بزرگ‌تر k (مثل $k=10$): باعث می‌شوند که ارزیابی مدل دقیق‌تر باشد و بایاس کمتری داشته باشد، اما واریانس بیشتر می‌شود، زیرا هر بخش آموزش شامل نمونه‌های کمتری از داده‌ها است که می‌تواند باعث نوسانات بیشتری در نتایج شود.

انتخاب مقدار مناسب برای k در اعتبارسنجی K-fold به تنظیم صحیح توازن بایاس-واریانس کمک می‌کند. انتخاب k کوچک باعث افزایش بایاس و کاهش واریانس می‌شود، در حالی که k بزرگ‌تر، بایاس را کاهش و واریانس را افزایش می‌دهد.

Cross-Validation برای مسایل کلاس بندی

تا اینجا از خطای MSE استفاده کردیم چون خروجی پیوسته بود. اگر خروجی گسسته باشد، باید از تعداد داده‌های به اشتیاه دسته‌بندی شده به عنوان خطا استفاده کنیم. فرض کنید داده‌ها را به K بخش تقریباً مساوی C_1, C_2, \dots, C_K تقسیم می‌کنیم، اگر n_k تعداد مشاهدات در کلاس C_k باشد داریم:

$$CV_K = \sum_{k=1}^K \frac{n_k}{n} \text{Err}_k$$

که $\text{Err}_k = \frac{\sum_{i \in C_k} I(y_i \neq \hat{y}_i)}{n_k}$. انحراف معیار استاندارد CV_K به صورت زیر تعریف می‌شود

$$\hat{SE}(CV_K) = \sqrt{\frac{1}{K} \sum_{k=1}^K \frac{(\text{Err}_k - \bar{\text{Err}}_k)^2}{K-1}}$$

سوال ۱۰

اعتبار سنجی مقاطع k-fold طبقه بندی شده و اعتبار مقاطع k-fold معمولی را مقایسه کنید. چه زمانی از اعتبار سنجی مقاطع k-fold طبقه بندی شده استفاده می‌کنیم؟

پاسخ

اعتبار سنجی مقابل (Cross-validation)

اعتبار سنجی مقابل یک روش پذیرفته شده در جامعه‌ی یادگیری ماشین برای ارزیابی عملکرد مدل است. در این روش، داده‌ها به چندین مجموعه تقسیم می‌شوند، مدل روی بخشی از داده‌ها آموزش داده می‌شود و سپس روی بخش باقی‌مانده تست می‌شود. یکی از روش‌های رایج در این زمینه، K-Fold Cross-Validation است.

در اعتبار سنجی k-fold معمولی، داده‌ها به k قسمت مساوی تقسیم می‌شوند و در هر دور از اعتبار سنجی، یکی از این بخش‌ها برای تست و مابقی برای آموزش مدل استفاده می‌شوند. این فرآیند تا زمانی که همه‌ی بخش‌ها یک بار برای تست استفاده شوند، تکرار می‌شود. مزیت این روش در این است که به طور منصفانه داده‌ها یک بار برای تست و چندین بار برای آموزش استفاده می‌شوند. فرمول محاسبه دقت در اعتبار سنجی K-fold به این صورت است:

$$\text{accuracy K-fold} = \frac{1}{k} \sum_{i=1}^k \text{accuracy}_i$$

با این روش می‌توان اطمینان حاصل کرد که تمام داده‌ها برای آموزش و ارزیابی مدل مورد استفاده قرار می‌گیرند و مدل به داده‌های خاصی بیش از اندازه وابسته نمی‌شود.

اما یک چالش احتمالی در این روش این است که اگر تقسیم‌بندی‌ها نماینده‌ی مناسبی از توزیع کلی داده‌ها نباشند، به ویژه در مورد کلاس‌های نامتوازن، چه باید کرد؟ در اینجا Stratified K-Fold Cross-Validation وارد می‌شود.

Stratified K-Fold Cross-Validation نوعی از K-Fold Cross-Validation است که اطمینان می‌دهد در هر fold نسبت هر کلاس در داده‌ها مشابه نسبت کلی مجموعه داده‌ها باقی بماند. این امر به خصوص در مواردی که یک کلاس بسیار کمتر از سایر کلاس‌ها مشاهده شده باشد، اهمیت دارد.

در K-fold طبقه‌بندی شده (Stratified K-fold)، علاوه بر اینکه داده‌ها به k قسمت تقسیم می‌شوند، داده‌ها به گونه‌ای تقسیم می‌شوند که هر کلاس در هر بخش از داده‌های آموزشی و آزمایشی به طور متوازن و متناسب حضور داشته باشد. این کار برای حفظ نسبت کلاس‌ها در هر دسته (Training, Testing) انجام می‌شود. توزیع کلاس‌های هدف نیز در هر بخش مشابه با توزیع اصلی مجموعه داده‌ها حفظ می‌شود. این امر باعث می‌شود که هر بار که مدل آموزش می‌بیند و تست می‌شود، توزیع کلاس‌ها متعادل بماند.

فرمول خطای K-Fold Cross-Validation مشابه Stratified K-Fold Cross-Validation معمولی است:

$$CV_k = \frac{1}{K} \sum_{i=1}^K \text{MSE}_i$$

در اینجا:

• تعداد "folds" ها است.

• MSE_i خطای میانگین مربعات (Mean Squared Error) در i -امین fold است.

این فرمول، میانگین خطای پیش‌بینی در همه های "fold" مختلف را محاسبه می‌کند و برای ارزیابی عملکرد کلی مدل استفاده می‌شود. K-Fold Stratified باعث می‌شود که هر fold به طور متعادل داده‌ها را بر اساس کلاس‌های هدف داشته باشد و از مشکلات مرتبط با عدم تعادل داده‌ها جلوگیری کند.

چه زمانی از K-fold طبقه‌بندی شده استفاده می‌کنیم؟

۱. تقسیم‌بندی‌های متوازن: هنگامی که داده‌ها نامتوازن باشند، به این معنا که یک یا چند کلاس به طور قابل توجهی کمتر از سایرین وجود داشته باشند. در K-Fold معمولی، ممکن است توزیع نامتوازن کلاس‌ها در میان fold‌ها به وجود آید. K-Fold Stratified تضمین می‌کند که هر fold دارای توزیع کلاس‌های مشابهی باشد.

۲. بهبود تعمیم‌پذیری: از آنجا که هر fold دارای توزیع مشابهی از کلاس‌ها است، مدل کمتر به سمت یک کلاس خاص تمایل پیدا می‌کند و در نتیجه عملکرد بهتری روی داده‌های جدید و دیده‌نشده خواهد داشت. این روش زمانی مفید است که بخواهیم مطمئن شویم در هر بخش از اعتبارسنجی، توزیع کلاس‌ها بازتاب‌دهنده توزیع کلی داده‌ها باشد و مدل ما عملکرد قابل اعتمادتری داشته باشد.

۳. معیارهای عملکرد قابل اعتمادتر: با حفظ تعادل توزیع کلاس‌ها در هر fold، معیارهایی مانند دقت (Accuracy)، دقت (Precision) و بازیابی (Recall) قابل اعتمادتر خواهند بود.

K-Fold Stratified پیاده‌سازی

کلاس Scikit-learn.model_selection.StratifiedKFold می‌تواند برای پیاده‌سازی این روش در کتابخانه `sklearn.model_selection` استفاده کند. این کلاس اندیس‌های آموزشی/آزمایشی را برای تقسیم داده‌ها به مجموعه‌های آموزشی/آزمایشی فراهم می‌کند.

Nomene برداری تصادفی و Sampling: Stratified

فرض کنید می‌خواهیم یک نظرسنجی از ۱۰۰۰ نفر در یک ایالت خاص انجام دهیم. اگر به طور تصادفی ۱۰۰۰ مرد یا ۱۰۰۰ زن یا ترکیب نامتقارنی از افراد را انتخاب کنیم، نمی‌توان به طور دقیق نظرات کل ایالت را مشخص کرد. این نمونه‌برداری تصادفی است. اما در نمونه‌برداری طبقه‌بندی شده، اگر ۵۱٪ از جمعیت مرد و ۴۸٪ زن باشند، ۵۱۳ مرد و ۴۸۷ زن را برای نظرسنجی انتخاب می‌شود. این روش توزیع دقیقی از کل جمعیت را فراهم می‌کند و به آن *Sampling Stratified* گفته می‌شود.

چه زمانی Cross-Validation K-Fold Stratified مناسب است؟

این روش به خصوص زمانی توصیه می‌شود که کلاس‌های داده نامتوازن هستند، زیرا استفاده از نمونه‌برداری تصادفی می‌تواند منجر به نتایج نادرست و عدم تعادل در داده‌ها شود.

نتیجه‌گیری:

در K-fold معمولی، ممکن است توزیع کلاس‌ها به طور تصادفی نامتوازن شود، در حالی که در Stratified K-fold، توزیع کلاس‌ها در هر fold به طور متناسب حفظ می‌شود. استفاده از Stratified K-fold در شرایطی که داده‌ها نامتوازن هستند، می‌تواند ارزیابی مدل را دقیق‌تر کند و بایاس ناشی از نمونه‌های نامتوازن را کاهش دهد. این روش در کاربردهای مختلفی مانند تشخیص بیماری‌ها، شناسایی تقلب و هر جایی که اهمیت کلاس‌های نادر بالا است، استفاده می‌شود.

سوال ۱۱

دقت، بازیابی و امتیاز F1 چیستند و چگونه در زمینه مدل‌های طبقه‌بندی با یکدیگر مرتبط می‌شوند، بهویژه در مورد مبادله بین دقت و بازیابی؟

پاسخ

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

در مدل‌های طبقه‌بندی، سه معیار مهم برای ارزیابی عملکرد مدل‌ها وجود دارند که عبارتند از دقت، بازیابی و امتیاز F1. این معیارها به صورت زیر تعریف می‌شوند:

۱. دقت: (Precision)

این معیار نشان می‌دهد که از بین تمام مواردی که مدل به عنوان "مثبت" پیش‌بینی کرده است، چند درصد از آن‌ها واقعاً مثبت بوده‌اند. فرمول دقت به شکل زیر است:

$$\text{Precision} = \frac{TP}{TP + FP}$$

که در آن:

(True Positive) TP - تعداد پیش‌بینی‌های مثبت صحیح

تعداد پیش‌بینی‌های مثبت اشتباه (False Positive) FP - حداکثر مقدار این معیار یک و یا 100 درصد و حداقل مقدار آن صفر است و هرچه مواردی که برنامه به غلط پیش‌بینی کرده است که به آن می‌گوییم نسبت به پیش‌بینی‌های درست یا True Positive باشد مقدار Precision کمتر خواهد شد.

۲. بازیابی (Recall)

این معیار (که به عنوان حساسیت یا نرخ مثبت صحیح نیز شناخته می‌شود) نشان می‌دهد که از بین تمام موارد "مثبت" واقعی، چند مورد توسط مدل به درستی شناسایی شده‌اند. فرمول بازیابی به صورت زیر است:

$$\text{Recall} = \frac{TP}{TP + FN}$$

که در آن:

تعداد پیش‌بینی‌های مثبت صحیح (True Positive) TP -
تعداد موارد مثبت واقعی که به اشتباه منفی پیش‌بینی شده‌اند (False Negative) FN -
حداکثر مقدار این معیار یک و یا 100 درصد و حداقل مقدار آن صفر است و هرچه مواردی که ما انتظار داشتیم پیش‌بینی شوند ولی برنامه پیش‌بینی نکرده است که به آن False Negative می‌گوییم نسبت به پیش‌بینی‌های درست یا True Positive باشد مقدار Recall کمتر خواهد شد.

۳. امتیاز F1 Score

امتیاز F1 میانگین هماهنگ دقت و بازیابی است این معیار یک عدد واحد را ارائه می‌دهد که هر دو را متعادل می‌کند. برای زمانی مناسب است که نیاز به تعادل بین دقت و بازیابی داریم. فرمول امتیاز F1 به صورت زیر است:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

زمانی که می‌خواهیم معیار ارزیابی ما میانگینی از دو مورد قبلی باشد یعنی همان Precision یا Recall می‌توان از میانگین هارمونیک این دو معیار استفاده کرد که به آن معیار $f1$ -score می‌گویند.

تعادل بین دقت و بازیابی

بین دقت و بازیابی معمولاً یک رابطه مبادله‌ای وجود دارد. یعنی افزایش دقت ممکن است منجر به کاهش بازیابی شود و برعکس. به عنوان مثال، اگر مدل به شکلی تنظیم شود که فقط مواردی را که به احتمال زیاد مثبت هستند، انتخاب کند، دقت بالا می‌رود ولی ممکن است برخی از موارد مثبت واقعی از دست بروند (کاهش بازیابی). در مقابل، اگر مدل بخواهد تمامی موارد مثبت را شناسایی کند، بازیابی افزایش می‌یابد، ولی ممکن است بسیاری از پیش‌بینی‌های اشتباه نیز مثبت تشخیص داده شوند (کاهش دقت). این مبادله اهمیت استفاده از معیارهایی مانند Score F1 را نشان می‌دهد که به ما اجازه می‌دهد تا هر دو معیار را به صورت همزمان ارزیابی کنیم و تعادل مناسبی بین دقت و بازیابی برقرار کنیم.

در مدل‌های دسته‌بندی، بهبود یک معیار اغلب منجر به کاهش معیار دیگر می‌شود: دقت بالا، بازخوانی پایین: مدل پیش‌بینی‌های مثبت نادرستی کمتری انجام می‌دهد، اما ممکن است بسیاری از موارد مثبت واقعی را از دست بدهد.

بازخوانی بالا، دقت پایین: مدل موارد مثبت بیشتری را شناسایی می‌کند، اما به هزینه اشتباه گرفتن تعداد بیشتری از نمونه‌های منفی به عنوان مثبت.

امتیاز F_1 این تعادل را حفظ می‌کند و زمانی که هر دو معیار دقت و بازخوانی اهمیت دارند، به ویژه در مجموعه داده‌های نامتوازن، ارزیابی پایدارتر و متعادل‌تر ارائه می‌دهد.

سوال ۱۲

منحنی ROC چیست، چه زمانی معمولاً برای ارزیابی مدل استفاده می‌شود، و چگونه می‌توان امتیاز زیر منطقه منحنی (AUC) را در ارتباط با عملکرد مدل طبقه‌بندی تفسیر کرد؟

پاسخ

نرخ مثبت واقعی (TPR) که به عنوان حساسیت نیز شناخته می‌شود، به صورت زیر محاسبه می‌شود:

$$TPR = \frac{TP}{TP + FN}$$

که در آن TP تعداد موارد مثبت واقعی و FN تعداد منفی‌های کاذب است. TPR به ما می‌گوید که چه نسبتی از طبقه مثبت به درستی طبقه بندی شده است. یک مثال ساده این است که مشخص شود چه نسبتی از افراد بیمار واقعی به درستی توسط مدل شناسایی شده اند.

نرخ مثبت کاذب (FPR) به صورت زیر محاسبه می‌شود:

$$FPR = \frac{FP}{TN + FP}$$

که در آن FP تعداد موارد منفی که به اشتباه به عنوان مثبت شناسایی شده‌اند و TN تعداد موارد منفی واقعی است. FPR به ما می‌گوید که چه نسبتی از کلاس منفی توسط طبقه بندی کننده به اشتباه طبقه بندی شده است.

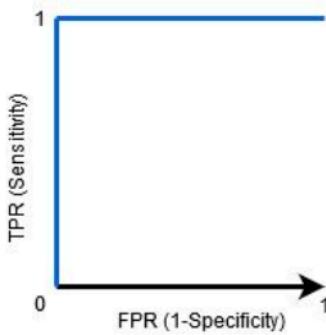
منحنی ROC (ویژگی‌های عملیاتی گیرنده) یک معیار گرافیکی است که برای ارزیابی عملکرد مدل‌های طبقه‌بندی باینری به کار می‌رود. این منحنی، نرخ مثبت واقعی (TPR) با True Positive Rate (FPR) را در برابر نرخ مثبت کاذب (FPR) یا False Positive Rate (FPR) در مقادیر مختلف آستانه‌ها رسم می‌کند. هدف از این منحنی نمایش کارایی مدل در تمایز بین کلاس‌ها است. اساساً، این منحنی به ما کمک می‌کند تا سیگنال (موارد مثبت واقعی) را از نویز (موارد مثبت کاذب) جدا کنیم. هر چه مدل بتواند با دقت بیشتری بین این دو تمایز قابل شود، عملکرد بهتری دارد.

مساحت زیر منحنی (AUC) معیاری است که توانایی مدل برای تمایز بین کلاس‌ها را اندازه‌گیری می‌کند. AUC می‌تواند به عنوان یک خلاصه از عملکرد کلی مدل در نظر گرفته شود؛ هرچه مقدار AUC بیشتر باشد (تا حداقل ۱)، مدل توانایی بیشتری در تمایز دقیق بین کلاس‌ها دارد.

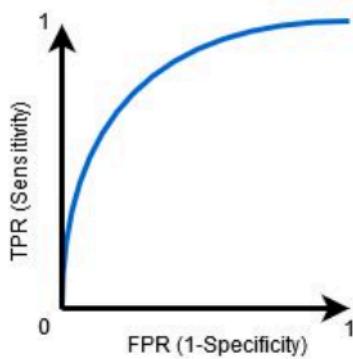
به طور خلاصه، منحنی ROC به ما کمک می‌کند تا حساسیت (TPR) و نرخ خطای مثبت کاذب (FPR) را در مدل‌های طبقه‌بندی باینری تحلیل کرده و مدل‌های مختلف را مقایسه کنیم.

هر چه AUC بالاتر باشد، عملکرد مدل در تشخیص کلاس‌های مثبت و منفی بهتر است.

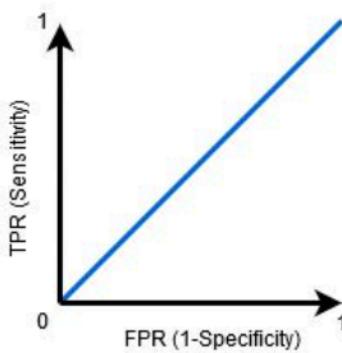
وقتی $AUC = 1$ ، طبقه‌بندی کننده می‌تواند کاملاً بین تمام نقاط کلاس مثبت و منفی به درستی تمایز قابل شود.



با این حال، اگر $AUC = 0$ بود، طبقه‌بندی‌کننده همه منفی‌ها را به عنوان مثبت و همه مثبت‌ها را به عنوان منفی پیش‌بینی می‌کرد.



وقتی AUC بین ۰.۵ و ۱ باشد، احتمال زیادی وجود دارد که طبقه‌بندی‌کننده بتواند مقادیر کلاس مثبت را از مقادیر کلاس منفی تشخیص دهد. این به این دلیل است که طبقه‌بندی‌کننده می‌تواند تعداد بیشتری از مثبت‌ها و منفی‌های درست را نسبت به منفی‌های غلط و مثبت‌های غلط تشخیص دهد.



در منحنی ROC، مقدار بالاتر محور X نشان‌دهنده تعداد بیشتری از تشخیص‌های مثبت کاذب (False Positive) نسبت به نقاط منفی حقيقی است. در حالی که مقدار بالاتر محور Y نشان‌دهنده تعداد بیشتری از تشخیص‌های مثبت حقیقی (True Positive) نسبت به نقاط منفی کاذب است. بنابراین، انتخاب آستانه به توانایی مدل در تعادل بین تشخیص‌های مثبت کاذب و منفی کاذب بستگی دارد. در این منحنی، محور افقی FPR (نرخ مثبت‌های کاذب) و محور عمودی TPR (نرخ مثبت‌های حقیقی یا حساسیت) رسم شده‌اند. نمودار خطی که از نقطه $(0, 0)$ تا نقطه $(1, 1)$ می‌گذرد، نشان‌دهنده این است که عملکرد طبقه‌بندی‌کننده تصادفی است. به این معنا که برای هر نرخ مثبت کاذب، یک نرخ مثبت حقیقی مشابه وجود دارد. بنابراین، مدل نمی‌تواند بین طبقه‌های مثبت و منفی تمایز قائل شود. به عبارت دیگر، در این حالت طبقه‌بندی‌کننده هیچ‌گونه اطلاعاتی از سیگنال را به درستی استخراج نمی‌کند و خروجی آن همانند حدس تصادفی عمل می‌کند. در این نمودار، خط یکنواخت نمایانگر این است که $TP = FP$ (یعنی نرخ مثبت‌های کاذب و نرخ مثبت‌های حقیقی

برابر هستند).

این وضعیت نشان دهنده عملکرد ضعیف طبقه بندی کننده است، چرا که نتایج تصادفی ارائه می دهد و هیچ تمایزی بین کلاس های مختلف قائل نمی شود.

مثال عملی

در نظر بگیرید که یک مطالعه پژوهشی یک آزمایش تشخیصی جدید برای بیماری را ارزیابی می کند که منحنی ROC و AUC برای ارزیابی اثربخشی آزمایش استفاده می شود. اگر آزمایش تشخیصی AUC بیش از ۰.۸ داشته باشد، این نشان می دهد که دقت بالایی در تمیز دادن بین بیماران مبتلا و افراد سالم دارد، که به تصمیم گیری های دقیق تر پژوهشی کمک می کند.

سوال ۱۳

توضیح دهید که چرا از نرم L1 در رگرسیون لاسو (Lasso) و از نرم L2 در رگرسیون ریج (Ridge) استفاده می شود. این نرم ها چگونه بر ضرایب مدل به طور متفاوتی تأثیر می گذارند؟ همچنین، رگولاریزاسیون Elastic Net را توصیف کنید، چگونگی ترکیب خواص ریج و لاسو را شرح دهید و سناریوهایی را که در آن ها Elastic Net به ویژه مفید است، توضیح دهید.

پاسخ

روش های فشرده سازی (Shrinkage Methods)

در روش های انتخاب زیرمجموعه، از روش کمترین مربعات برای یافتن تخمین زیرمجموعه ای از پارامترها استفاده می شود. اما در روش های فشرده سازی، تمام پارامترها در مدل هستند و با اعمال محدودیت های منظم سازی، با فشرده کردن ضرایب به سمت صفر، سعی در کاهش واریانس ضرایب داریم. دو روش محبوب فشرده سازی، ridge و lasso هستند.

- Regression Ridge در رگرسیون خطی، از روش کمترین مربعات خطای برای تخمین ضرایب استفاده می کردیم و به دنبال یافتن ضرایبی بودیم که خطای زیر را کمینه کند:

$$RSS = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2$$

روش رگرسیون ridge بسیار مشابه روند بالاست اما یک تفاوتی دارد. در این روش به دنبال یافتن ضرایبی هستیم که مقدار خطای زیر را کمینه کند:

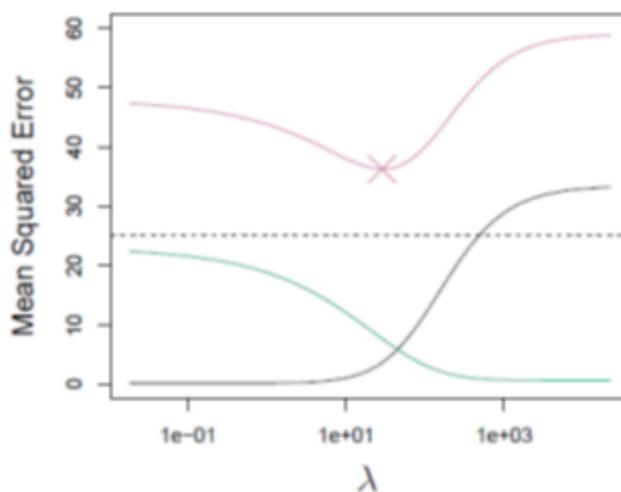
$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = RSS + \lambda \sum_{j=1}^p \beta_j^2 \quad (1)$$

که $\lambda \geq 0$ یک پارامتر تنظیم کننده است. در واقع با این تابع خطای هم به دنبال کمینه کردن خطای آموزشی و هم به دنبال کم کردن واریانس ضرایب و فشرده کردن آن ها به سمت صفر هستیم. هر چه ضریب λ بزرگتر باشد، فشرده سازی بیشتر صورت می گیرد و ضرایب بسیار به صفر نزدیک می شوند. به ازای هر مقدار λ دسته ضرایب $\hat{\beta}_R^\lambda$ متفاوتی خواهیم داشت و باید مقدار λ با CV به درستی انتخاب شود. دقت کنید که محدودیت فشرده سازی به عرض از مبدأ اعمال نمی شود. مقدار عرض از مبدأ به نوعی نماینده میانگین خروجی ها در زمانی که ورودی ها صفر

هستند، می‌باشد. مقدار λ و β رابطه عکس دارند یعنی اگر ضریب λ زیاد شود به دنبال کم کردن β رهستیم و میخواهیم β هایی که تاثیر ندارند نزدیک صفر شوند چرا که هدف مینیمم کردن تابع $\sum_{j=1}^p \beta_j^2 + \lambda \text{RSS}$ است.

چرا ridge، روش کمترین مربعات خطأ را بهبود می‌دهد؟

مزیت روش ridge به کمترین مربعات خطأ در کاهش واریانس است. هر چه افزایش یابد، واریانس ضرایب کاهش می‌یابد. در شکل تاثیر λ بر میزان واریانس (منحنی سیز رنگ)، بایاس (منحنی سیاه رنگ) و نرخ خطای تست (منحنی بنفش رنگ) بر روی داده‌های شبیه‌سازی با $n = 50$ پارامتر و $p = 45$ داده آموزشی نشان داده شده است. وقتی λ صفر است و مدل مانند مدل کمترین مربعات خطأ است، بایاس صفر است اما واریانس ضرایب بالا است (یعنی تغییرات جزئی در ورودی منجر به تغییرات فاحش در خروجی می‌شود) بنابراین زیاد در واریانس به بھای افزایش در بایاس، منجر به کاهش زیاد خطای تست می‌شود. وقتی $n > p$ باشد، روش کمترین مربعات خطأ جواب یکتا ندارد، اما روش ridge می‌تواند با تحمل مقدار کمی بایاس، واریانس را به شدت کاهش دهد و جواب معقولی پیدا کند. علاوه بر این، روش ridge به لحاظ محاسباتی بسیار کارا است زیرا در روش انتخاب بهترین زیرمجموعه، می‌بایست 2^p مدل را بررسی کنیم اما در روش ridge به طور موازی می‌توان مدل را به ازای هر مقدار λ با محاسباتی به سرعت تخمین یک مدل ridge روش کمترین مربعات خطأ پیاده‌سازی کرد.



• روش Lasso

روش ridge یک عیب دارد و آن این است که در این روش، تمام متغیرها در مدل باقی می‌مانند و این باعث می‌شود تفسیرپذیری مدل دشوار باشد. ما انتظار داریم که مدل نهایی تنها شامل پارامترهای مهم باشد، اما در ridge تمام پارامترهای دیگر هم در مدل باقی ماند و به ندرت پارامتری از مدل حذف می‌شود. روش Lasso یک جایگزین جدید برای روش ridge است که در آن هدف تخمین ضریب به نحوی ات که تابع خطای زیر کمینه شود:

$$\text{RSS}_{\text{Lasso}} = \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij})^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (2)$$

این تابع خطأ بسیار شبیه به تابع خطای ridge است اما با این تفاوت که در بخش فشرده‌سازی به جای نرم L_2 از نرم L_1 استفاده شده است و همین تفاوت منجر به صفر شدن بسیاری از ضرایب و تولید مدل‌های اسپارس می‌شود. بنابراین با روش Lasso به نوعی انتخاب زیرمجموعه هم انجام می‌شود و این باعث می‌شود تفسیرپذیری مدل آسان‌تر شود.

• فرمول‌های دیگری از Ridge و Lasso

می‌توان نشان داد که تخمین ضرایب Lasso و Ridge معادل حل کردن مسائل زیر است:

$$\min_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 \right) \quad \text{to subject} \quad \sum_{j=1}^p |\beta_j| \leq s \quad (3)$$

$$p = 2 \rightarrow |\beta_1| + |\beta_2| \leq s$$

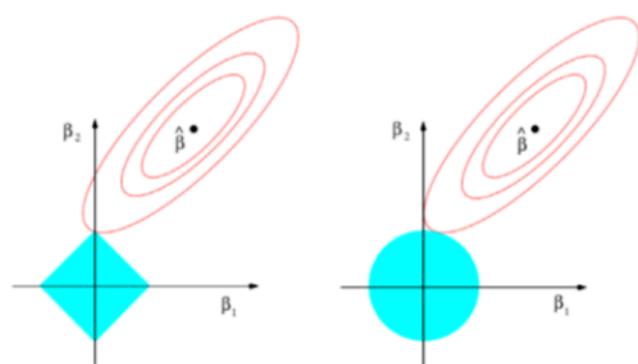
$$\min_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 \right) \quad \text{to subject} \quad \sum_{j=1}^p \beta_j \leq s \quad (4)$$

$$p = 2 \rightarrow \beta_1^2 + \beta_2^2 \leq s$$

به بیان دیگر، به ازای هر مقدار λ ، جواب (2) و (3) معادلند؛ همچنین جواب‌های (1) و (4) نیز معادلند. یعنی به ازای هر مقدار λ ، وجود دارد s ای که جواب این دو دسته مسئله با هم برابر باشند. علاوه بر این، می‌توان مسئله یافتن بهترین زیرمجموعه را به صورت زیر بازنویسی کرد:

$$\min_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 \right) \quad \text{to subject} \quad \sum_{j=1}^p I(\beta_j \neq 0) \leq s \quad (3)$$

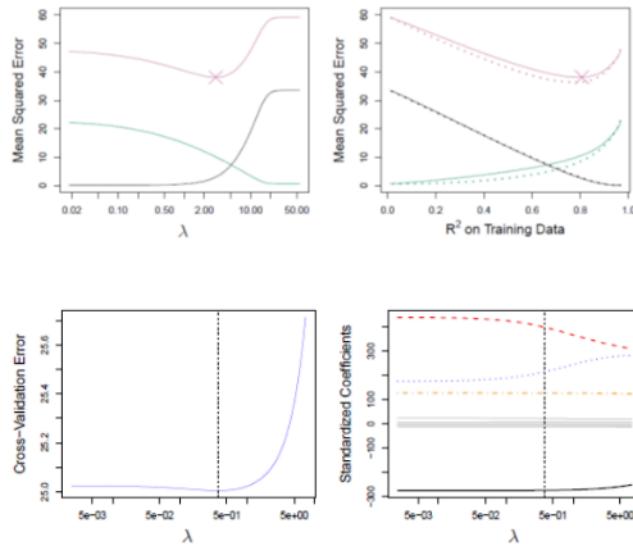
که در آن $s \leq \sum_{j=1}^p I(\beta_j \neq 0)$ به معنای آن است که تنها تا s از متغیرها می‌توانند ضرایب ناصرف داشته باشند. یا جواب بهینه این مسئله نیازمند بررسی $\binom{p}{s}$ حالت است. گرچه حل این مسئله به لحاظ محاسباتی غیرقابل حل است، اما حل مسائل (3) و (4) که جایگزین‌هایی برای این مسئله هستند، به راحتی امکان‌پذیر است. اگر s به اندازه کافی کوچک باشد، lasso در (3) مشابه انتخاب زیرمجموعه عمل می‌کند.



مقایسه ridge و lasso

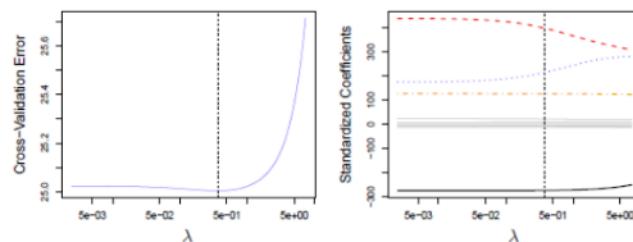
در دو شکل زیر دو مثال از اعمال ridge و lasso بر روی داده‌های شبیه‌سازی شده را مشاهده می‌کنید. در تصاویر سمت چپ تاثیر اعمال lasso بر بایاس، واریانس و MSE نشان داده شده است. همانند lasso Ridge، نیز با افزایش اندازی بایاس، باعث کاهش شدید واریانس می‌شود. نقاط بهینه در نمودارها با ضربدر نمایان شده است. در تصاویر سمت راست، مقایسه ridge و lasso (با خط ممتد) نشان داده شده است. در مثال اول هر دو روش دارای بایاس برابری هستند اما ridge واریانس کمتر و در نتیجه MSE کمتری دارد. در حالی‌که در مثال دوم بر عکس این ماجرا اتفاق افتاده و lasso عملکرد بهتری دارد. یک نکته که در این دو مثال بسیار مهم است این است که

در مثال اول، داده‌ها به نحوی تولید شده‌اند که تمام ۴۵ متغیر به خروجی مرتبط هستند (یعنی مقادیر واقعی هیچ کدام از ضرایب متغیرها صفر نیست) و چون lasso بعضی از متغیرها را حذف می‌کند، نتوانسته عملکرد خوبی در مثال اول داشته باشد. اما در مثال دوم، خروجی تنها به دو متغیر از بین ۴۵ متغیر وابسته است و بر عکس lasso توانسته بهتر عمل کند. این دو مثال نشان می‌دهد که هیچ کدام از این دو روش به طور عام بر دیگری ارجحیت ندارد. به طور کلی در شرایطی که تعداد کمی از متغیرها به خروجی مرتبط هستند، lasso و زمانی که خروجی به پارامترهای متعددی وابسته است، ridge بهتر عمل می‌کند. البته این اطلاعات معمولاً مجھول است و باید از cross-validation برای کشف آن‌ها کمک گرفت.



انتخاب پارامتر تنظیم‌کننده

در روش‌های فشرده‌سازی باید پارامتر λ در رابطه‌های (۱) و (۲) یا پارامتر s در رابطه‌های (۳) و (۴) را تعیین کنیم. برای تعیین مقدار این پارامترها از روش cross-validation استفاده می‌کنیم. به ازای هر مقدار مختلف λ یک بار cross-validation می‌زنیم و نهایتاً مقداری را برای λ انتخاب می‌کنیم که مدل آن دارای خطای کمتری باشد.



شکل بالا اعمال LOOCV بر روی روش ridge برای مثال داده‌های اعتبار کارت بانکی را نشان می‌دهد. خطچین عمودی نشان‌گر مقدار بهینه برای انتخاب λ است. در تصویر سمت راست نیز ضرایبی که در به ازای مقادیر مختلف λ تخمین زده شده را نشان می‌دهد. خطچین عمودی در این تصویر نمایان‌گر ضرایب بهینه در روش ridge هستند. همین تکنیک را می‌توان برای lasso نیز به کار برد.

دلیل استفاده از نرم L_1 در رگرسیون لاسو و نرم L_2 در رگرسیون ریج در رگرسیون خطی، هدف یافتن ضرایب β است که تابع هزینه زیر را کمینه کند:

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^n (y_i - X_i \beta)^2 \right\}$$

برای جلوگیری از بیش‌پرازش، از منظم‌سازی (Regularization) استفاده می‌کنیم که شامل افزودن یک جمله پنالتی به تابع هزینه است. در رگرسیون لاسو و ریج، این جمله پنالتی به ترتیب شامل نرم‌های L_1 و L_2 ضرایب است.

رگرسیون لاسو و نرم L_1

مسئله بهینه‌سازی:

$$\min_{\beta} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - X_i \beta)^2 + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

دلیل ریاضی استفاده از نرم L_1 :

۱. شرایط بهینگی (KKT Conditions):

برای مسئله بهینه‌سازی محدب با تابع هدف:

$$L(\beta) = \frac{1}{n} \|y - X\beta\|^2 + \lambda \|\beta\|_1$$

شرایط بهینگی به صورت زیر است:

برای هر j :

$$\begin{cases} \frac{1}{n} X_j^T (y - X\beta) = \lambda \cdot \text{sgn}(\beta_j) & \beta_j \neq 0 \\ |\frac{1}{n} X_j^T (y - X\beta)| \leq \lambda & \beta_j = 0 \end{cases}$$

این شرایط نشان می‌دهد که اگر مقدار مطلق مشتق جزئی تابع خطأ نسبت به β_j کمتر از λ ، یعنی:

$$\left| \frac{1}{n} X_j^T (y - X\beta) \right| < \lambda$$

باشد، آنگاه $\beta_j = 0$ خواهد بود.

۲. عدم مشتق‌پذیری در صفر:

نم L_1 در نقطه $\beta_j = 0$ مشتق‌پذیر نیست، زیرا:

$$\frac{d}{d\beta_j} |\beta_j| = \begin{cases} 1 & \beta_j > 0 \\ -1 & \beta_j < 0 \\ \text{نامعین} & \beta_j = 0 \end{cases}$$

این عدم مشتق‌پذیری به بهینه‌سازی اجازه می‌دهد تا برخی ضرایب دقیقاً صفر شوند.

۳. تفسیر هندسی:

منطقه مجاز (Constraint Region) مربوط به نرم L_1 یک چندوجهی با گوششای تیز است. خطوط همتراز تابع خطأ (بیضی‌ها) ممکن است در گوششای این چندوجهی با منطقه مجاز برخورد کنند، که منجر به ضرایب صفر می‌شود.

رگرسیون ریج و نرم L_2

مسئله بهینه‌سازی:

$$\min_{\beta} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - X_i \beta)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right\}$$

دلیل ریاضی استفاده از نرم L_2 :

۱. مشتق‌پذیری در همه نقاط:

نرم L_2 در همه جا مشتق‌پذیر است:

$$\frac{d}{d\beta_j} \beta_j^2 = 2\beta_j$$

این مشتق‌پذیری منجر به حل بسته (Closed-form Solution) برای رگرسیون ریج می‌شود.

۲. شرایط بهینگی:

مشتق تابع هزینه نسبت به β :

$$\frac{\partial L}{\partial \beta} = -\frac{1}{n} X^T (y - X\beta) + 2\lambda\beta = 0$$

با حل این معادله، داریم:

$$(X^T X + n\lambda I)\beta = X^T y$$

که I ماتریس همانی است و $X^T X + n\lambda I$ معکوس‌پذیر است.

۳. تفسیر هندسی:

منطقه مجاز مربوط به نرم L_2 یک کره (Hypersphere) است. خطوط همتراز تابع خطا در نقاطی با ضرایب کوچک اما غیرصفر با منطقه مجاز تماس پیدا می‌کنند.

— مقایسه و نتیجه‌گیری

— صفر شدن ضرایب در لاسو:

— به دلیل عدم مشتق‌پذیری نرم L_1 در صفر و شرایط بهینگی، برخی ضرایب دقیقاً صفر می‌شوند. — این امر به ایجاد مدل‌های پراکنده و انتخاب ویژگی کمک می‌کند.

— کوچکسازی ضرایب در ریج:

— به دلیل مشتق‌پذیری نرم L_2 در همه جا، ضرایب به طور پیوسته به سمت صفر کشیده می‌شوند اما به صفر نمی‌رسند. — این امر به کاهش واریانس مدل و مقابله با چندخطی‌بودن کمک می‌کند.

نتیجه‌گیری نهایی

استفاده از نرم L_1 در رگرسیون لاسو به دلیل ویژگی‌های ریاضی آن منجر به مدل‌هایی با تعداد کمی ویژگی‌های غیرصفر می‌شود، زیرا شرایط بهینگی اجازه صفر شدن ضرایب را می‌دهد. از سوی دیگر، استفاده از نرم L_2 در رگرسیون ریج باعث کوچکسازی ضرایب بدون صفر کردن آن‌ها می‌شود، زیرا مشتق‌پذیری نرم L_2 منجر به حل پیوسته ضرایب می‌شود.

به طور خلاصه، دلیل ریاضی استفاده از نرم‌های L_1 و L_2 در رگرسیون لاسو و ریج به تأثیر آن‌ها بر روی ضرایب مدل و ویژگی‌های

Elastic Net Regularization

یک روش رگرسیون منظم‌سازی است که به طور خطی ویژگی‌های روش‌های Lasso و Ridge را ترکیب می‌کند. در واقع، این روش از ترکیب دو جریمه L_1 و L_2 استفاده می‌کند. معادله هدف در روش Elastic Net به شکل زیر است:

$$\min_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j X_{ij} \right)^2 + \lambda_1 \sum_{j=1}^p |\beta_j| + \lambda_2 \sum_{j=1}^p \beta_j^2 \right)$$

در اینجا: - λ_1 جریمه L_1 است که باعث می‌شود ضرایب کوچک به صفر نزدیک شوند و ویژگی‌های کم اهمیت حذف شوند (مانند روش Lasso).

- λ_2 جریمه L_2 است که باعث کوچک شدن ضرایب بدون صفر شدن آنها می‌شود (مانند روش Ridge).

ترکیب خواص Lasso و Ridge

- ویژگی‌های Lasso: قادر است برخی ضرایب را کاملاً به صفر برساند و در نتیجه ویژگی‌های کم اهمیت حذف کند. این امر به مدل کمک می‌کند که ساده‌تر و قابل فهم‌تر باشد و از نظر تفسیرپذیری بهتری داشته باشد.

- ویژگی‌های Ridge : Ridge کمک می‌کند تا همبستگی بین ویژگی‌های ورودی را بهتر مدیریت کند. در صورت وجود همبستگی زیاد بین ویژگی‌ها، Ridge ضرایب را به طور همزمان کوچک می‌کند به طوری که اهمیت همه ویژگی‌ها حفظ شود.

سناریوهای مفید برای Elastic Net

- داده‌های با ابعاد بالا و همبستگی بالا: زمانی که تعداد ویژگی‌ها بیشتر از نمونه‌ها است و همبستگی بین ویژگی‌ها وجود دارد، Elastic Net می‌تواند بسیار مفید باشد. ترکیب جریمه‌های L_1 و L_2 کمک می‌کند تا هم ویژگی‌های غیرضروری حذف شوند و هم اثرات همبستگی بین ویژگی‌ها مدیریت شود. مدل باید همزمان از مزایای انتخاب ویژگی‌های لasso و کوچک‌سازی ضرایب ریج استفاده کند. این روش به دلیل ترکیب خواص هر دو رگرسیون، تعادل خوبی بین انتخاب ویژگی و کاهش همبستگی ویژگی‌ها فراهم می‌کند.

- داده‌های اسپارس (Sparse Data): در مواردی که داده‌ها دارای تعداد زیادی ویژگی هستند اما بسیاری از آنها بی‌اثر یا کم‌اثر هستند، با استفاده از جریمه L_1 ویژگی‌های غیرمفید را حذف می‌کند.