مبانی یادگیری ماشین

مولف: احمد پورامینی

انتشارات ناقوس
NAGHOOS
PUBLICATION

تقدیم به همسر مهربانم . . .

مقدمه مؤلف

یادگیری ماشین دارای حوزه نظری و عملی مجزایی است و شامل مباحث گسترده‌ای است. کتاب‌های مختلفی در این زمینه منتشر شده است که هر کدام بر روی حوزه و یا مبحث خاصی تمرکز دارد. اگر شما دو کتاب با رویکردهای متفاوت را بخوانید، ممکن است تصور کنید که در حال خواندن موضوعات جداگانه‌ای هستید. در این کتاب ما سعی کرده‌ایم تعادلی میان نظریه و عمل برقرار سازیم و موضوعات اصلی که هر دانشجو و یا محققی باید در زمینه نظری و عملی این شاخه بداند را پوشش دهیم.

این کتاب شامل پنج فصل است که در فصل ۱ با ارائه برخی مثال‌های عملی و مفاهیم ریاضی، چارچوب مشترک مسائل یادگیری ماشین ارائه می‌شود. فصل ۲ نظری‌ترین فصل این کتاب است. در این فصل نظریه تعمیم پوشش داده می شود، این نظریه نقشی محوری در تحلیل مسائل یادگیری ماشین بازی می‌کند و ما سعی کرده‌ایم فهم آن را برای طیف گسترده‌ای از خوانندگان آسان کنیم. در فصل ۳ با مدل‌های خطی و برخی الگوریتم‌های کاربردی در این زمینه آشنا می‌شوید. مدل‌های خطی یکی از اصلی‌ترین و رایج‌ترین ابزارها در حوزه یادگیری ماشین محسوب می‌شوند. در فصل ۴ با برخی ملاحظات مهم عملی در بکارگیری الگوریتم‌های یادگیری و چگونگی انتخاب یک مدل مناسب برای یک مسئله داده شده آشنا می‌شوید. و در آخر در فصل ۵ با مرور برخی اصول کلی در جمع‌آوری و پردازش داده‌های موردنیاز مسائل یادگیری، دیدی جامع نسبت به ماهیت کار ارائه می‌شود. امیدواریم که خواننده با مطالعه فصل‌های مختلف این کتاب با اصول و مبانی یادگیری ماشین آشنا شود.

هدف اصلی ما در این ویرایش ارائه یک دید جامع و کلی نسبت به حوزه یادگیری ماشین است و با استفاده از نگارشی داستان‌وار سعی کرده‌ایم خواننده را تا انتها مجذوب بحث نگاه داریم. ما خواندن این کتاب را به کلیه علاقه‌مندان این حوزه توصیه می‌کنیم. این کتاب همینطور

می تواند به عنوان یک کتاب درسی نیز در دانشگاه‌ها تدریس شود و در صورت استقبال اساتید در ویرایش‌های بعدی مسائل و تمریناتی به آن اضافه خواهیم کرد. لطفاً با ارسال نظرات و پیشنهادات خود در مورد این کتاب از طریق رایانامه مؤلف و یا وبسایتی که آدرس آن در انتهای متن آمده است، ما را در بهبود مطالب کتاب یاری کنید.

در پایان قصد دارم از خانواده و همسرم که پشتیبان و مشوق بنده در این کار بودند تشکر کنم و به خاطر صبر و تحملشان در طول تألیف این کتاب تشکر کنم.

آدرس وبسایت پشتیانی کتاب: selfreading.com/fml

احمد پورامینی
pouramini@sirjantech.ac.ir
دانشگاه صنعتی سیرجان
خرداد ۱۳۹۵

فهرست مطالب

فصل اول

مسئله یادگیری

۱-۱ مقدمه

۱-۱-۱ تاریخچه یادگیری ماشین

برای درک بهتر معنی و کاربرد "یادگیری ماشین" اجازه دهید نگاهی مختصر به تاریخچه این علم بیندازیم. یادگیری ماشین از زیرشاخه‌های علوم رایانه‌ای است که در حوزه هوش مصنوعی تکامل یافته است. گاهی اوقات ممکن است یادگیری ماشین و هوش مصنوعی به‌جای یکدیگر استفاده شوند. اما درواقع آنها دو شاخه مجزا اما مرتبط به یکدیگر هستند.

یکی از اهداف هوش مصنوعی ایجاد ماشینی هوشمند همانند انسان است. درنتیجه چنین ماشینی نیازمند توانایی یادگیری است. به این منظور، برخی دانشمندان سعی کردند از نحوه یادگیری مغز انسان تقلید کنند. این تلاش‌ها منجر به ظهور "شبکه‌های عصبی مصنوعی" شد. هرچند بعدها مشخص شد که شبکه عصبی مصنوعی اختراع دوباره همان "مدل خطی تعمیم‌یافته[۱]" در علم آمار است.

با ظهور رویکردهای مبتنی بر دانش در دهه ۶۰، شکافی میان هوش مصنوعی و یادگیری ماشین ایجاد شد و برای مدتی استفاده از روش‌های آماری در هوش مصنوعی دچار افول شد. هرچند این خط خارج از هوش مصنوعی در حوزه "شناسایی الگو" و "بازیابی اطلاعات" دنبال شد. در همین زمان نظریات جدیدی در مورد یادگیری در شبکه‌های عصبی مصنوعی توسط دانشمندانی مانند هاپفیلد مطرح شد.

در دهه ۹۰، یادگیری ماشین دوباره مورد توجه قرار گرفت. اما این بار مانند بسیاری دیگر

[۱] Generalized Linear Model

از زیرشاخه‌های هوش مصنوعی توجه خود را از دستیابی به هوش مصنوعی معطوف به حل مسائل کاربردی در حوزه‌های مختلف نمود. همین‌طور از رویکردهای نمادمحور [۱] مانند منطق و جستجو فاصله گرفت و تمرکز خود را بر روی روش‌های آماری قرار داد. در این رابطه، ظهور اینترنت و تولید گسترده داده‌های دیجیتال کمک زیادی به توسعه یادگیری ماشین کرد. درواقع امروزه یادگیری ماشین ارتباط نزدیک‌تری به علم آمار و داده‌کاوی دارد تا به هوش مصنوعی. با این تفاوت که تمرکز داده‌کاوی بر کشف الگوهای ناشناخته و استخراج دانش از حجم بالایی از داده‌ها است، درحالی‌که یادگیری ماشین سعی دارد با استفاده از ویژگی‌های شناخته‌شده در یک نمونه از داده‌ها، پیش‌بینی‌هایی در مورد داده‌های مشابه انجام دهد.

۱-۱-۲ یادگیری ماشین چیست

اگر به یک کودک سه‌ساله تصویر یک درخت را نشان دهید و از او سؤال کنید که در تصویر چه می‌بیند، به‌احتمال‌زیاد پاسخ درست را خواهید شنید. حال اگر از یک فرد سی‌ساله بخواهید درخت را تعریف کند، ممکن است پاسخ مبهمی دریافت کنید. درواقع ما مفهوم درخت را با مطالعه تعریف ریاضی آن یاد نمی‌گیریم؛ بلکه آن را با مشاهده درختان پیرامون خود یاد می‌گیریم. به‌عبارت‌دیگر ما آن را "از روی داده" یاد می‌گیریم. ما مثال‌هایی از درختان مختلف را می‌بینیم و مفهوم آن را به موارد مشابه تعمیم می‌دهیم. در این تعریف یادگیری به معنی استخراج یک مفهوم یا رابطه از روی یک سری مشاهدات و تعمیم آن به موارد مشابه دیگر است.

در حوزه هوش مصنوعی و یادگیری محاسباتی می‌توان به یک تعریف متداول از آرتور ساموئل در سال ۱۹۵۰ اشاره کرد: "شاخه‌ای علمی که به رایانه‌ها توانایی یادگیری بدون برنامه‌ریزی شدن به شکل صریح را می‌دهد" .

طبق این تعریف، یادگیری ماشین به دنبال مطالعه و ایجاد الگوریتم‌هایی است که به جای پیروی از دستورات ثابت و برنامه‌ریزی شده قادرند از روی یک مجموعه نمونه از مشاهدات ورودی چیزهایی را یادگرفته (مُدل یا رابطه‌ای ایجاد کرده) و پیش‌بینی‌ها و یا تصمیماتی را روی داده‌های مشابه (سایر داده‌های فضای ورودی) تولید کنند. این روش زمانی استفاده می‌شود که ما راه‌حلی تحلیلی در اختیار نداریم، اما داده‌های کافی برای ساخت یک راه‌حل تجربی موجود هستند. این وضعیت را می‌توان در مسائل زیادی مشاهده کرد. امروزه یادگیری

[۱] symbolic

ماشین یکی از پرکاربردترین روش‌ها در شاخه‌های مختلف علوم، مهندسی، اقتصاد و دیگر حوزه‌هاست.

در این فصل، ابتدا مسائلی از یادگیری ماشین را ارائه می‌کنیم و با توجه به این مثال‌ها این مفهوم را فرمول‌بندی می‌کنیم. در این راستا، مفاهیم اصلی مرتبط با یادگیری و مدل‌های مختلف توسعه داده‌شده در این زمینه را مورد بحث قرار می‌دهیم.

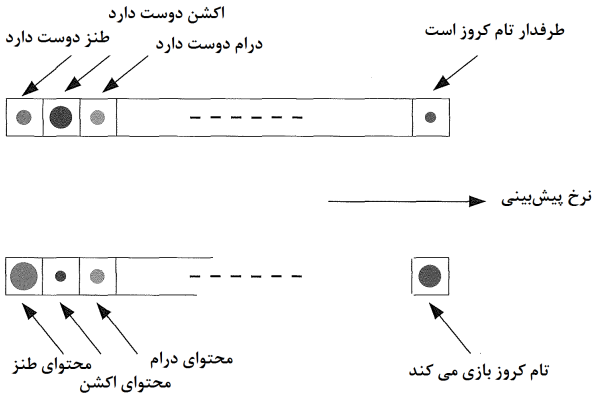
۱-۲ چارچوب مسئله یادگیری

نقطه اشتراک مسائلی مانند پیش‌بینی‌های اقتصادی، تشخیص علت بیماری، بینایی ماشین و موتورهای جستجو چیست؟ همه این مسائل از یادگیری از داده بهره می‌برند. فهرست چنین کاربردهایی بسیار طولانی است. اجازه دهید بحث را با یک مثال از زندگی واقعی شروع کنیم و نشان دهیم که یادگیری از داده به چه شکل صورت می‌گیرد.

مسئله امتیازدهی به یک فیلم را در نظر بگیرید. اگر شما صاحب یک شرکت اجاره فیلم باشید، این مسئله برایتان مهم است. زیرا قصد دارید بدانید هر مشتری احتمالاً به چه نوع فیلمی علاقه‌مند است و فیلم مناسب را به وی پیشنهاد دهید.

مشکل اصلی در این مسئله این است که معیارهای افراد برای امتیازدهی به یک فیلم بسیار پیچیده و متنوع است. مدل‌سازی این فرآیند کار آسانی نیست و شاید هرگز نتوان به یک راه‌حل تحلیلی مناسب رسید. در این شرایط ممکن است امتیازاتی که مشتریان قبلاً به فیلم‌های مختلف داده‌اند موجود باشد. این اطلاعات می‌توانند چیزهای زیادی در مورد سلایق مشتریان فاش کنند. درنتیجه ممکن است از این طریق بتوانیم به یک راه‌حل تجربی خوب برسیم. معمولاً حجم زیادی از این نوع داده‌ها در شرکت‌های اجاره دهنده فیلم موجود است، زیرا آنها اغلب از مشتریان خود می‌خواهند که پس از دیدن هر فیلم به آن امتیاز دهند.

شکل ۱-۱ یک رویکرد متداول برای بهره‌برداری از این اطلاعات را نشان می‌دهد. مطابق این رویکرد، هر فیلم را می‌توان به صورت آرایه‌ای از ویژگی‌های مختلف نشان داد. برای مثال اینکه چقدر حاوی طنز است، چقدر صحنه‌هایش پیچیده است، چقدر هنرپیشه اصلی معروف یا خوش‌ظاهر است و غیره. به همین شکل هر تماشاگر را نیز با آرایه‌ای از ویژگی‌های متناظر نمایش می‌دهیم، چقدر از طنز لذت می‌برد، چقدر از صحنه‌های ساده یا پیچیده لذت می‌برد، چقدر نام یا ظاهر هنرپیشه اصلی برایش مهم است و غیره. حال اینکه یک مشتری چه امتیازی



شکل ۱-۱: مدلی برای نحوه امتیازدهی به فیلم ها توسط یک مشتری

به یک فیلم خواهد داد می‌تواند وابسته به میزان انطباق ویژگی‌های فیلم با ویژگی‌های آن مشتری باشد. برای مثال اگر موضوع یک فیلم کاملاً طنز باشد و مشتری به طنز علاقه نداشته باشد، شانس اینکه امتیاز بالایی به فیلم بدهد بسیار کم است. اگر ده‌ها مورد از این عوامل که وجوه مختلف یک فیلم و سلایق یک مشتری را نشان می‌دهند را در کنار هم قرار دهیم، نتیجه‌گیری بر اساس انطباق این عوامل می‌تواند یک پیش‌بینی کننده خوب از امتیازی باشد که یک مشتری به یک فیلم می‌دهد.

مزیت یادگیری ماشین این است که کل این فرایند می‌تواند به شکل خودکار و بدون نیاز به تحلیل سلایق مشتریان و یا تحلیل محتوای فیلم‌ها صورت گیرد. برای این کار الگوریتم یادگیری طی یک فرآیند مهندسی معکوس سعی می‌کند از روی امتیازاتی که مشتریان درگذشته به فیلم‌ها داده‌اند، وزن کلیه ویژگی‌های مشتریان و فیلم‌ها را به دست آورد. این الگوریتم، با یک سری وزن‌های تصادفی برای ویژگی‌ها شروع به کار می‌کند، سپس در یک فرآیند تکراری این وزن‌ها را به شکلی تنظیم می‌کند که با امتیازاتی که مشتریان به فیلم‌های مختلف داده‌اند، سازگار شوند. وزنی که نهایتاً برای هر ویژگی به دست می‌آید ممکن است خیلی ملموس و قابل‌درک نباشد. بااین‌حال، الگوریتم سعی می‌کند بهترین روش برای پیش‌بینی امتیازی که یک مشتری به یک فیلم می‌دهد را پیدا کند، اما لزوماً توضیح نمی‌دهد که این امر چگونه انجام می‌شود.

۱-۲-۱ اجزای یادگیری

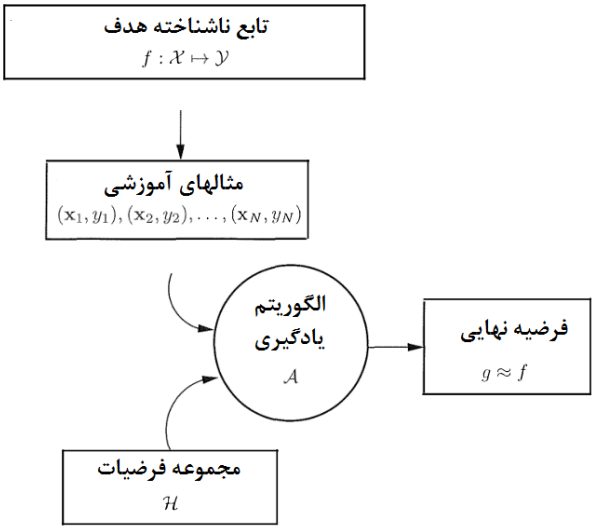
مثال قبل اساس روش یادگیری ماشین را نشان می‌دهد. به همین شکل می‌توان مثال‌های دیگری از حوزه‌های مختلف را مطرح کرد. برای اینکه هسته مشترک این مسائل را توضیح دهیم، ما یک مثال نوعی را انتخاب می‌کنیم و از آن برای توصیف اجزاء مختلف یادگیری استفاده می‌کنیم. شما می‌توانید این اجزاء را به مسائل مشابه دیگر تعمیم دهید. اجازه دهید به این منظور از مثال صدور کارت اعتباری استفاده کنیم.

فرض کنید یک بانک روزانه هزاران درخواست صدور کارت اعتباری را دریافت می‌کند. بانک قصد دارد پروسه ارزیابی این درخواست‌ها را خودکار کند. مانند مثال قبل هیچ فرمول جادویی برای تائید اعتبار یک مشتری وجود ندارد، اما داده‌های مرتبط زیادی وجود دارند. در این شرایط ممکن است یادگیری ماشین گزینه مناسبی باشد. به این صورت که می‌توان از روی سوابق مشتریان قبلی، یک راه‌حل تجربی مناسب را برای تائید اعتبار مشتریان جدید پیدا کرد.

پرونده هر مشتری حاوی یک سری اطلاعات مانند میزان دستمزد، سابقه کار، وام‌های عمده و غیره است که می‌توان از آنها برای ارزیابی اعتبار یک مشتری استفاده کرد. همین‌طور این موضوع که آیا تائید اعتبار آن مشتری عمل درستی بوده است یا خیر در پرونده سابقه مشتری وجود دارد. به این معنی که آیا مشتری فرد خوش‌حسابی بوده و توانسته است پولی به بانک برگرداند یا خیر. این داده‌ها می‌توانند برای ساخت یک فرمول مناسب برای تائید اعتبار یک مشتری استفاده شوند و می‌توان از این فرمول برای تائید درخواست‌های جدید استفاده کرد.

اجازه دهید ما این مسئله را با استفاده از علائم و اسامی فرموله کنیم. یک ورودی $\mathbf{x}$ وجود دارد که شامل اطلاعات یک مشتری است. یک تابع هدف ناشناخته به شکل $f : \mathcal{X} \to \mathcal{Y}$ وجود دارد که همان فرمول ایدئال برای تائید اعتبار مشتری است و در آن $\mathcal{X}$ نشان‌دهنده فضای ورودی (مجموعه تمام مشتریان) و $\mathcal{Y}$ نشان‌دهنده فضای خروجی است (مجموعه تمام خروجی‌های ممکن که در این مثال تنها شامل دو مقدار بلی و خیر است). همین‌طور یک مجموعه داده $\mathcal{D}$ از مثال‌های ورودی خروجی به شکل $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ وجود دارد که در آن هر $\mathbf{x}$ متناظر با پرونده یک مشتری قبلی و هر y متناظر با درست یا غلط بودن تصمیم اخذشده مبنی بر تائید آن مشتری است. درواقع برای $n = 1 \ldots N$، $y_n = f(\mathbf{x}_n)$. به این مثال‌ها معمولاً نقطه‌داده [۱] گفته می‌شود. در انتها، الگوریتم یادگیری را داریم که با استفاده از مجموعه داده $\mathcal{D}$ سعی

[۱] data point



شکل ۱-۲: چارچوب پایه مسئله یادگیری

می‌کند با تابع $g : \mathcal{X} \to \mathcal{Y}$، تابع f را تقریب بزند. الگوریتم یادگیری تابع g را از مجموعه‌ای از توابع کاندید که به آنها مجموعه فرضیات $\mathcal{H}$ گفته می‌شود انتخاب می‌کند. برای مثال $\mathcal{H}$ می‌تواند مجموعه همه توابع خطی باشد. در آن صورت الگوریتم یادگیری بهترین تابع خطی که با داده‌های مجموعه $\mathcal{D}$ همخوانی دارد را انتخاب می‌کند.

زمانی که یک مشتری جدید برای صدور کارت اعتباری درخواست می‌دهد، بانک تصمیم خود را بر مبنای فرضیه g (فرضیه‌ای که الگوریتم یادگیری انتخاب کرده است) قرار می‌دهد و نه بر مبنای تابع f که تابع هدف ایدئال است و همچنان برای ما ناشناخته است. این تصمیم به میزانی که g تابع f را تقریب می‌زند، می‌تواند تصمیم درستی باشند. برای رسیدن به یک تقریب خوب، الگوریتم یادگیری فرضیه‌ای را انتخاب می‌کند که بیشترین انطباق با f را روی مثال‌های آموزشی دارد، به این امید که این فرضیه برای مشتریان جدید نیز به‌خوبی با f همخوانی خواهد داشت. اینکه این امید چقدر قابل توجیه بوده است، چیزی است که بعداً مشخص می‌شود.

شکل ۱-۲ اجزای مسئله یادگیری را مطابق آنچه بیان شد نمایش می‌دهد. ما از چارچوبی که در این شکل است به‌عنوان تعریفمان از مسئله یادگیری استفاده خواهیم کرد. البته در صورت نیاز یک سری اصلاحات را در نظر خواهیم گرفت، اما اصل مسئله یادگیری همچنان همین باقی خواهد ماند: هدفی وجود دارد که قرار است یادگرفته شود؛ این هدف برای ما ناشناخته است؛

ما یک سری مثال داریم که توسط تابع هدف تولید شده است؛ مسئله یادگیری از این مثال‌ها برای یافتن بهترین فرضیه‌ای که تابع هدف را تقریب می‌زند استفاده می‌کند.

۱-۲-۲ یک مدل ساده یادگیری

اجازه دهید به اجزاء مختلف شکل ۱-۲ نگاه دوباره‌ای بیندازیم. با فرض یک مسئله یادگیری داده‌شده، تابع هدف و مثال‌های یادگیری توسط مسئله دیکته می‌شوند؛ اما انتخاب الگوریتم یادگیری و مجموعه فرضیات در اختیار خود ما هستند. درواقع این‌ها ابزارهای اصلی ما برای یافتن راه‌حل مناسب هستند. مجموعه فرضیات و الگوریتم یادگیری به شکل غیررسمی به‌عنوان "مدل یادگیری" شناخته می‌شوند. در اینجا یک مدل ساده را ارائه می‌کنیم. فرض کنید $\mathcal{X} = \mathbb{R}^d$ فضای ورودی باشد که در آن $\mathbb{R}^d$ یک فضای اقلیدسی d بعدی است. همینطور فرض کنید $y = \{+1, -1\}$ فضای خروجی باشد که نشان‌دهنده یک تصمیم بله و خیر است. در مثال کارت اعتباری مختصه‌های مختلف بردار ورودی $\mathsf{x} \in \mathbb{R}^d$ متناظرند با حقوق، سال‌های اشتغال، وام‌های عمده و بقیه فیلدهای موجود در پرونده درخواست کارت و خروجی دودویی y متناظر با تائید یا عدم تائید اعتبار مشتری برای دریافت کارت است. ما مجموعه فرضیات $\mathcal{H}$ را توسط یک فرم تابعی که تمام فرضیات $h \in \mathcal{H}$ در آن مشترک هستند نشان می‌دهیم. فرم تابعی $h(\mathsf{x})$ که ما در اینجا انتخاب کردیم وزن‌های متفاوتی را به مؤلفه‌های مختلف x نسبت می‌دهد که هر یک منعکس‌کننده اهمیت نسبی یک مؤلفه در تصمیم تائید اعتبار یک مشتری است. این مؤلفه‌های وزن‌دار با هم ترکیب می‌شوند تا یک نمره کلی اعتبار را تولید کنند. این نمره با یک حد آستانه مقایسه می‌شود. اگر نمره از حد آستانه بیشتر شود، اعتبار مشتری تائید می‌شود؛ در غیر این صورت درخواست مشتری رد می‌شود.

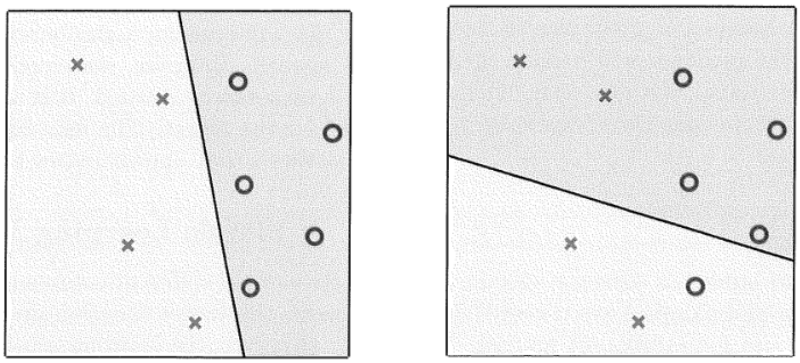
اگر $\sum_{i=1}^{d} w_i x_i > threshold$ قبول کن

اگر $\sum_{i=1}^{d} w_i x_i < threshold$ رد کن

این فرمول را می‌توان به شکل زیر نوشت:

$$h(\mathsf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + b\right) \qquad (۱-۱)$$

که در آن $x_1, x_2 \ldots, x_d$ اجزاء مؤلفه بردار x هستند. $h(\mathsf{x}) = +1$ یعنی اعتبار تائید شده و $h(\mathsf{x}) = -1$ به معنی رد اعتبار است. تابع علامت $\text{sign}(s)$ برای $s > 0$ برابر با $+1$ و برای



(الف) داده های طبقه بندی نشده

(ب) داده های طبقه بندی شده

شکل ۱-۳: طبقه بندی پرسپترون در یک فضای دو بعدی الف) برخی مثالها به درستی طبقه بندی نشدند. ب) فرضیه نهایی که مثالها را به شکل ایدئال طبقه بندی می کند.

$s < 0$ برابر با -1 است. وزن‌های بکار رفته عبارتند از $w_1, \ldots, w_d$ و حد آستانه توسط ترم بایاس b مشخص می‌شود. بنابراین طبق این معادله، چنانچه حاصل جمع بزرگ‌تر از $-b$ شود، اعتبار تائید می‌شود. این مدل عموماً مدل "پرسپترون" [۱] خوانده می‌شود که مدلی شناخته‌شده در حوزه شبکه‌های عصبی مصنوعی است.

الگوریتم یادگیری مجموعه فرضیات $\mathcal{H}$ را با هدف یافتن وزن‌ها و مقدار بایاسی که به‌خوبی روی مجموعه داده $\mathcal{D}$ کار می‌کنند جستجو می‌کند. برخی از وزن‌های $w_1, \ldots, w_d$ ممکن است نهایتاً مقدار منفی بگیرند که به معنی تأثیر معکوس آنها روی تصمیم تائید اعتبار است. برای مثال وزن متغیر وام عمده باید نهایتاً منفی شود، چون وام بیشتر نشانه خوبی برای اعتبار نیست. همین‌طور مقدار بایاس b ممکن است نهایتاً بزرگ یا کوچک باشد که نشاندهنده میزان سخت‌گیری بانک در تائید اعتبار است. مقادیر بهینه وزن‌ها و بایاس فرضیه نهایی $g \in \mathcal{H}$ که الگوریتم تولید می‌کند را مشخص می‌کنند.

شکل ۱-۳ نحوه عملکرد یک پرسپترون در یک فضای دوبعدی را نشان می‌دهد. این صفحه توسط یک خط به دو ناحیه تقسیم می‌شود که عبارت‌اند از ناحیه تصمیم $+1$ و ناحیه تصمیم -1. مقادیر متفاوت پارامترهای w_1، w_2 و b، خطهای متفاوت $w_1x_1 + w_2x_2 + b = 0$ را مشخص می‌کنند. اگر مجموعه داده به شکل خطی تفکیک‌پذیر[۲] باشد، انتخابی برای مقادیر این

[۱] perceptron

[۲] linearly separable

پارامترها وجود دارد که طبق آنها تمام مثال‌های آموزشی به درستی طبقه‌بندی می‌شوند.

برای ساده‌سازی فرمول پرسپترون ما ترم بایاس b را به‌عنوان وزن $w_0 = b$ در نظر می‌گیریم و آن را با بقیه وزن‌ها ادغام می‌کنیم. سپس کلیه وزن‌ها را به شکل یک بردار وزن $\mathbf{w} = [w_0, w_1 \dots, w_d]^{\mathrm{T}}$ نشان می‌دهیم که در آن $^{\mathrm{T}}$ نشان‌دهنده ترانهاده ماتریس است (چون ماتریس وزن ستونی است). به همین شکل ورودی را می‌توان توسط بردار ستونی $\mathbf{x} = [x_0, x_1 \dots, x_d]^{\mathrm{T}}$ نمایش داد . در این بردار x_0 که ضریب w_0 است را به عنوان مقدار ثابت $x_0 = 1$ در نظر می‌گیریم. با توجه به این فرضیات، فضای ورودی به شکل زیر است:

$$\mathcal{X} = \{1\} \times \mathbb{R}^d = \{[x_0, x_1 \dots, x_d]^{\mathrm{T}} \mid x_0 = 1, x_1 \in \mathbb{R}, \dots, x_d \in \mathbb{R}\}$$

با توجه به اینکه $\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{d} w_i x_i$ معادله ۱-۱ را می توان به شکل برداری زیر نوشت:

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \qquad (۱-۲)$$

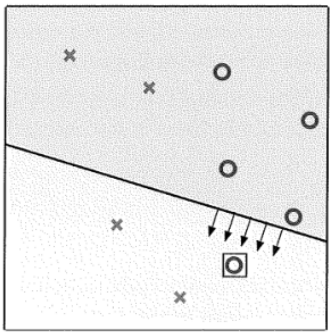
در ادامه روش یادگیری پرسپترون را ارائه می‌کنیم. این الگوریتم با توجه به داده‌ها، سعی می‌کند بردار $\mathbf{w}$ بهینه را بیابد. اجازه دهید فرض کنیم که مجموعه داده به شکل خطی تفکیک‌پذیر است. به این معنی که یک بردار $\mathbf{w}$ وجود دارد که اگر در فرمول جایگذاری شود ، روی تمام مثال‌های ورودی به تصمیم درست $h(\mathbf{x}_n) = y_n$ می‌رسیم (همانند شکل ۱-۳).

الگوریتم یادگیری موردنظر سعی می‌کند با استفاده از یک چرخه تکراری، $\mathbf{w}$ را به دست آورد. روش کار به این ترتیب است. در تکرار t ام $(t = 0, 1, \dots)$ یک مقدار جاری برای بردار وزن وجود دارد که ما آن را $\mathbf{w}(t)$ می‌نامیم. در این وضعیت، الگوریتم مثالی را از مجموعه مثال‌های $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ که به نادرستی طبقه‌بندی شده‌اند را انتخاب می‌کند. فرض کنید این مثال $(\mathbf{x}(t), y(t))$ باشد (چون در هر t یک مثال انتخاب می‌شود). با استفاده از این مثال ما $\mathbf{w}(t)$ را طبق فرمول زیر به‌روزرسانی می‌کنیم. چون مثال به درستی طبقه‌بندی نشده است خواهیم داشت $y(t) \neq \mathrm{sign}(\mathbf{w}^{\mathrm{T}}(t)\mathbf{x}(t))$

$$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t). \qquad (۱-۳)$$

این فرمول خط مرزی جداکننده دو ناحیه را اندکی به سمت مثال بدطبقه‌بندی‌شده $\mathbf{x}(t)$

جابه‌جا می‌کند (همانند شکل زیر). به همین ترتیب الگوریتم تا زمانی که همه مثال‌ها درست طبقه‌بندی شوند، این چرخه را تکرار می‌کند.



هرچند عمل به‌روزرسانی تنها مثال بدطبقه‌بندی شده فعلی را در نظر می‌گیرد و ممکن است به نظر برسد که این عمل، طبقه‌بندی بقیه مثال‌ها را بر هم می‌زند؛ اما می‌توان اثبات کرد که این الگوریتم نهایتاً به یک راه‌حل درست دست می‌یابد؛ البته به شرطی که داده‌ها به شکل خطی تفکیک‌پذیر باشند. این نتیجه ربطی به مقدار اولیه بردار وزن و همین‌طور ترتیب بررسی مثال‌های بدطبقه‌بندی‌شده ندارد و درهرصورت ما به راه‌حل درست می‌رسیم. بردار وزن اولیه می‌تواند به شکل تصادفی مقداردهی شود و یا برابر با صفر انتخاب شود. همین‌طور مثال بدطبقه‌بندی‌شده می‌تواند در هر تکرار به شکل تصادفی انتخاب شود (برای مثال با استفاده از یک حلقه جهت جستجو در کلیه مثال‌ها و انتخاب اولین مثالی که به‌درستی طبقه‌بندی نشده است).

الگوریتم پرسپترون موفق شد با یک چرخه تکراری ساده، در فضای نامتناهی بردار وزن‌ها بردار وزنی را بیابد که داده‌ها را به‌درستی تفکیک می‌کند. این امر نشان می‌دهد که یک الگوریتم یادگیری می‌تواند یک فضای نامتناهی از فرضیات را با گام‌هایی ساده و متناهی جستجو کند. این ویژگی، مشخصه تکنیک‌های بسیاری است که در یادگیری ماشین استفاده می‌شوند. برخی از آنها به‌مراتب از الگوریتم پرسپترون پیچیده‌تر هستند.

الگوریتم یادگیری پرسپترون با یافتن فرضیه‌ای که تمام نقاط مجموعه آموزشی را به‌درستی تفکیک می‌کند توانست به هدف خود برسد. اما آیا این به این معنی است که این فرضیه نقاط خارج از $\mathcal{D}$ را نیز با موفقیت طبقه‌بندی می‌کند؟ این یک سؤال کلیدی است که ما قصد داریم در

این کتاب به‌طور کامل در مورد آن بحث کنیم.

۱-۲-۳ تفاوت یادگیری و طراحی

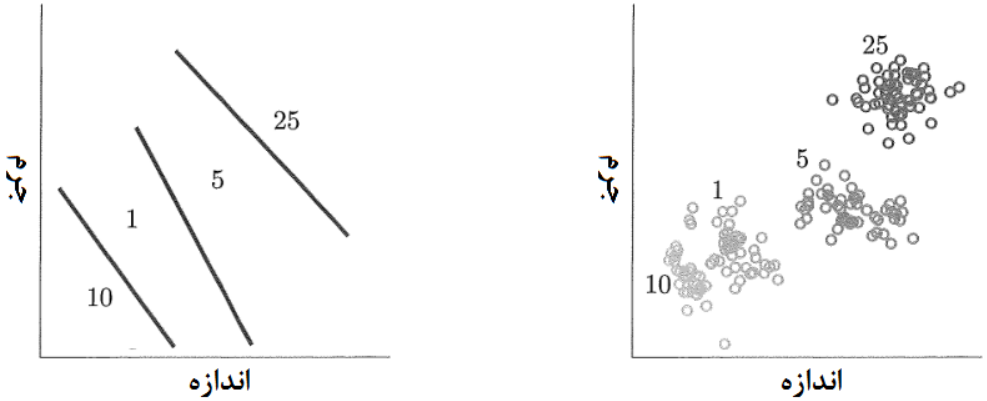
تاکنون گفتیم که یادگیری چیست، در اینجا قصد داریم بگوییم یادگیری چه چیزی نیست. هدف این قسمت این است که بتوانیم میان یادگیری و رویکرد مرتبط دیگری به نام "طراحی" که در مسائل مشابه مطرح می‌شود، تمایز قائل شویم. اصولاً یادگیری بر اساس داده‌ها قرار دارد، درحالی‌که روش طراحی از داده‌ها استفاده نمی‌کند و بر اساس یک سری مشخصه‌ها[1] قرار دارد. معمولاً این رویکرد در حوزه تشخیص الگو درکنار رویکرد یادگیری مطرح می‌شود.

برای مثال مسئله تشخیص نوع سکه واردشده به یک ماشین فروش خودکار را در نظر بگیرید. در این مسئله انتظار داریم که ماشین سکه‌های ۵ تومانی ۱۰ تومانی و ۲۵ تومانی را تشخیص دهد. برای این مسئله هر دو روش یادگیری ماشین و روش طراحی با استفاده از مشخصه‌ها را توضیح می‌دهیم. فرض کنید هر سکه با دو مشخصه اندازه و جرم در قالب یک ورودی دوبعدی مشخص می‌شود. در رویکرد یادگیری، ما نمونه‌هایی از هر نوع سکه را در اختیار داریم که از آنها به‌عنوان مجموعه داده استفاده می‌کنیم. در این حالت جرم و اندازه را به‌عنوان بردار ورودی و نوع سکه را به‌عنوان بردار خروجی در نظر می‌گیریم. شکل ۱-۴ نحوه پراکندگی نقاط مجموعه داده در فضای دوبعدی ورودی را نشان می‌دهد. همان‌طور که مشاهده می‌شود، از هر نوع سکه، سکه‌هایی با تفاوت جزئی در جرم و اندازه وجود دارد که در یک خوشه قرار می‌گیرند (مقادیر اندازه و وزن برای کلیه سکه‌های هم نوع یکسان نیست).

الگوریتم یادگیری به دنبال فرضیه‌ای است که این داده‌ها را به‌خوبی طبقه‌بندی کند. با داشتن چنین فرضیه‌ای، چنانچه ماشین بخواهد سکه جدیدی را تشخیص دهد، ابتدا جرم و ابعاد آن را اندازه می‌گیرد و طبق فرضیه یادگیری‌شده آن را طبقه‌بندی می‌کند (شکل قسمت ب).

در ادامه به توضیح روش "طراحی" می‌پردازیم. در این روش ما مستقیماً به ضرابخانه مراجعه می‌کنیم و در مورد مشخصه‌های سکه‌های مختلف از آنها اطلاعات می‌گیریم. همین‌طور در مورد تعداد سکه‌های ضرب شده از هر نوع سکه سؤال می‌کنیم. با این مقادیر می‌توانیم تخمینی از فراوانی نسبی هر سکه را به دست آوریم. سرانجام همه این اطلاعات را کنار هم می‌گذاریم و یک توزیع احتمال توأم از اندازه، جرم و نوع سکه را محاسبه می‌کنیم. با استفاده

[1] Specifications



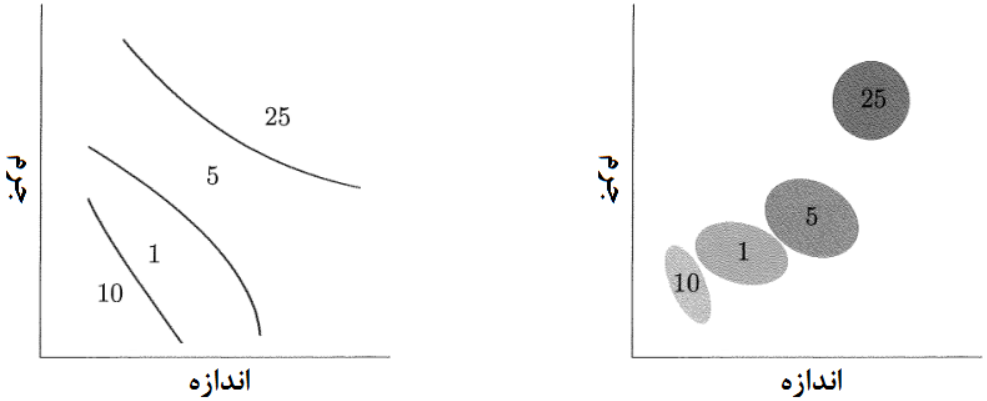
(الف) داده‌های سکه‌ها (ب) طبقه‌بند یادگیری شده

شکل ۱-۴: رویکرد یادگیری به مسئله طبقه‌بندی سکه‌ها الف)داده‌های آموزشی سکه‌های مختلف بر حسب جرم و اندازه که در خوشه‌های جداگانه قرار می‌گیرند ب) قانون یادگیری شده که خوشه‌ها را تفکیک می‌کند. یک سکه جدید بر حسب ناحیه‌ای که در صفحه اندازه و جرم در آن قرار می‌گیرد، طبقه‌بندی می‌شود.

از این توزیع، می‌توانیم قانون بهینه تصمیم‌گیری برای طبقه‌بندی سکه‌ها بر اساس جرم و اندازه را ایجاد کنیم (شکل ۱-۴ (ب)). این قانون برای یک جرم و اندازه داده‌شده، نوع سکه‌ای که دارای بیشترین احتمال انطباق با این مشخصات است را انتخاب می‌کند، درنتیجه به کمترین احتمال خطا در تشخیص یک سکه دست پیدا خواهد کرد. این قانون نظریه تصمیم بهینه بیزین[۱] خوانده می‌شود. برخی از مدل‌های یادگیری بر اساس همین نظریه بنا نهاده شده‌اند و احتمال را از روی داده‌ها تخمین می‌زنند.

تفاوت اصلی میان رویکرد یادگیری و رویکرد طراحی در نقش متفاوتی است که داده‌ها در هر یک بازی می‌کنند. در رویکرد طراحی پارامترهای مسئله مشخص هستند و ما می‌توانیم به شکل تحلیلی f را بدون نیاز به استفاده از هیچ نوع داده‌ای به دست آوریم. در رویکرد یادگیری مشخصات مسئله کمتر واضح هستند و ما برای فهمیدن f نیاز به داده داریم. هر دوی این روش‌ها ممکن است در برخی از کاربردها قابل‌استفاده باشند، اما در کاربردهایی که تابع هدف ناشناخته است تنها رویکرد یادگیری است که قابل‌استفاده است. در اینجا قصد نداریم کاربردها و یا کارایی این دو رویکرد را مقایسه کنیم. تنها خاطرنشان کردیم که رویکرد طراحی از رویکرد یادگیری متفاوت است و این کتاب در مورد یادگیری است.

[۱] Bayesian optimal decision theory



(ب) طبقه‌بند نتیجه‌شده (الف) مدل احتمالاتی داده‌ها

شکل ۱-۵: رویکرد طراحی به مسئله طبقه‌بندی سکه‌ها الف) یک مدل احتمالاتی برای جرم، اندازه و نوع سکه که از مشخصات داده شده استخراج می‌شود. نواحی با احتمال بالا برای هر نوع سکه مشخص شده است. ب) قانون طبقه‌بندی که به روش تحلیلی برای کمینه کردن احتمال خطای تشخیص یک سکه براساس جرم و اندازه استخراج شده است. نواحی استخراج شده برای هر سکه نمایش داده شده‌اند.

۱-۳ انواع یادگیری

مبنای اساسی یادگیری ماشین، استفاده از مجموعه‌ای از مشاهدات برای آشکار ساختن فرایند زیرین است. این یک قضیه خیلی کلی است و در یک چارچوب مشخص جای نمی‌گیرد. درنتیجه گونه‌های متفاوت یادگیری برای موقعیت‌ها و فرضیات متفاوت ایجاد شده‌اند. در این قسمت به معرفی برخی از این گونه‌ها می‌پردازیم.

گونه‌ای که تاکنون بحث شد موسوم به "یادگیری با ناظر" [۱] است. این روش یکی از پرکاربردترین روش‌های یادگیری است و مطالعات زیادی نیز بر روی آن انجام شده است.

برخی نسخه‌های یادگیری با ناظر خیلی ساده هستند و می‌توان آنها را در یک چارچوب مشترک قرار داد، اما برخی دیگر پیچیده‌تر هستند و نیاز به مفاهیم و تکنیک‌های مستقلی دارند. نسخه‌های اصلی بر اساس مشخصات ذاتی هر مجموعه داده تقسیم‌بندی شده‌اند.

۱-۳-۱ یادگیری با ناظر

زمانی که در مثال‌های مجموعه داده آموزشی در کنار هر ورودی، خروجی صحیح به شکل صریح داده‌شده باشد، ما در قلمرو یادگیری با ناظر هستیم و تا اینجا در مورد آن بحث کردیم.

[۱] supervised learning

برای مثال مسئله تشخیص ارقام دست‌نویس را در نظر بگیرید (ارقام ۰ تا ۹). یک مجموعه داده مناسب برای این کار، مجموعه‌ای از تصاویر ارقام دست‌نویس است به‌شکلی‌که رقم موجود در هر تصویر صریحاً مشخص شده باشد. درآن‌صورت ما مجموعه‌ای از مثال‌ها به شکل (تصویر، رقم) خواهیم داشت. این روش "با ناظر" خوانده می‌شود، زیرا یک ناظر انسانی به خود زحمت داده است و به هر تصویر ورودی نگاه کرده و رقم خروجی را در کنار آن مشخص کرده است.

زمانی که ما از نسخه‌های یادگیری باناظر صحبت می‌کنیم، منظورمان روش‌های مختلفی است که این مجموعه داده می‌تواند به فرایند یادگیری تزریق شود. معمولاً این مجموعه قبل از فرایند یادگیری به طور کامل ایجاد شده و در اختیار الگوریتم یادگیری قرار می‌گیرد. برای مثال در مثال کارت اعتباری سوابق مشتریان از قبل موجود است و یا در مسئله توصیه فیلم، امتیازات داده‌شده توسط مشتریان به فیلم‌ها موجود هستند. در حوزه عملی، وجود داده‌های "آماده" یک قاعده بسیار متداول است و ما نیز در این کتاب بر روی این نسخه از یادگیری تمرکز خواهیم کرد. هرچند دو نسخه مهم دیگر نیز وجود دارند که در ادامه به آنها اشاره می‌کنیم.

یکی از این نسخه‌ها "یادگیری فعال"[۱] نامیده می‌شود که در آن مثال‌های آموزشی با توجه به درخواست‌هایی که مادر حین یادگیری صورت می‌دهیم فراهم می‌شوند. در این حالت ما یک ورودی x را انتخاب می‌کنیم و ناظر خروجی صحیح برای این ورودی را مشخص می‌کند. این امر به ما اجازه می‌دهد تا ورودی x را به شکل راهبردی انتخاب کنیم. به این صورت که ما سعی می‌کنیم با هر انتخاب، مقدار اطلاعات کسب‌شده را بیشینه کنیم. برای درک این موضوع، مسابقه بیست‌سؤالی را در نظر بگیرید که ما در آن سعی می‌کنیم با استفاده از یک سری سؤالات راهبردی به جواب نهایی برسیم.

نسخه مهم دیگر یادگیری، "یادگیری برخط"[۲] نامیده می‌شود. در این حالت مجموعه داده آموزشی به صورت "یک مثال در یک زمان داده" داده می‌شود. این امر زمانی اتفاق می‌افتد که ما جریانی از داده‌ها (مثل جریان داده‌های صوتی یا تصویری) داریم که باید به یک الگوریتم در حال اجرا تزریق شوند. برای مثال فرض کنید پس از اینکه سامانه توصیه فیلم استقرار یافت، این سامانه بتواند امتیازات جدیدی که کاربران به فیلم‌ها می‌دهند را نیز مورد پردازش قرار دهد. یادگیری برخط زمانی که ما برای پردازش یک‌باره داده‌ها با محدودیت‌های محاسباتی و ذخیره‌سازی روبرو هستیم می‌تواند مفید واقع شود. البته این نوع یادگیری مخصوص یادگیری

[۱] Active Learning
[۲] Online Learning

باناظر نیست و در دیگر گونه‌های یادگیری نیز استفاده می‌شود.

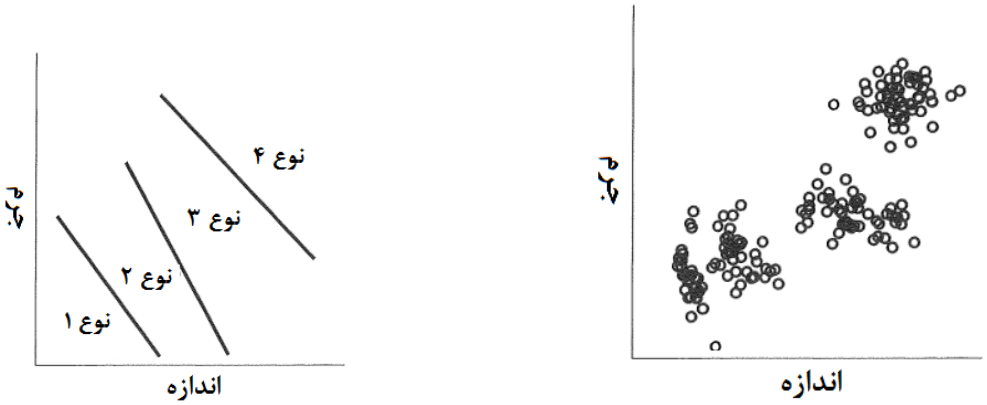
۱-۳-۲ یادگیری تقویتی

چنانچه داده‌های آموزشی به‌طور صریح خروجی درست را برای هر ورودی مشخص نکنند، دیگر در شرایط یادگیری باناظر قرار نداریم. برای مثال نوزادی را تصور کنید که به‌تدریج یاد می‌گیرد به یک فنجان چای داغ دست نزند. تجربه این نوزاد شامل موقعیت‌هایی است که نوزاد با یک فنجان چای داغ روبرو می‌شود و با این تصمیم مواجه است که آیا به فنجان دست بزند یا خیر. هر زمان که نوزاد به فنجان دست می‌زند، نتیجه آن یک درد زیاد است و هر زمان که دست نمی‌زند، نتیجه درد کمتری است (به علت عدم ارضاء حس کنجکاوی). نهایتاً نوزاد به این نتیجه می‌رسد که بهتر است به فنجان دست نزند. این مثال‌های آموزشی به نوزاد نمی‌گویند که در هر زمان چه کاری را باید انجام دهد (خروجی درست چیست)، بلکه به کارهایی که نوزاد انجام می‌دهد امتیاز می‌دهند و نوزاد از این مثال‌ها برای تقویت کارهای بهتر استفاده می‌کند تا نهایتاً متوجه شود در موقعیت‌های مشابه باید به چه شکل عمل کند. این موضوع وجه اساسی یادگیری تقویتی است که در آن داده‌های آموزشی حاوی خروجی هدف نیستند، بلکه شامل برخی خروجی‌های ممکن و درجه‌ای از میزان مطلوبیت یک خروجی هستند. در مقایسه با یادگیری باناظر که مثال‌های آموزشی به شکل (ورودی، خروجی صحیح) هستند، در یادگیری تقویتی قالب کلی مثال‌ها به شکل زیر است:

(ورودی، تعدادی خروجی، درجه‌ای برای این خروجی)

دقت کنید که یک مثال بیان نمی‌کند که بقیه خروجی‌ها برای این ورودی ویژه چقدر مطلوب خواهند بود و تنها میزان مطلوب بودن یک خروجی را مشخص می‌کنند.

یادگیری تقویتی به‌طور ویژه برای یادگیری نحوه انجام یک بازی روش مناسبی است. برای مثال در بازی تخته‌نرد، تصور کنید که شما در موقعیتی هستید که قدرت انتخاب میان چندین عمل مختلف را دارید و قصد دارید بهترین عمل ممکن را انتخاب کنید. این که ما در هر موقعیت از بازی بدانیم بهترین عمل ممکن چیست، کار ساده‌ای نیست. به همین دلیل به‌آسانی نمی‌توان مجموعه مثال‌های آموزشی مورد نیاز یادگیری باناظر را فراهم نمود. اما اگر در عوض از یادگیری تقویتی استفاده کنیم، تنها کاری که باید بکنیم این است که در آن موقعیت یک سری حرکات انجام دهیم و درجه مطلوب بودن هر یک را گزارش کنیم. به این طریق شما یک سری مثال‌های آموزشی خواهید ساخت. وظیفه الگوریتم یادگیری تقویتی این است که اطلاعاتی که



(ب) یادگیری بدون ناظر

(الف) داده‌های بدون برچسب سکه‌ها

شکل ۱-۶: یادگیری بدون‌ناظر برای طبقه‌بندی سکه‌ها الف) داده‌های شکل ۱-۴ (الف) اما این بار بدون برچسب. این داده‌ها همچنان در چند خوشه قرار می‌گیرند. ب)یادگیری بدون ناظر با خوشه‌ها به عنوان گونه‌های مختلف رفتار می‌کند. این قوانین ممکن است گاهی اوقات مبهم باشند، برای مثال نوع ۱ و نوع ۲ ممکن است یک نوع باشند.

از مثال‌های مختلف می‌آید را مرتب کند تا تشخصی دهد بهترین مسیر بازی چیست.

۱-۳-۳ یادگیری بدون ناظر

در یادگیری بدون ناظر، داده‌های آموزشی هیچ‌گونه اطلاعی در مورد خروجی ندارند و ما تنها مثال‌هایی به شکل $\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_N$ در اختیار داریم. شاید تعجب کنید که چگونه می‌توان با نگاه کردن به یک سری داده چیزی یاد گرفت. برای مثال، مسئله تشخیص نوع سکه با توجه به اندازه و جرم سکه را در نظر بگیرید. فرض کنید در داده‌های آموزشی ما هیچ اطلاعی در مورد نوع هر سکه نداریم. این داده‌های بدون برچسب در شکل ۱-۶ نمایش داده‌شده‌اند. همان‌طور که مشاهده می‌کنید ما همچنان خوشه‌های از داده‌های مشابه را داریم که هیچ برچسبی ندارند. مرزهای تصمیم‌گیری در یادگیری بدون ناظر ممکن است شبیه یادگیری باناظر باشند، با این تفاوت که طبقه‌ها بدون برچسب هستند. هرچند در اینجا خوشه‌بندی صحیح وضوح کمتری دارد و حتی ممکن است تعداد خوشه‌ها مبهم باشد.

بااین‌حال این مثال نشان می‌دهد که ما همچنان می‌توانیم از خود داده چیزی یاد بگیریم. یادگیری بدون ناظر را می‌توان به شکل یافتن یک سری الگو و یا ساختار در داده‌های ورودی ملاحظه کرد. برای مثال اگر هدف طبقه‌بندی مجموعه‌ای از کتاب‌ها در موضوعات مختلف باشد و تنها مشخصات عمومی کتاب‌ها در دسترس باشد، می‌توانیم کتاب‌هایی که دارای مشخصات

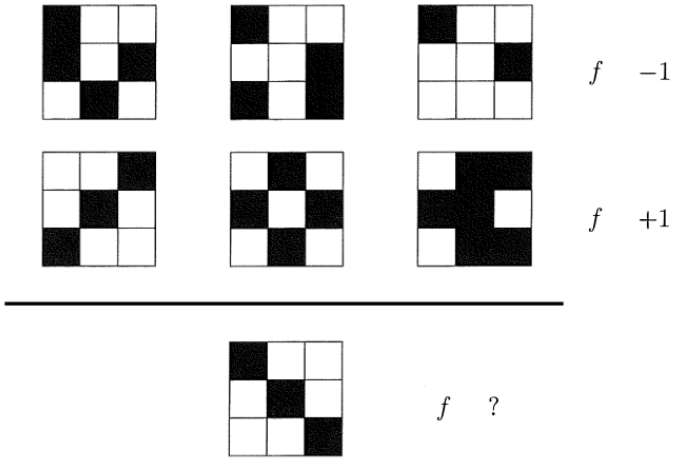
مشابهی هستند را پیدا کرده و در یک دسته قرار دهیم، بدون آنکه بخواهیم برچسبی برای هر دسته بگذاریم.

یادگیری بدون ناظر را همین‌طور می‌توان به‌عنوان روشی برای نمایش داده در سطح بالاتر ملاحظه کرد. برای مثال، تصور کنید که شما از زبان اسپانیایی چیزی نمی‌دانید و شرکت شما قصد دارد از ماه آینده محل کارتان را به اسپانیا منتقل کند. پس از استقرار در اسپانیا قرار است یک سری کلاس‌های زبان اسپانیایی برای شما برگزار شود، اما شما قصد دارید قبل از رفتن به آنجا خودتان را آماده کنید. تنها چیزی که در اختیار دارید یک ایستگاه رادیویی اسپانیایی‌زبان است. شما به مدت یک ماه خودتان را در معرض جملات این زبان قرار می‌دهید. این عمل درواقع یک تجربه یادگیری بدون ناظر است، زیرا شما معنی کلماتی که می‌شنوید را نمی‌دانید. هرچند به‌تدریج نمایش بهتری از زبان در ذهنتان شکل می‌گیرد و با ساختار کلی جملات و نحوه ادای کلمات وفق پیدا می‌کنید. درنتیجه زمانی که به اسپانیا می‌رسید در شرایط بهتری برای یادگیری زبان قرار دارید. درواقع می‌توان از یادگیری بدون ناظر به شکل یک مرحله مقدماتی برای یادگیری باناظر استفاده کرد. هرچند در جاهایی نیز می‌توان از آن به عنوان یک روش مستقل استفاده کرد.

۱-۳-۴ اشکال دیگر یادگیری

همان‌طور که در ابتدای این فصل گفته شد، مبحث یادگیری در حوزه‌های مختلف به شکل مستقل تکامل یافته است و هر کدام ممکن است اصطلاحات متفاوتی را برای مفاهیم مشابه به کار ببرند و یا تأکیدات متفاوتی روی این مفاهیم داشته باشند. به همین دلیل این مبحث ممکن است شامل موضوعاتی باشد که در ادبیات مرتبط، با نام‌های متفاوتی از آنها یاد می‌شود. یکی از این نام‌ها "یادگیری ماشین" است که به طور کلی سعی می‌کند مبحث یادگیری ماشین را از یادگیری انسان متمایز کند. در این کتاب ما بیشتر بر روی کاربرد عمومی این حوزه تمرکز داریم و چارچوب نظری و ملاحظات عملی آن را شرح می‌دهیم. در ادامه به‌طور خلاصه به دو حوزه مشابه دیگر نیز اشاره می‌کنیم.

یکی از این حوزه‌ها آمار است. همانند مبحث یادگیری، در این علم نیز موضوع اصلی استفاده از یک سری مشاهدات به‌منظور آشکار کردن یک فرایند زیرین است. در این مورد، فرایند زیرین یک توزیع احتمال است که ما از آن اطلاعی نداریم و مشاهدات نمونه‌هایی از این توزیع هستند. ازآنجاکه آمار شاخه‌ای از ریاضیات است، تأکید بیشتر روی مسائلی است که



شکل ۱-۷: یک مسئله یادگیری تصویری. دو سطر اول نمایشگر مثالهای آموزشی هستند. برای داده های سطر اول مقدار f برابر با $+1$ و برای سطر دوم برابر با -1 است. وظیفه شما این است که با توجه به این مجموعه داده fرا یادبگیرید و در مورد نمونه آزمایشی (سطر ۳) مقدار fرا بدست آورید. آیا $+1$ یا -1 را بدست می آورید؟

باید برای آنها یک اثبات قوی ارائه شود. به عبارت دیگر آمار روی مدل‌های ایدئال تمرکز دارد و آنها را با جزئیات زیاد بررسی می‌کند. این مطلب درواقع تفاوت اصلی میان روش‌های آماری با روش‌هایی است که ما در یادگیری از آنها استفاده می‌کنیم. در یادگیری ما محدودیت‌های کمتری داریم و نسبت به آمار با مدل‌های عمومی‌تری مواجه هستیم. درنتیجه به همان نسبت ممکن است به نتایج ضعیف‌تری برسیم، اما این نتایج قابلیت تعمیم بیشتری دارند.

داده‌کاوی شاخه کاربردی دیگری است که تمرکزش بر روی یافتن الگوها، روابط و ناهنجاری‌ها در یک پایگاه داده رابطه‌ای حجیم است. برای مثال می‌توانیم با استفاده از سوابق پزشکی تعداد زیادی بیمار، یک رابطه علت-معلولی میان یک داروی ویژه و اثرات درازمدت آن پیدا کنیم. از لحاظ تکنیکی، داده‌کاوی همان یادگیری ماشین است اما به جای پیش‌بینی، بر روی تحلیل داده‌ها تأکید دارد. ازآنجاکه پایگاه داده‌های مورد استفاده معمولاً بسیار حجیم هستند، در این روش ما با پیامدهای محاسباتی روبرو هستیم.

۱-۴ آیا یادگیری امکان‌پذیر است

تابع هدف f موضوع یا هدف یادگیری است. مهم‌ترین چیزی که می‌توان در مورد آن ادعا کرد این است که تابعی ناشناخته است. این مطلب سؤالی طبیعی را به دنبال دارد. چگونه

یک مجموعه داده محدود می‌تواند اطلاعات لازم برای فهم کامل تابع هدف را فراهم کند؟ شکل ۱-۷ این مشکل را به تصویر می‌کشد. یک مسئله یادگیری ساده همراه با شش مثال آموزشی از یک تابع هدف دودویی با دو مقدار $+1$ و -1 نمایش داده‌شده است. سعی کنید با توجه به این مثال‌ها تابعی که روی آنها اعمال می‌شود را پیدا کنید (یاد بگیرید) و آن را روی تک داده آزمایشی اعمال کنید. روی این داده به چه مقداری می‌رسید؟ اکنون این مسئله را به دوستان خود نشان دهید و ببینید که آیا آنها نیز به همان نتیجه شما می‌رسند. به‌احتمال‌زیاد جواب واحدی دریافت نخواهید کرد و البته دلایل خوبی نیز برای این موضوع وجود دارد. درحقیقت بیش از یک تابع روی این شش مثال آموزشی قابل تطبیق است و برخی از این توابع مقدار -1 و برخی دیگر مقدار $+1$ را روی داده آزمایشی برمی‌گردانند. برای مثال اگر تابع هدف f زمانی که الگو متقارن است دارای مقدار $+1$ باشد، مقدار آن برای داده آزمایشی $+1$ است. اما اگر زمانی که خانه بالای سمت چپ سفید است تابع هدف دارای مقدار -1 باشد، این مقدار برای داده آزمایشی -1 خواهد بود. هر دو این توابع با تمام مثال‌های مجموعه داده نشان داده‌شده مطابقت دارند، درنتیجه برای اینکه بدانیم کدام یک تابع واقعی است، اطلاعات کافی وجود ندارد. به نظر می‌رسد این مسئله با امکان‌پذیری یادگیری همخوانی ندارد. برای اینکه وضعیت را بدتر کنیم، باید بگوییم که مشکلی که در این مثال مشاهده شد تنها یک استثنا نیست، بلکه یک قاعده عمومی است.

۱-۴-۱ خارج از مجموعه آموزشی

پس از اینکه داده‌های آموزشی را دریافت کردیم (برای مثال دو سطر اول شکل ۱-۷)، به مقدار f روی کلیه نقاط این مجموعه دسترسی داریم. اما این به این معنی نیست که ما f را یاد گرفته‌ایم، زیرا دانستن این مقادیر چیزی در مورد f خارج از مجموعه $\mathcal{D}$ را تضمین نمی‌کنند. ما تنها چیزی را می‌دانیم که دیده‌ایم، اما این یادگیری نیست، این "به خاطر سپردن" است. آیا $\mathcal{D}$ چیزی در مورد خارج از خود به ما می‌گوید که ما قبلاً نمی‌دانستیم؟ اگر جواب بله باشد، پس چیزی یاد گرفته‌ایم. اگر جواب خیر باشد، می‌توانیم نتیجه بگیریم که یادگیری ممکن نیست.

ازآنجاکه ما همواره f را تابعی ناشناخته می‌دانیم، می‌توانیم اثبات کنیم که این تابع خارج از $\mathcal{D}$ نیز همچنان ناشناخته باقی خواهد ماند. به جای ارائه یک اثبات رسمی و عمومی، ایده اصلی را با استفاده از یک مثال عینی نشان می‌دهیم.

برای مثال یک تابع بولی را روی یک فضای ورودی سه‌بعدی $\mathcal{X} = \{0,1\}^{+3}$ در نظر بگیرید.

ما مجموعه‌ای از ۵ مثال داریم که در جدول زیر مشخص شده‌اند. جهت سادگی نمایش، خروجی هر مثال را با یک نقطه سیاه یا سفید نشان داده‌ایم. مزیت این تابع ساده این است که به تمام فضای ورودی که تنها شامل ۸ مقدار مختلف است دسترسی داریم ($2^3 = 8$) و درنتیجه تمام توابع هدفی که روی این دامنه ورودی قابل‌اعمال است را می‌توانیم بررسی کنیم ($2^8 = 256$ تابع مختلف وجود دارد).

x_n			y_n
0	0	0	○
0	0	1	●
0	1	0	●
0	1	1	○
1	0	0	●

اکنون اجازه دهید به مشکل یادگیری f بپردازیم. ازآنجاکه تابع f به جز در داخل مجموعه $\mathcal{D}$ در بقیه نقاط ناشناخته است، هر تابعی که با مقادیر این تابع در مجموعه $\mathcal{D}$ مطابقت داشته باشد، شانس این را دارد که تابع f حقیقی باشد. جدول زیر چنین توابعی را نمایش می‌دهد. ازآنجاکه سه نقطه خارج از مجموعه $\mathcal{D}$ قرار دارند، ۸ تابع متفاوت قابل‌تصور است ($(f_1 \dots f_8)$).

x_n			y_n	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	0	○	○	○	○	○	○	○	○	○	○
0	0	1	●	●	●	●	●	●	●	●	●	●
0	1	0	●	●	●	●	●	●	●	●	●	●
0	1	1	○	○	○	○	○	○	○	○	○	○
1	0	0	●	●	●	●	●	●	●	●	●	●
1	0	1		?	○	○	○	○	●	●	●	●
1	1	0		?	○	○	●	●	○	○	●	●
1	1	1		?	○	●	○	●	○	●	○	●

در این جدول، نقاط مجموعه $\mathcal{D}$ توسط یک خط از بقیه نقاط دامنه ورودی جدا شده‌اند. همین‌طور فرضیه g که قرار است f را تقریب بزند، نشان داده‌شده است. این فرضیه بر اساس ۵ مثال موجود انتخاب می‌شود. جدول موردی از g را نشان می‌دهد که بر اساس انطباق آن با f روی مجموعه $\mathcal{D}$ انتخاب شده است.

اگر ما همچنان به این موضوع که تابع هدف f ناشناخته است پایبند باشیم، نمی‌توانیم هیچ‌یک از توابع f_1 تا f_8 را از f حقیقی بودن مستثنا کنیم. اکنون تناقضی پیش می‌آید. تمام هدف یادگیری این است که ما بتوانیم مقدار f را روی نقاطی که دیده نشده‌اند پیش‌بینی کنیم و کیفیت یادگیری به این وابسته است که چقدر پیش‌بینی ما درست باشد. اما فارغ از اینکه g در سه نقطه دیده نشده چه مقداری را پیش‌بینی کند (با علامت ؟ مشخص شده‌اند)، بسته به اینکه

کدام‌یک از توابع f_1 تا f_8 تابع هدف باشد، این پیش‌بینی می‌تواند با این تابع مطابقت داشته باشد یا نداشته باشد. درواقع هر سه مقداری که جایگزین علامت ؟ شوند، به‌خوبی مقادیر دیگر هستند و همه شانس یکسانی در این رابطه دارند.

مهم نیست که الگوریتم یادگیری به چه شکل عمل می‌کند و از چه مجموعه فرضیات $\mathcal{H}$ ای استفاده می‌کند. همین‌طور اینکه آیا $\mathcal{H}$ حاوی فرضیه‌ای است که کاملاً با $\mathcal{D}$ مطابقت دارد و آیا الگوریتم یادگیری این فرضیه را انتخاب می‌کند و یا فرضیه دیگری را انتخاب می‌کند که با $\mathcal{D}$ مطابقت ندارد، تا زمانی که کارایی خارج از مجموعه $\mathcal{D}$ موردنظر باشد، هیچ‌یک از این مسائل اهمیتی ندارند.

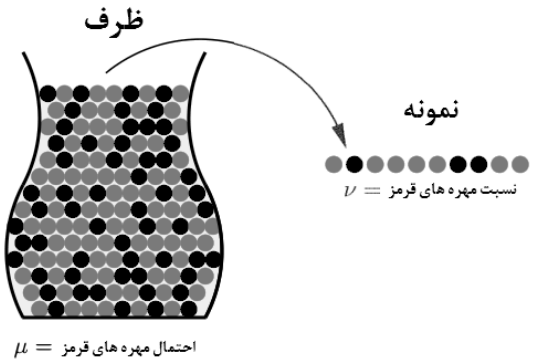
این تناقض تنها معطوف به توابع بولی نیست، بلکه به مسئله یادگیری در حالت کلی بسط پیدا می‌کند. تا زمانی که f ناشناخته باشد، آگاهی از مقادیر این تابع روی مجموعه $\mathcal{D}$ نمی‌تواند هیچ الگویی از مقادیر f خارج از $\mathcal{D}$ را کنار بگذارد. درنتیجه پیش‌بینی g خارج از $\mathcal{D}$ بی‌معنی است. آیا این موضوع به این معنی است که واقعاً یادگیری امکان‌پذیر نیست؟ اگر چنین بود، این کتاب جای بحث دیگری نداشت. خوشبختانه یادگیری قابل حصول است و ما در ادامه دلیل آن را توضیح می‌دهیم.

۱-۴-۲ تحلیل امکان‌پذیری یادگیری با استفاده از احتمال

در این قسمت نشان می‌دهیم که می‌توان تنها با استفاده از مجموعه $\mathcal{D}$، چیزی خارج از آن را استنتاج کرد البته به شکل احتمالاتی. ممکن است چیزی که نهایتاً به دست می‌آوریم، یک تابع هدف کامل نباشد، اما به‌هرحال این امر نشان می‌دهد که خارج از مجموعه $\mathcal{D}$ قابل‌دستیابی است. زمانی که ما این مطلب را به‌طور کامل تثبیت کردیم، از آن در مسئله عمومی یادگیری استفاده خواهیم کرد و نشان خواهیم دهیم که چه چیزی قابل یادگیری است و چه چیزی قابل یادگیری نیست.

اجازه دهید که مسئله ساده‌تر انتخاب یک نمونه[۱] را در نظر بگیریم و ببینیم چه زمانی می‌توان در مورد اشیاء خارج از نمونه چیزی استنتاج نمود. برای مثال ظرفی را در نظر بگیرید که حاوی بی‌شمار مهره‌های سبز و قرمز است. اگر مهره‌ای را به شکل تصادفی از ظرف خارج کنیم، احتمال اینکه مهره قرمز باشد را μ و احتمال اینکه سبز باشد را $1-\mu$ در نظر بگیرید. فرض می‌کنیم که مقدار μ برای ما ناشناخته است. ما نمونه‌ای از N مهره مستقل را با جایگذاری

[۱] Sample



شکل ۱-۸: انتخاب یک نمونه تصادفی از ظرفی حاوی مهره‌های قرمز و سبز.

برمی‌داریم و نسبت مهره‌های قرمز در این نمونه را یادداشت می‌کنیم. این نسبت را ν می‌نامیم. آیا مقدار ν چیزی در مورد μ به ما می‌گوید؟ یک پاسخ این است که صرف‌نظر از اینکه مهره‌های نمونه چه رنگی باشند، ما چیزی در مورد رنگ مهره‌هایی که برنداشته‌ایم نمی‌دانیم. ممکن است اکثر مهره‌هایی نمونه سبز باشند، درصورتی‌که اکثر مهره‌های درون ظرف قرمز باشند. هرچند این امر ممکن است، اما محتمل نیست.

این وضعیت مانند زمانی است که ما یک نظرسنجی انجام می‌دهیم. معمولاً نظرات افراد در یک نمونه تصادفی به نظر عمومی کل جمعیت گرایش دارد. توزیع احتمال متغیر تصادفی ν برحسب پارامتر μ به‌خوبی شناخته‌شده است و زمانی که اندازه نمونه به اندازه کافی بزرگ شود، ν به سمت μ میل می‌کند. برای اینکه این رابطه را کمّی کنیم ما از یک کران ساده که به نامعادله هافدینگ [۱] معروف است، استفاده می‌کنیم. این نامعادله بیان می‌کند که برای یک نمونه تصادفی به اندازه N:

$$\mathbb{P}[\mid \nu - \mu \mid > \epsilon] \leq 2e^{-2\epsilon^2 N} \qquad (۴-۱)$$

در این فرمول $\mathbb{P}[.]$ احتمال یک پیشامد را مشخص می‌کند. در این مثال مقدار این احتمال با توجه به نمونه تصادفی که انتخاب می‌شود به دست می‌آید (ν متغیر تصادفی است). ϵ می‌تواند هر مقدار مثبت داده‌شده‌ای باشد. به زبان ساده این نامعادله بیان می‌کند که با افزایش اندازه

[۱] Hoeffding inequality

نمونه N، احتمال اینکه ν بیشتر از میزان تحمل[۱] ϵ از μ منحرف شود، بسیار کاهش می‌یابد. تنها پارامتری که در این نامعادله تصادفی است ν است که به نمونه تصادفی وابسته است. دقت کنید μ مقداری تصادفی نیست، بلکه یک مقدار ثابت است، هرچند این مقدار برای ما ناشناخته است.

یک نکته ظریف در اینجا وجود دارد. کاربرد این رابطه تخمین مقدار μ با استفاده از ν است، اما در واقع این μ است که روی ν اثر می‌گذارد و نه برعکس. ازآنجاکه ν به سمت μ میل می‌کند ما ممکن است نتیجه بگیریم که μ به ν نزدیک است، اما توجه داشته باشید که μ یک مقدار ثابت ناشناخته است و در حقیقت ν به μ نزدیک می‌شود.

هرچند احتمال $\mathbb{P}[\mid \mu - \nu \mid > \epsilon]$ به علت وجود μ در عبارت به μ وابسته است، و مقدار μ روی توزیع ν اثر می‌گذارد، بااین‌حال می‌توانیم این احتمال را با کران $2e^{-2\epsilon^2 N}$ محدود کنیم که به μ وابستگی ندارد. همین‌طور توجه کنید که تنها اندازه نمونه یا N روی این کران تأثیر می‌گذارد و اندازه ظرف تأثیری بر آن ندارد. ظرف می‌تواند کوچک یا بزرگ، متناهی یا نامتناهی باشد، اما تا زمانی که از نمونه‌ای با همان اندازه استفاده کنیم، همچنان همان کران را خواهیم داشت.

اگر برای اینکه ν تقریب خوبی از μ باشد، ما ϵ را خیلی کوچک انتخاب کنیم، در آن صورت نیاز به یک نمونه بزرگ (N بزرگ) خواهیم داشت تا سمت راست معادله به اندازه کافی کوچک شود (احتمال انحراف از بازه تحمل کاهش یابد). در آن صورت می‌توانیم ادعا کنیم که ν تقریب خوبی از μ است. هرچند این ادعا به ما مقدار دقیق μ را نمی‌دهد و حتی تضمین نمی‌کند که این تقریب صد در صد برقرار است، بااین‌حال همین که می‌دانیم ما در اغلب اوقات در یک بازه $\pm\epsilon$ از μ قرار داریم، نسبت به زمانی که چیزی نمی‌دانستیم پیشرفت بزرگی محسوب می‌شود.

این موضوع که ما نمونه را به شکل تصادفی انتخاب کردیم به ما اجازه می‌دهد ادعا کنیم که ν به μ نزدیک است. اما چنانچه نمونه به شکل تصادفی انتخاب نمی‌شد و مهره‌ها در یک روال مشخص خارج می‌شدند، ما مزیت استفاده از یک تحلیل احتمالاتی را از دست می‌دادیم و همچنان نسبت به خارج از نمونه در تاریکی می‌ماندیم.

اما مثال ظرف چگونه به مسئله یادگیری مرتبط می‌شود؟ به نظر می‌رسد در مثال ظرف تنها چیزی که نامشخص بود μ بود، اما در مسئله یادگیری کل تابع f ناشناخته است. با این حال

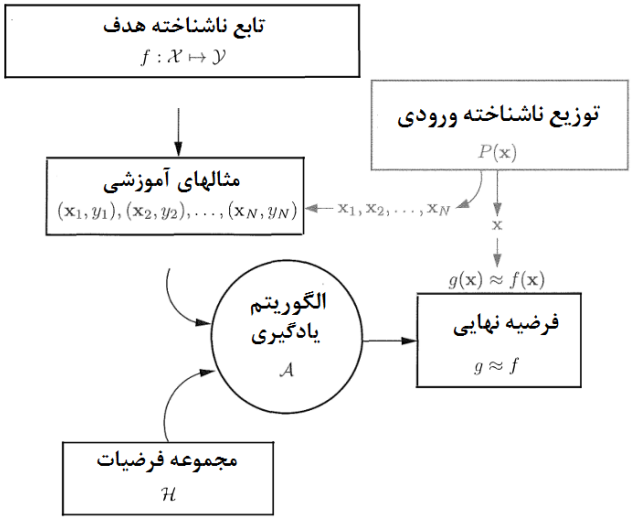
[۱] Tolerance

می‌توان این دو وضعیت را به یکدیگر متناظر کرد. یک فرضیه $h \in \mathcal{H}$ را در نظر بگیرید و آن را در هر نقطه $\mathsf{x} \in \mathcal{X}$ با f مقایسه کنید. اگر $h(\mathsf{x}) = f(\mathsf{x})$ بود، x را سبز و اگر $h(\mathsf{x}) \neq f(\mathsf{x})$ بود، x را قرمز در نظر بگیرید. ازآنجاکه f ناشناخته است، رنگی که هر نقطه به خود می‌گیرد نیز ناشناخته است. هرچند اگر x را به شکل تصادفی مطابق با یک توزیع احتمال P روی فضای ورودی $\mathcal{X}$ انتخاب کنیم، در آن صورت می‌دانیم که x با احتمالی که آن را μ می‌نامیم قرمز خواهد بود (یعنی با f مطابقت ندارد) و با احتمال $1-\mu$ سبز خواهد بود (یعنی با f مطابقت دارد). فارغ از اینکه مقدار μ چه باشد، فضای $\mathcal{X}$ همانند ظرف در شکل ۱-۸ عمل خواهد کرد.

مجموعه مثال‌های آموزشی نقش نمونه‌ای که ما از ظرف بیرون می‌کشیم را بازی می‌کنند. اگر ورودی‌های $\mathsf{x}_1, \mathsf{x}_2 \ldots, \mathsf{x}_N$ در $\mathcal{D}$ به شکل مستقل مطابق با توزیع P انتخاب شوند، در آن صورت ما یک نمونه تصادفی از نقاط قرمز ($h(\mathsf{x}) \neq f(\mathsf{x})$) و سبز ($h(\mathsf{x}) = f(\mathsf{x})$) خواهیم داشت. هر نقطه با احتمال μ قرمز و با احتمال $1-\mu$ سبز خواهد بود. در این مجموعه، رنگ هر نقطه برای ما مشخص است زیرا هر دو مقدار $h(\mathsf{x}_n)$ و $f(\mathsf{x}_n)$ برای کلیه N نقطه داده شده‌اند. مسئله یادگیری اکنون به مسئله ظرف تقلیل می‌یابد، با این شرط که نقاط مجموعه $\mathcal{D}$ به شکل مستقل با یک توزیع P روی $\mathcal{X}$ انتخاب شوند. هر توزیع P به یک μ در ظرف معادل ترجمه می‌شود. ازآنجاکه μ اجازه داشت ناشناخته باشد، P هم اجازه دارد ناشناخته باشد. شکل ۱-۹ این جزء احتمالاتی را به چارچوب پایه‌ای که در شکل ۱-۲ برای مسئله یادگیری ارائه شد اضافه می‌کند.

با توجه به تناظر یادشده، نامعادله هافدینگ می‌تواند به مسئله یادگیری نیز اعمال شود. این امر به ما امکان می‌دهد تا بتوانیم در مورد خارج از $\mathcal{D}$ پیش‌بینی‌ای داشته باشیم. استفاده از ν برای پیش‌بینی μ، چیزی در مورد f به ما می‌گوید اما دقیقاً مشخص نمی‌کند که f چیست. آنچه ν به ما می‌گوید نرخ خطایی است که h در تقریب f ایجاد می‌کند. اگر ν به صفر نزدیک باشد، می‌توانیم پیش‌بینی کنیم که h به‌خوبی f را روی فضای ورودی تقریب خواهد زد، در غیر این صورت شانسی نخواهیم داشت.

متأسفانه در وضعیت فعلی، ما هیچ کنترلی را روی ν نداریم، زیرا ν بر پایه فرضیه منفرد h بنا نهاده شده است. در یادگیری واقعی ما مجموعه فرضیات $\mathcal{H}$ را برای یافتن h ای که دارای کمترین نرخ خطا است جستجو می‌کنیم. اما اگر تنها یک فرضیه در دست داشته باشیم، در آن صورت در حال یادگیری نخواهیم بود، بلکه در حال بررسی این هستیم که به چه میزان آن



شکل ۱-۹: افزودن احتمال به چارچوب پایه مسئله یادگیری

فرضیه خاص خوب یا بد است. اجازه دهید تناظر ظرف را به وضعیتی تعمیم دهیم که چندین فرضیه در اختیار داریم، تا بتوانیم یادگیری واقعی را بهتر نشان دهیم.

به این منظور ابتدا اسامی توصیفی بیشتری را برای اجزایی که استفاده خواهیم کرد ارائه می‌کنیم. نرخ خطا در نمونه که متناظر با ν در مدل ظرف است و خطای درون-نمونه [۱] خوانده می‌شود عبارت است از:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$$

که در آن درصورتی‌که جمله داخل کروشه ($[\![.]\!]$) درست باشد، این عبارت برابر با ۱ و در غیر این صورت برابر با ۰ است. با این معادله ارتباط میان $E_{in}(h)$ و فرضیه در حال بررسی h صریحاً مشخص شده است. به همین شکل خطای خارج-از-نمونه [۲] را به شکل زیر تعریف می‌کنیم:

$$E_{out}(h) = \mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x})]$$

که متناظر با μ در مدل ظرف است. احتمال نشان داده‌شده در این فرمول بر پایه توزیع P روی $\mathcal{X}$ قرار دارد که ما از آن برای نمونه‌برداری نقاط $\mathcal{X}$ استفاده کرده‌ایم. با جایگذاری

[۱] in-sample error
[۲] out-of-sample error

نماد E_{in} به جای ν و E_{out} به جای μ در معادل هافدینگ می‌توانیم این نامعادله را به شکل زیر بازنویسی کنیم:

$$\mathbb{P}[\mid E_{in}(h) - E_{out}(h) \mid > \epsilon] \leq 2e^{-2\epsilon^2 N} \qquad (۵-۱)$$

که در آن N تعداد مثال‌های آموزشی است. خطای E_{in} همانند ν یک متغیر تصادفی است که به نمونه تصادفی انتخاب‌شده وابسته است و E_{out} همانند μ یک ثابت ناشناخته است اما تصادفی نیست.

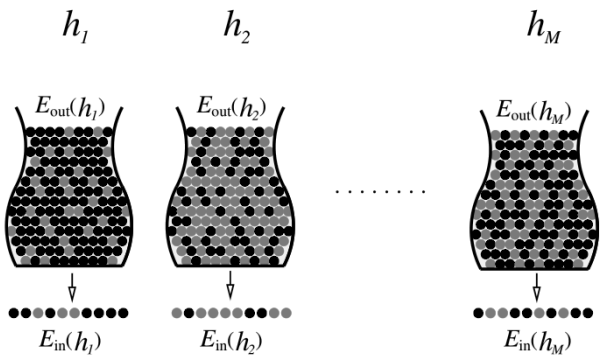
اجازه دهید به جای یک h منحصربه‌فرد تمام مجموعه فرضیات $\mathcal{H}$ را در نظر بگیریم. برای سادگی، ابتدا حالتی را در نظر می‌گیریم که این مجموعه شامل تعداد فرضیات متناهی است:

$$\mathcal{H} = \{h_1, \ldots, h_M\}$$

در این حالت می‌توان با فرض داشتن M ظرف متفاوت، تناظر مدل ظرف را برای تمامی این توابع برقرار ساخت (شکل ۱-۱۰). هر ظرف همچنان فضای ورودی $\mathcal{X}$ را نشان می‌دهد. مهره‌های قرمز در ظرف m ام برای $m = 1 \ldots M$ نشان‌دهنده نقاط $\mathbf{x} \in \mathcal{X}$ هستند که برای آنها $h_m(\mathbf{x}) \neq f(\mathbf{x})$ است. احتمال مهره‌های قرمز در داخل ظرف m برابر با $E_{out}(h_m)$ است و درصد مهره‌های قرمز در نمونه m ام برداشته‌شده از این ظرف برابر با $E_{in}(h_m)$ است. هرچند که نامعادله هافدینگ روی هر ظرف به شکل مستقل قابل‌اعمال است، اما زمانی که ما همه ظرف‌ها را هم‌زمان در نظر بگیریم، وضعیت پیچیده‌تر می‌شود. چرا؟ نامعادله هافدینگ برای یک h مشخص بیان می‌کند:

$$\mathbb{P}[\mid E_{in}(h) - E_{out}(h) \mid > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

که در آن قبل از اینکه ما مجموعه داده را تولید کنیم، فرضیه h تثبیت شده است (از قبل انتخاب شده است) و احتمال تنها به مجموعه داده تصادفی وابسته است. تأکید می‌کنیم که فرض " h قبل از تولید مجموعه داده تثبیت می‌شود" برای معتبر بودن این کران بسیار حیاتی است. اگر شما اجازه داشتید h را بعد از ایجاد مجموعه داده تغییر دهید، در آن صورت فرضیاتی که برای اثبات نامعادله هافدینگ مورد نیاز هستند، دیگر برقرار نخواهند بود.



شکل ۱-۱۰: ظروف چندگانه نشاندهنده مسئله یادگیری با M فرضیه

با داشتن چند فرضیه در $\mathcal{H}$، الگوریتم یادگیری فرضیه نهایی g را بر اساس $\mathcal{D}$ انتخاب می‌کند. به‌عبارت‌دیگر فرضیه g پس از اینکه مجموعه داده تولید شد انتخاب می‌شود.

جمله‌ای که ما قصد داریم اثبات کنیم این نیست که برای هر $h_m \in \mathcal{H}$:

$\mathbb{P}[\mid E_{in}(h_m) - E_{out}(h_m) \mid > \epsilon]$ کوچک است.

بلکه قصد داریم نشان دهیم:

$$\mathbb{P}[\mid E_{in}(g) - E_{out}(g) \mid > \epsilon]$$

برای فرضیه نهایی g کوچک است.

فرضیه g قبل از ایجاد مجموعه داده مشخص نیست و اینکه چه فرضیه‌ای به‌عنوان g انتخاب خواهد شد به داده‌ها بستگی دارد. به همین دلیل نمی‌توان به سادگی در نامعادله هافدینگ جای h را با g عوض کرد. یک راه برای دور زدن این مشکل این است که این کران را طوری محاسبه کنیم که مستقل از این باشد که الگوریتم یادگیری کدام h را به‌عنوان g انتخاب می‌کند. روشی ساده اما خیلی کلی برای این کار وجود دارد. فارغ از نوع الگوریتم و محتوای مجموعه داده، ما می‌دانیم که g باید یکی از h ها باشد. درنتیجه خواهیم داشت:

$$|E_{in}(g) - E_{out}(g)| > \epsilon \Longrightarrow |E_{in}(h_1) - E_{out}(h_1)| > \epsilon$$
$$\textbf{or} |E_{in}(h_2) - E_{out}(h_2)| > \epsilon$$
$$\textbf{or} \ldots$$
$$\textbf{or} |E_{in}(h_M) - E_{out}(h_M)| > \epsilon$$

که در آن $\beta_1 \Longrightarrow \beta_2$ به این معنی است که β_1 ، β_2 را التزام می‌کند.
همین‌طور طبق قانون احتمال داریم:

$$\texttt{if } \beta_1 \Longrightarrow \beta_2 \texttt{ then } \mathbb{P}[\beta_1] \leq \beta_2$$

باز طبق قانون کران اجتماع در احتمال داریم:

$$\mathbb{P}[\beta_1 \text{ or } \beta_2 \text{ or} \ldots \text{or } \beta_M] \leq \mathbb{P}[\beta_1] + \mathbb{P}[\beta_2 + \cdots + \mathbb{P}[\beta_M]]$$

با ترکیب این فرمول و اعمال هم‌زمان نامعادله هافدینگ به کلیه M فرضیه h_m، می‌توانیم هر یک را با $2e^{-2\epsilon^2 N}$ کران‌دار کنیم و درنتیجه خواهیم داشت:

$$\mathbb{P}[|\ E_{in}(g) - E_{out}(g)\ | > \epsilon] \leq 2M2e^{-2\epsilon^2 N} \qquad (۶\text{-}۱)$$

از لحاظ ریاضی این نامعادله یک نسخه "یکنواخت" از نامعادله هافدینگ در ۱-۵ است. ما سعی می‌کنیم به‌طور هم‌زمان همه $E_{out}(h_m)$ ها را با $E_{in}(h_m)$ متناظرشان تقریب بزنیم. سپس الگوریتم یادگیری فرضیه‌ای که دارای کمترین E_{in} است را انتخاب می‌کند و انتظار دارد فارغ از اینکه چه فرضیه‌ای انتخاب می‌شود، E_{out} متناظر به شکل یکنواخت از کران وضع‌شده روی کل مجموعه فرضیات پیروی کند.

عیب اصلی تخمین فوق این است که کران احتمالاتی $2M2e^{-2\epsilon^2 N}$ این بار وابسته به M است و به همین نسبت از کران وضع‌شده برای یک فرضیه منفرد سُست‌تر[۱] است. همین‌طور زمانی که تعداد فرضیات M نامتناهی است، این تخمین دیگر معنایی ندارد. در فصل بعد سعی می‌کنیم این کران را بهبود ببخشیم.

[۱] Loose

۱-۴-۳ امکان‌پذیری یادگیری

ما تا اینجا دو گزاره متضاد را در مورد امکان‌پذیری یادگیری ارائه کردیم. یک گزاره می‌گوید که ما قادر به یادگیری چیزی خارج از $\mathcal{D}$ نیستیم، درحالی‌که گزاره دوم این عمل را ممکن می‌داند. در این قسمت قصد داریم مصالحه‌ای میان این دو گزاره برقرار سازیم و به این سؤال پاسخ دهیم که چه زمانی یادگیری امکان‌پذیر است.

۱- اینکه آیا $\mathcal{D}$ چیزی خارج از خود به ما می‌گوید که ما قبلاً نمی‌دانستیم دو پاسخ متفاوت دارد. اگر به دنبال یک پاسخ قطعی باشیم به این معنی که $\mathcal{D}$ به شکل قطعی چیزی در مورد f خارج از خود به ما می‌گوید، در آن صورت پاسخ خیر است. اما اگر به یک پاسخ احتمالاتی قانع باشیم به این معنی که $\mathcal{D}$ چیز محتملی در مورد f خارج از خود به ما می‌گوید، در آن صورت جواب بله است.

با استفاده از دید احتمالاتی، ما بدون اینکه هزینه زیادی کنیم، به یک جواب بله برای امکان‌پذیری یادگیری رسیدیم. تنها فرضی که ما در مورد این چارچوب احتمالاتی داشتیم این است که مثال‌ها در $\mathcal{D}$ باید به شکل مستقل تولید شوند. ما هیچ اصراری بر استفاده از یک تابع توزیع احتمال خاص نداریم و یا حتی اینکه بدانیم چه توزیعی استفاده شده است. هرچند هر توزیعی که ما برای تولید مثال‌ها استفاده می‌کنیم، باید از همان توزیع برای ارزیابی g زمانی که قصد داریم بدانیم g به چه نسبت f را تقریب می‌زند، استفاده کنیم (شکل ۱-۱۱). این فرضی است که به ما امکان استفاده از معادله هافدینگ را می‌دهد. البته ممکن است در عمل همیشه این وضعیت ایدئال اتفاق نیافتد و حالت‌های مختلفی از آن در حوزه یادگیری موردبررسی قرار گرفته است.

۲- اجازه دهید در اینجا منظورمان از امکان‌پذیری یادگیری را روشن سازیم. یادگیری فرضیه g را برای تقریب تابع هدف ناشناخته f تولید می‌کند. اگر یادگیری موفقیت‌آمیز باشد در آن صورت g باید به‌خوبی f را تقریب بزند، به این معنی که $E_{out}(g) \approx 0$ باشد. هرچند این نتیجه چیزی نیست که ما از تحلیل احتمالاتی به دست آوردیم. چیزی که ما به دست آوردیم این است که $E_{out}(g) \approx E_{in}(g)$ است. پس همچنان ما باید سعی کنیم $E_{in}(g)$ را به صفر نزدیک کنیم، تا بتوانیم نتیجه بگیریم که $E_{out}(g)$ نیز به صفر نزدیک است.

ما نمی‌توانیم از قبل تضمین کنیم که فرضیه‌ای در مجموعه فرضیات خواهیم یافت که $E_{in}(g)$ برای آن صفر خواهد شد، مگر آنکه عملاً آن را بیابیم. به خاطر داشته باشید که

$E_{out}(g)$ کمیتی ناشناخته است زیرا f ناشناخته است، اما $E_{in}(g)$ کمیتی است که می‌توانیم آن را ارزیابی کنیم. درنتیجه ما شرط $E_{out}(g) \approx 0$ را که نمی‌توانیم در مورد آن ادعایی داشته باشیم را با شرط $E_{in}(g) \approx 0$ که می‌توانیم آن را تصدیق کنیم معاوضه کردیم. چیزی که به ما این امکان را می‌دهد، نامعادله هافدینگ است:

$$\mathbb{P}[\mid E_{in}(g) - E_{out}(g) \mid > \epsilon] \leq 2M2e^{-2\epsilon^2 N}$$

که به ما اطمینان می‌دهد $E_{out}(g) \approx E_{in}(g)$ و بنابراین می‌توانیم از E_{in} به‌عنوان نماینده‌ای از E_{out} استفاده کنیم.

البته مواقعی نیز وجود دارد که ما اصراری نداریم $E_{in}(g) \approx 0$ باشد. برای مثال در پیش‌بینی‌های مالی به علت غیرقابل‌پیش‌بینی بودن وضعیت بازار این انتظار که خطای یک پیش‌بینی به صفر نزدیک باشد، انتظار بالایی است. چیزی که ما امیدواریم داشته باشیم یک پیش‌بینی است که اغلب اوقات درست باشد. به این معنی که فرضیه‌ای که $E_{in}(g)$ آن زیر 0.5 باشد برای ما مطلوب است، البته به این شرط که $E_{out}(g)$ به اندازه کافی به $E_{in}(g)$ نزدیک باشد.

با توجه به این توضیحات، امکان‌پذیری یادگیری را می‌توان به دو سؤال اساسی تقسیم کرد:

۱. آیا می‌توانیم مطمئن باشیم که $E_{out}(g)$ به اندازه کافی به $E_{in}(g)$ نزدیک است؟

۲. آیا ما می‌توانیم $E_{in}(g)$ را به اندازه کافی کوچک کنیم؟

نامعادله هافدینگ به سؤال اول پاسخ می‌دهد. سؤال دوم زمانی پاسخ داده می‌شود که ما الگوریتم یادگیری را روی داده‌های واقعی اعمال کنیم و ببینیم که تا چه اندازه می‌توانیم E_{in} را کوچک کنیم. تفکیک مسئله امکان‌پذیری یادگیری به این دو سؤال بینش عمیق‌تری را نسبت به نقشی که هر کدام از اجزاء یادگیری بازی می‌کنند فراهم می‌کند. این بینش نسبت به "پیچیدگی" هر کدام از این اجزاء است که در ادامه به آنها خواهیم پرداخت.

پیچیدگی $\mathcal{H}$: اگر تعداد فرضیات M زیاد شود، طبق نامعادله هافدینگ ریسک اینکه $E_{in}(g)$ تخمین خوبی از $E_{out}(g)$ باشد افزایش می‌یابد. می‌توان M را به‌عنوان مقیاسی از پیچیدگی مجموعه فرضیات $\mathcal{H}$ قلمداد کرد. اگر قصد داریم پاسخ مثبتی به سؤال اول بدهیم، باید همواره

بر پیچیدگی $\mathcal{H}$ نظارت داشته باشیم. هرچند اگر قصد داریم پاسخ مثبتی به سؤال دوم بدهیم، با $\mathcal{H}$ پیچیده‌تر شانس ما بیشتر خواهد بود؛ زیرا g باید نهایتاً از $\mathcal{H}$ انتخاب شود و یک $\mathcal{H}$ پیچیده انعطاف بیشتری را برای یافتن g ای که بتواند داده‌ها را به‌خوبی برازش کند فراهم می‌کند و این امر متعاقباً منتهی به یک $E_{in}(g)$ کوچک خواهد شد. ایجاد این موازنه در پیچیدگی $\mathcal{H}$ یک تِم اصلی در نظریه یادگیری است که در فصل بعد بیشتر به آن خواهیم پرداخت.

پیچیدگی f:به نظر می‌رسد یک تابع هدف پیچیده‌تر نسبت به یک تابع ساده‌تر، هدف سخت‌تری برای یادگیری باشد. اجازه دهید بررسی کنیم آیا این موضوع طبق دو سؤالی که در بالا پرسیدیم قابل توجیه است. نگاهی دقیق‌تر به نامعادله ۱-۶ آشکار می‌کند که پیچیدگی $\mathcal{H}$ تأثیری روی تخمین $E_{in}(g)$ از $E_{out}(g)$ ندارد. اگر ما مجموعه فرضیات و همین‌طور تعداد مثال‌های آموزشی را ثابت نگاه داریم، در آن صورت چه بخواهیم یک تابع ساده (مانند یک تابع ثابت) را یاد بگیریم و چه بخواهیم یک تابع پیچیده (مانند یک تابع غیرخطی) را یاد بگیریم، در هر دو مورد نامعادله کران یکسانی را تولید خواهد کرد. هرچند این به این معنی نیست که می‌توانیم توابع پیچیده را به‌آسانی توابع ساده یاد بگیریم. در نظر داشته باشید که نامعادله ۱-۶ تنها به سؤال اول پاسخ می‌دهد. چنانچه تابع هدف پیچیده باشد، سؤال دوم نقش بازی خواهد کرد؛ زیرا برازش داده‌هایی که از یک تابع پیچیده می‌آیند نسبت به برازش داده‌های یک تابع ساده‌تر با سختی بیشتری همراه است. به عبارت دیگر زمانی که f پیچیده است، مقدار بدتری برای $E_{in}(g)$ حاصل خواهد شد. برای رفع این مشکل ممکن است سعی کنیم مجموعه فرضیات $\mathcal{H}$ را پیچیده‌تر کنیم به این امید که داده‌ها را بهتر برازش کرده و به $E_{in}(g)$ کوچک‌تری دست پیدا کنیم؛ اما در آن حالت طبق نامعادله ۱-۶، دیگر $E_{out}(g)$ به $E_{in}(g)$ نزدیک نخواهد بود. از هر منظر که به این مسئله نگاه کنیم یک تابع پیچیده‌تر هدف سخت‌تری برای یادگیری است. در بدترین حالت، چنانچه f بیش‌ازحد پیچیده باشد، ممکن است هرگز قادر به یادگیری آن نباشیم.

خوشبختانه بسیاری از توابعی که ما در زندگی واقعی با آنها سروکار داریم، خیلی پیچیده نیستند و ما قادریم با استفاده از یک مجموعه داده مناسب و یک مجموعه فرضیات معقول آنها را یاد بگیریم. البته این حرف بیشتر یک مشاهده تجربی است تا یک قضیه ریاضی. حتی اگر نتوانیم یک f خاص را یاد بگیریم، حداقل قادریم بگوییم که این امر میسر نیست. تا زمانی که مطمئن هستیم پیچیدگی $\mathcal{H}$ به ما یک کران هافدینگ خوب می‌دهد، موفقیت یا شکست ما در

یادگیری f تنها به‌وسیله موفقیت یا شکست ما در برازش داده‌های آموزشی تعیین می‌شود.

۱-۵ خطا و نویز

این فصل را با مرور دو مفهوم مهم در مبحث یادگیری با هدف نزدیک ساختن آنها به مسائل جهان واقعی به پایان می‌بریم. مفهوم اول در رابطه با معنی 'تقریب' است زمانی که می‌گوییم «فرضیه بدست آمده 'تقریب' خوبی از تابع f فراهم می‌کند». مفهوم دوم در مورد ذات تابع هدف است. در بسیاری از مواقع وجود نویز باعث می‌شود تا خروجی f به شکل مستقیم و یکتا توسط ورودی تعیین نشود. پیامدهای چنین تابع نویزی در مسئله یادگیری چه هستند؟ در ادامه به توضیح هر یک از این موارد می‌پردازیم.

۱-۵-۱ اندازه‌گیری خطا

از یادگیری انتظار نمی‌رود که تابع هدف را به‌طور کامل تکرار کند. فرضیه g نهایی تنها تقریبی از f است. برای کمّی‌سازی تقریب g از f، نیاز به معرفی روشی برای اندازه‌گیری خطا داریم تا بتوانیم میزان دوری فرضیه g از هدف را کمّی کنیم.

مقیاس خطایی که انتخاب می‌کنیم تأثیر زیادی روی نتیجه فرآیند یادگیری دارد. حتی اگر تابع هدف و داده‌ها یکسان باشند، مقیاس خطاهای متفاوت ممکن است منتهی به فرضیه‌های نهایی متفاوتی شوند. زیرا ممکن است میزان خطا بر اساس یک مقیاس خطا کم باشد، درحالی‌که در همان وضعیت با استفاده از یک مقیاس خطای دیگر زیاد باشد. درنتیجه مقیاس خطایی که استفاده می‌کنیم بر آنچه یاد می‌گیریم اثر می‌گذارد. معیار ما برای ترجیح یک مقیاس خطا به مقیاس دیگر چیست؟ در ادامه به این موضوع خواهیم پرداخت.

اجازه دهید ابتدا کمی مفهوم بالا را فرموله کنیم. مقیاس خطا تقریبی که هر فرضیه h در مدل از تابع هدف فراهم می‌کند را کمّی می‌کند.

$$Error = E(h, f).$$

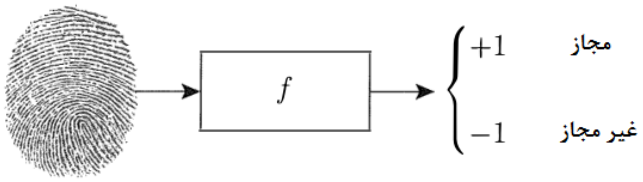
درحالی‌که اصولاً $E(h, f)$ بر اساس تمامیت توابع h و f قرار دارد، اما عموماً بر اساس خطای نقاط منفرد ورودی $\mathbf{x}$ تعریف می‌شود. اگر یک مقیاس خطای نقطه‌به‌نقطه $E(h(\mathbf{x}), f(\mathbf{x}))$ را تعریف کنیم، خطای کلی مقدار میانگین این خطای نقطه‌به‌نقطه خواهد بود.

تاکنون ما با خطای طبقه‌بندی کار می‌کردیم که به شکل زیر بود:

$$e(h(\mathbf{x}), f(\mathbf{x})) = [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$$

به شکل ایدئال $E(h, f)$ باید توسط کاربر تعیین شود. ممکن است یک تکلیف یادگیری واحد در شرایط مختلف نیاز به استفاده از مقیاس خطاهای متفاوت داشته باشد. می‌توان به $E(h, f)$ به‌عنوان هزینه استفاده از h زمانی که باید f استفاده می‌شد، نگاه کرد. این هزینه به این بستگی دارد که h به چه منظور استفاده می‌شود و نمی‌تواند تنها توسط روش یادگیری دیکته شود. در اینجا یک مثال ارائه می‌کنیم.

مثال ۱-۱ (سامانه تشخیص اثرانگشت): مسئله تشخیص اثرانگشت را در نظر بگیرید که در آن قصد داریم بدانیم آیا یک اثرانگشت متعلق به شخص خاصی هست یا خیر. مقیاس خطای مناسب در این مسئله چیست؟



تابع هدف اثرانگشت را به‌عنوان ورودی دریافت می‌کند و چنانچه متعلق به یک فرد مجاز باشد 1 و در غیر این صورت -1 را برمی‌گرداند. در این وضعیت دو نوع خطا ممکن است اتفاق بیافتد. اگر یک شخص مجاز رد شود، در آن صورت h برابر با -1 است درحالی‌که f برابر با 1 است. به این حالت رد اشتباه [۱] گفته می‌شود. در مقابل اگر یک شخص غیرمجاز تائید شود به این معنی که h برابر با $+1$ و f برابر با -1 باشد، به آن قبول اشتباه [۲] گفته می‌شود. زمانی که مقادیر این دو تابع یکسان باشد، خطایی اتفاق نیافتده است.

[۱] False Reject
[۲] False Accept

		f	
		+1	-1
h	+1	no error	false accept
	-1	false reject	no error

مقیاس خطا در این مثال چگونه باید تعریف شود؟ اگر یک شخص مجاز تائید شود و یا شخص غیرمجازی رد شود، مسلماً خطا صفر است. ما باید مقدار خطا را برای حالت‌های قبول اشتباه و رد اشتباه مشخص کنیم. مقدار مناسب به کاربرد بستگی دارد.

دو مشتری بالقوه برای سیستم تشخیص اثر انگشت را در نظر بگیرید. یکی از آنها می‌تواند مغازه‌داری باشد که از این سامانه برای تشخیص مشتریانی که عضو یک برنامه تخفیف هستند استفاده می‌کند. مشتری دوم یک سازمان امنیتی است که از این سامانه در ورودی یک مکان سری استفاده می‌کند تا تشخیص دهد آیا افراد اجازه ورود به آن مکان را دارند یا خیر.

برای مغازه‌دار یک رد اشتباه، به این معنی که مشتری عضو بوده اما رد شده است، هزینه‌بر خواهد بود. اگر یک مشتری به‌اشتباه رد شود، ممکن است دلسرد شده و در آینده به آن مغازه وفادار باقی نماند. در این حالت سودی که می‌توانست از این مشتری عاید مغازه‌دار شود از دست خواهد رفت. اما از سوی دیگر هزینه یک قبول اشتباه بالا نیست. در این حالت شما به شخصی که استحقاق نداشته است، مقداری تخفیف داده‌اید. البته کسی هم که چنین کاری کند باید شخص شجاعی باشد.

درمقابل برای سازمان امنیتی یک قبول اشتباه ممکن است فاجعه‌بار باشد. در این حالت یک شخص غیرمجاز اجازه ورود به یک مکان بسیار حساس را پیدا کرده است. این موضوع خود را در هزینه بالای قبول اشتباه نشان می‌دهد. اما در این مورد رد اشتباه چندان هزینه‌بر نخواهد بود، زیرا افراد مجاز کارمندان سازمان هستند و این که به خاطر رد اشتباه مجبور شوند دوباره سعی کنند را جزئی از شغل خود می‌دانند و با آن کنار می‌آیند.

می‌توان هزینه انواع مختلف خطا را به‌وسیله یک ماتریس نشان داد. برای مثال‌های بالا، هزینه‌ها به شکل زیر است:

		f	
		+1	-1
h	+1	0	1
	-1	10	0

(الف) مغازه دار

		f	
		+1	-1
h	+1	0	1000
	-1	1	0

(ب) سازمان امنیتی

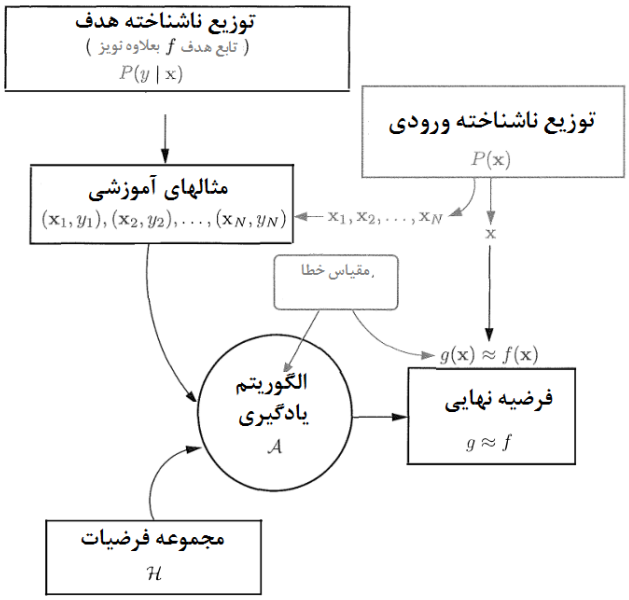
مقادیر این ماتریس باید برای وزن دهی انواع مختلف خطا به هنگام محاسبه خطای کلی استفاده شوند. در آن صورت زمانی که الگوریتم یادگیری سعی می‌کند این مقیاس خطای وزن دهی شده توسط هزینه‌ها را کاهش دهد، به‌طور ضمنی کاربرد فرضیه تولیدشده را نیز به حساب خواهد آورد. این امر ممکن است در سناریوهای سازمان امنیتی و مغازه‌دار منتهی به دو فرضیه نهایی کاملاً متفاوت شود.

هدف از ارائه این مثال این بود که نشان دهیم چگونه انتخاب مقیاس خطا می‌تواند وابسته به کاربرد سیستم باشد، تا اینکه صرفاً یک معیار ذاتی بوده و به شکل مستقل در فرایند یادگیری قابل تعیین باشد. بااین‌حال در عمل این انتخاب ایدئال ممکن است به دو دلیل امکان‌پذیر نباشد. اول اینکه ممکن است کاربر مشخصات انواع خطا را ارائه نکند و این امر خیلی غیرمعمول نیست. دوم اینکه ممکن است هزینه وزن دهی شده تابع سختی برای روش‌های بهینه‌سازی باشد. بنابراین ما برای تعریف مقیاس خطا اغلب به دنبال راه‌های دیگری هستیم و برخی اوقات ممکن است آن را با توجه به برخی ملاحظات تحلیلی و یا عملی تعریف کنیم. ما تاکنون مثالی از این حالت را با ارائه مقیاس خطای دودویی در این فصل مشاهده کردیم و مقیاس خطاهای دیگری را نیز در فصل‌های آینده ارائه خواهیم کرد.

۱-۵-۲ اهداف نویزی

در بسیاری از کاربردهای عملی، داده‌های مورد نیاز یادگیری توسط یک تابع هدف قطعی تولید نمی‌شوند، بلکه در یک فرایند نویزی تولید می‌شوند به شکلی که خروجی منحصراً توسط ورودی تعیین نمی‌شود. برای نمونه در مثال کارت اعتباری ارائه‌شده در فصل ۱.۱، دو مشتری ممکن است دارای حقوق یکسان، وام‌های عمده یکسان و مشخصات یکسان دیگری باشند، اما از لحاظ اعتباری رفتارهای متفاوتی نشان دهند. درنتیجه تابع هدف، یک تابع قطعی نیست بلکه تابعی نویزی است. بااین‌حال این وضعیت را نیز می‌توان توسط همان چارچوب قبلی مدل‌سازی کرد. در این حالت، به جای $y = f(\mathbf{x})$ می‌توانیم y را به شکل یک متغیر تصادفی در نظر بگیریم که متأثر از ورودی $\mathbf{x}$ است (به جای اینکه مستقیماً توسط ورودی تعیین شود). به شکل رسمی‌تر به جای اینکه دقیقاً تابع هدف $f(\mathbf{x})$ را داشته باشیم، یک توزیع هدف $P(y \mid \mathbf{x})$ را خواهیم داشت. در این وضعیت، یک نقطه‌داده $(\mathbf{x}, y)$ با ترکیب دو توزیع احتمال $P(\mathbf{x}, y) = P(\mathbf{x})P(y \mid \mathbf{x})$ تولید می‌شود.

می‌توان یک هدف نویزی را به شکل یک تابع قطعی به اضافه نویز ملاحظه کرد. اگر برای



شکل ۱-۱۱: مسئله عمومی یادگیری باناظر

مثال y مقداری حقیقی باشد، می‌توان مقدار مورد انتظار y برای $\mathbf{x}$ داده‌شده را با تابع قطعی $f(\mathbf{x})$ مشخص کرد و $y - f(\mathbf{x})$ را به‌عنوان نویز خالصی که به f اضافه شده است در نظر گرفت. از این منظر حتی یک تابع هدف قطعی را می‌توان شکل خاصی از هدف نویزی تلقی کرد که در آن نویز صفر است. به شکل رسمی هر تابع f را می‌توان به شکل توزیع $P(y \mid \mathbf{x})$ نشان داد که در آن احتمال $P(y \mid \mathbf{x})$ برای همه y ها به‌جز $y = f(\mathbf{x})$صفر است. درنتیجه چنانچه هدف یادگیری را به جای یک تابع قطعی، یک توزیع احتمال در نظر بگیریم، چیزی از کلیت مسئله کاسته نمی‌شود. شکل ۱-۱۱ نسخه اصلاح‌شده مسئله عمومی یادگیری که قبلاً در شکل‌های ۱-۲ و ۱-۹ نشان داده‌شده بود را برای پوشش هر دو حالت قطعی و حالت نویزی نشان می‌دهد.

نقش توزیع $P(y \mid \mathbf{x})$ در مسئله یادگیری متفاوت از نقش توزیع $P(\mathbf{x})$ است. درحالی‌که هر دو توزیع وجوه احتمالاتی $\mathbf{x}$ و y را مدل‌سازی می‌کنند، توزیع $P(y \mid \mathbf{x})$ هدفی است که ما سعی داریم آن را یاد بگیریم، درحالی‌که توزیع $P(\mathbf{x})$ اهمیت نسبی نقطه $\mathbf{x}$ را برای اندازه‌گیری میزان یادگیری صورت گرفته مشخص می‌کند.

تمام تحلیل‌هایی که تاکنون در مورد امکان‌پذیری یادگیری ارائه کردیم، به یک هدف نویزی

نیز قابل تعمیم است. دلیل ابتدایی این است که نامعادله هافدینگ به هر نوع تابع هدف دلخواه ناشناخته قابل‌اعمال است. فرض کنید ما تمام y ها را طبق توزیع $P(y \mid \mathbf{x})$ به شکل تصادفی روی فضای ورودی $\mathcal{X}$ انتخاب کرده‌ایم. این تحقق خاص از $P(y \mid \mathbf{x})$ خود می‌تواند یک تابع هدف باشد، درنتیجه نامعادله هافدینگ فارغ از اینکه تحقق تصادفی تابع هدف به چه شکلی است همچنان معتبر است.

البته این به این معنی نیست که یادگیری یک هدف نویزی به‌آسانی یادگیری یک هدف قطعی است. دو سؤال یادگیری را به خاطر بیاورید. با استفاده از یک مدل یادگیری یکسان، ممکن است برای یک تابع نویزی، اختلاف E_{out} و E_{in} به همان نسبت یک تابع قطعی باشد، اما E_{in} برای تابع نویزی به‌مراتب بدتر خواهد بود زیرا برازش داده‌های همراه با نویز بسیار دشوارتر است.

در فصل ۲ زمانی که نسخه قوی‌تری از ۱-۶ را ارائه کردیم، تابع هدف را به شکل توزیع $P(y \mid \mathbf{x})$ در نظر خواهیم گرفت تا یک مورد کلی را پوشش دهیم.

فصل دوم

آموزش در مقابل آزمایش

قبل از امتحان نهایی ممکن است استاد یک سری سؤالات و راه‌حلشان را در اختیار دانش‌آموزان قرار دهد. هرچند این سؤالات دقیقاً سؤالات امتحان نیستند، اما مطالعه آنها به شما کمک می‌کند که عملکرد بهتری در امتحان داشته باشید. این سؤالات معادل "مجموعه آموزشی" در مسئله یادگیری هستند.

اگر هدف استاد این است که به شما در امتحان کمک کند، پس چرا اصل سؤالات را به شما نمی‌دهد؟! در واقع کسب نمره خوب در امتحان به‌خودی‌خود هدف نیست، هدف این است که شما محتویات درس را به‌خوبی یاد بگیرید و امتحان تنها معیاری برای سنجش میزان یادگیری شما است. اگر شما سؤالات امتحان را از قبل بدانید، نمره‌ای که می‌گیرید دیگر نشان‌دهنده میزان یادگیری‌تان نخواهد بود.

همین تمایز را می‌توان میان مجموعه آموزشی و مجموعه آزمایشی در مسئله یادگیری قائل شد. در این فصل یک نظریه ریاضی را برای یادگیری ارائه خواهیم داد که وجوه مختلف این تمایز را مشخص می‌کند. همین‌طور پیامدهای نظری و عملی این تمایز را مورد بحث قرار خواهیم داد.

۲-۱ نظریه تعمیم

خطای خارج-از-نمونه E_{out} به ما نشان می‌دهد که تا چه اندازه آموزش بر روی $\mathcal{D}$ به داده‌های که قبلاً ندیده بودیم تعمیم پیدا کرده است. درواقع E_{out} نشان‌دهنده کارایی روی کل فضای ورودی $\mathcal{X}$ است. درنتیجه اگر قصد داریم مقدار E_{out} را با استفاده از نمونه‌ای از نقطه‌داده‌ها تقریب بزنیم، این نقاط باید نقاطی دست‌نخورده باشند و قبلاً در مرحله آموزش

استفاده نشده باشند، همانند سؤالات امتحان نهایی که قبلاً در کلاس گفته نشده‌اند.

در مقابل، خطای درون-نمونه E_{in} بر اساس نقطه‌داده‌هایی به دست می‌آید که برای آموزش استفاده شده‌اند. این مقدار مشخصاً بازدهی آموزش را اندازه می‌گیرد و مانند بازدهی یک دانش‌آموز روی سؤالات داده‌شده قبل از امتحان نهایی است. این بازدهی از این مزیت سود می‌برد که دانش‌آموز به راه‌حل‌ها دسترسی دارد و می‌تواند خود را با سؤالات هماهنگ کند، درنتیجه این بازدهی ممکن است بازتاب‌دهنده بازدهی واقعی وی در امتحان اصلی نباشد.

ما تحلیل خطای E_{in} را در فصل یک شروع کردیم و در این فصل آن را به یک مورد کلی گسترش می‌دهیم. همین‌طور تفاوت مجموعه آموزشی و آزمایشی را بیشتر توضیح خواهیم داد.

۲-۱-۱ خطای تعمیم

تاکنون بحث کردیم که مقدار خطای E_{in} همیشه قابل‌تعمیم به مقدار خطای E_{out} نیست. قابلیت تعمیم نکته‌ای کلیدی در مسئله یادگیری است. می‌توان خطای تعمیم را به‌عنوان اختلاف میان E_{in} و E_{out} تعریف کرد. نامعادله هافدینگ یک کران احتمالاتی را برای خطای تعمیم ارائه می‌کند:

$$\mathbb{P}[\mid E_{in}(g) - E_{out}(g) \mid > \epsilon] \leq 2Me^{-2\epsilon^2 N}$$

برای یک $\epsilon > 0$ می‌توان این فرمول را به شکل دیگری بازنویسی کرد. یک سطح تحمل δ را در نظر بگیرید (برای مثال $\delta = 0.05$) و با احتمال $1 - \delta$ کران زیر را به دست بیاورید:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}} \qquad (۲-۱)$$

ما به این نامعادله "کران تعمیم" [۱] می‌گوییم، زیرا کرانی را برای E_{out} برحسب E_{in} ایجاد می‌کند. برای اینکه متوجه شوید چگونه این نامعادله از نامعادله هافدینگ در ۱-۶ نتیجه می‌شود، نامعادله هافدینگ را به شکل زیر بازنویسی کنید:

با احتمال حداقل $1 - 2Me^{-2\epsilon^2 N}$ خواهیم داشت $|E_{out} - E_{in}| \leq \epsilon$ که نتیجه می‌دهد $E_{out} \leq E_{in} + \epsilon$. می‌توان تشخیص داد که $\delta = 2Me^{-2\epsilon^2 N}$ است و طبق آن ϵ به شکل $\epsilon = \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$ حاصل می‌شود و معادله ۲-۱ بدست می‌آید.

[۱] generalization bound

دقت کنید که نامعادله $|E_{out} - E_{in}| \leq \epsilon$ طرف دیگری به شکل $E_{out} \geq E_{in} - \epsilon$ نیز دارد که برای تمامی $h \in \mathcal{H}$ برقرار است. این نتیجه نیز در مسئله یادگیری مهم است. زیرا نه‌تنها ما مایلیم بدانیم که فرضیه انتخاب‌شده g (فرض کنید فرضیه‌ای که کمترین خطا را در درون نمونه دارد) در خارج از نمونه نیز به‌خوبی درون نمونه کار می‌کند ($E_{out} \leq E_{in} + \epsilon$)، بلکه همین‌طور می‌خواهیم مطمئن باشیم که ما بهترین فرضیه ممکن را از مجموعه فرضیات $\mathcal{H}$ انتخاب کرده‌ایم و فرضیه دیگری E_{out} بهتری را فراهم نمی‌کند. ازآنجاکه طرف دیگر نامعادله یعنی $E_{out} \geq E_{in} - \epsilon$ برای کلیه فرضیه‌ها برقرار است، پس هر فرضیه دیگری که انتخاب شود، دارای E_{in} بالاتری از g است و درنتیجه دارای E_{out} بالاتری نیز خواهد بود.

کران خطای $\sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$ نشان داده‌شده در ۲-۱ وابسته به M، اندازه مجموعه فرضیات $\mathcal{H}$ است. اگر $\mathcal{H}$ مجموعه‌ای نامتناهی باشد این کران به سمت بی‌نهایت میل می‌کند و درنتیجه کاربرد خود را از دست خواهد داد. از طرفی اکثر مدل‌های جالب یادگیری که ما با آنها سروکار داریم دارای مجموعه فرضیات نامتناهی هستند. این وضعیت حتی شامل مدل ساده پرسپترون که در فصل ۱ مرور شد نیز می‌شود.

برای اینکه ویژگی تعمیم را به چنین مدل‌هایی بسط دهیم، نیاز به ایجاد کرانی معادل ۲-۱ برای مجموعه فرضیات نامتناهی داریم. به این منظور مایلیم M را با یک مقدار متناهی جایگزین کنیم به‌طوری‌که این کران همچنان معنی‌دار باشد. در قسمت بعد مفهومی به نام "تابع رشد[۱]" را معرفی خواهیم کرد که "تعداد مؤثر فرضیات[۲]" را فرموله می‌کند و می‌تواند جایگزین مناسبی برای M باشد.

۲-۱-۲ تعداد مؤثر فرضیات

این قسمت را با تعریف تابع رشد و مطالعه ویژگی‌های اساسی آن شروع می‌کنیم. سپس نشان می‌دهیم که چگونه می‌توان مقدار تابع رشد را کران‌دار کرد. سرانجام نشان خواهیم داد که می‌توان M در کران تعمیم را با تابع رشد جایگزین کرد. این سه گام کران تعمیمی که به آن نیاز داریم را فراهم خواهند کرد و می‌توانیم آن را به مجموعه فرضیات نامتناهی اعمال کنیم. برای شروع، ابتدا روی توابع هدف دودویی متمرکز می‌شویم، به شکلی که در آن هر $h \in \mathcal{H}$ ورودی $\mathcal{X}$ را به مجموعه $\{-1, +1\}$ نگاشت می‌دهد.

[۱] Growth function

[۲] Effective Number of Hypotheses

تعریف تابع رشد بر اساس تعداد فرضیات مختلفی است که $\mathcal{H}$ می‌تواند روی یک نمونه متناهی از نقاط پیاده‌سازی کند. دقت کنید که در این تعریف به جای کل فضای ورودی $\mathcal{X}$، ما تنها تعداد مشخصی از نقاط را به‌کارمی‌بریم. اگر $h \in \mathcal{H}$ به یک نمونه متناهی $\mathsf{x}_1, \ldots, \mathsf{x}_N \in \mathcal{X}$ اعمال شود، در آن صورت یک N-تایی به شکل $h(\mathsf{x}_1), \ldots, h(\mathsf{x}_N)$ از مقادیر ± 1 حاصل خواهد شد. چنین N-تایی یک "دوبخشی" [۱] خوانده می‌شود، زیرا نقاط $\mathsf{x}_1, \ldots, \mathsf{x}_N$ را به دو بخش تقسیم می‌کند، نقاطی که برای آنها h برابر با -1 و نقاطی که برای آنها h برابر با $+1$ است. هر $h \in \mathcal{H}$ یک دوبخشی را روی $\mathsf{x}_1, \ldots, \mathsf{x}_N$ ایجاد می‌کند، اما ممکن است دو h متفاوت دوبخشی یکسانی را تولید کنند، به این معنی که هر دو الگوی یکسانی از ± 1 را روی این نمونه تولید کنند.

تعریف ۲-۱: فرض کنید نقاط $\mathsf{x}_1, \ldots, \mathsf{x}_N$ متعلق به دامنه $\mathcal{X}$ هستند، مجموعه دو بخشی‌هایی که توسط $\mathcal{H}$ روی این نقاط تولید می‌شوند به شکل زیر تعریف می‌شود:

$$\mathcal{H}(\mathsf{x}_1, \ldots, \mathsf{x}_N) = \{(h(\mathsf{x}_1), \ldots, h(\mathsf{x}_N)) \mid h \in \mathcal{H}\} \qquad (۲\text{-}۲)$$

می‌توان دوبخشی‌های $\mathcal{H}(\mathsf{x}_1, \ldots, \mathsf{x}_N)$ را همانند خود $\mathcal{H}$ به‌عنوان یک مجموعه فرضیات در نظر گرفت با این تفاوت که این مجموعه فرضیات از منظر تنها N نقطه دیده می‌شوند. یک مجموعه $\mathcal{H}(\mathsf{x}_1, \ldots, \mathsf{x}_N)$ بزرگ‌تر به معنی تنوع بیشتر در $\mathcal{H}$ و تولید دو بخشی‌های بیشتری روی $\mathsf{x}_1, \ldots, \mathsf{x}_N$ است. تابع رشد بر اساس تعداد این دوبخشی ها تعریف می‌شود.

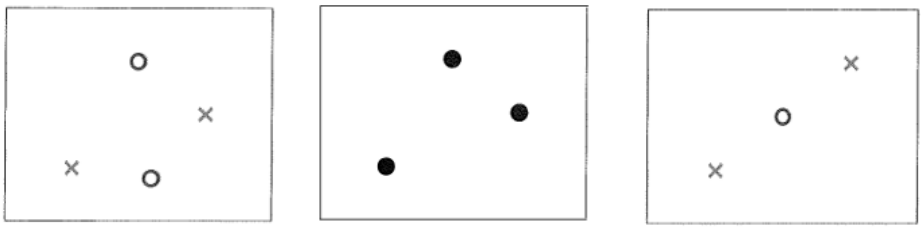
تعریف ۲-۲: تابع رشد برای یک مجموعه فرضیات $\mathcal{H}$ برابر است با:

$$m_{\mathcal{H}}(N) = \max_{\mathsf{x}_1, \ldots, \mathsf{x}_N \in \mathcal{X}} |\mathcal{H}(\mathsf{x}_1, \ldots, \mathsf{x}_N)|$$

که در آن $|.|$ به معنی تعداد اعضای یک مجموعه است.

به زبان ساده $m_{\mathcal{H}}(N)$ حداکثر تعداد دوبخشی‌هایی است که می‌توان توسط $\mathcal{H}$ روی N نقطه از دامنه $\mathcal{X}$ تولید کرد. برای اینکه $m_{\mathcal{H}}(N)$ را محاسبه کنیم، کلیه انتخاب‌های ممکن از N نقطه $\mathsf{x}_1, \ldots, \mathsf{x}_N \in \mathcal{X}$ را در نظر می‌گیریم و نمونه‌ای را انتخاب می‌کنیم که به ما بیشترین

[۱] dichotomy



(ج) (ب) (الف)

شکل ۲-۱: تابع رشد برای یک پرسپترون دو بعدی. الف) نمونه‌ای از یک دوبخشی با سه نقطه که توسط پرسپترون قابل تولید نیست. ب) سه نقطه که تمامی ۸ دوبخشی ممکن روی آنها قابل تولید است. ج) نمونه‌ای از یک دوبخشی روی چهار نقطه که توسط پرسپترون قابل تولید نیست. حداکثر ۱۴ دوبخشی از ۱۶ دوبخشی ممکن روی هر ۴ نقطه‌ای قابل تولید هستند.

دوبخشی‌ها را می‌دهد. همانند M، $m_{\mathcal{H}}(N)$ نیز مقیاسی از تعداد فرضیات موجود در $\mathcal{H}$ است، با این تفاوت که به جای کل فضای $\mathcal{X}$، هر فرضیه روی تنها N نقطه اعمال می‌شود.

ازآنجاکه برای هر $\mathcal{H}$، $\mathcal{H}(\mathsf{x}_1, \ldots, \mathsf{x}_N) \subseteq \{-1, +1\}^N$ (مجموعه تمام دوبخشی‌های ممکن روی N نقطه)، درنتیجه مقدار $m_{\mathcal{H}}(N) \leq 2^N$ است. اگر $\mathcal{H}$ بتواند همه دوبخشی‌های ممکن را روی $\mathsf{x}_1, \ldots, \mathsf{x}_N$ تولید کند، گفته می‌شود که $\mathcal{H}$، $\mathsf{x}_1, \ldots, \mathsf{x}_N$ را "خرد[۱]" می‌کند. این امر به این معنی است که $\mathcal{H}$ تا حد امکان روی این نمونه خاص متنوع است. **مثال ۲-۱**: اگر $\mathcal{X}$ یک صفحه اقلیدسی و $\mathcal{H}$ یک پرسپترون دوبعدی باشد، مقادیر $m_{\mathcal{H}}(3)$ و $m_{\mathcal{H}}(4)$ را به دست آورید. شکل ۲-۱ (الف) دوبخشی‌ای را روی سه نقطه نشان می‌دهد که پرسپترون قادر به تولید آن نیست. قسمت (ب) سه نقطه دیگر را نشان می‌دهد که پرسپترون آنها را خرد می‌کند و تمامی $2^3 = 8$ دوبخشی ممکن را روی آنها تولید می‌کند. علی رقم اینکه پرسپترون قادر نبود دوبخشی قسمت (الف) را تولید کند، اما ازآنجاکه تعریف $m_{\mathcal{H}}(N)$ بر اساس حداکثر تعداد دوبخشی ها است، بنابراین $m_{\mathcal{H}}(3) = 8$ است.

برای چهار نقطه، شکل ۲-۱ (ج) دوبخشی‌ای را نشان می‌دهد که پرسپترون قادر به تولید آن نیست. می‌توان نشان داد که هیچ چهار نقطه‌ای وجود ندارد که پرسپترون بتواند آنها را خرد کند. بیشترین تعداد دوبخشی که پرسپترون روی چهار نقطه می‌تواند تولید کند برابر با ۱۴ مورد از ۱۶ حالت ممکن است. درنتیجه $m_{\mathcal{H}}(4) = 14$ است.

---

[۱] shatter

تعریف ۲-۳: اگر $\mathcal{H}$ نتواند هیچ مجموعه داده‌ای با اندازه k را خرد کند، در آن صورت k نقطه توقف [۱] $\mathcal{H}$ نامیده می‌شود.

اگر k یک نقطه توقف $\mathcal{H}$ باشد در آن صورت $m_{\mathcal{H}}(k) < 2_k$.

مثال قبل نشان داد که $k = 4$ یک نقطه توقف برای پرسپترون‌های دوبعدی است. در حالت کلی ساده‌تر است که به جای محاسبه تابع کامل رشد $\mathcal{H}$، یک نقطه توقف را برای آن پیدا کنیم.

اکنون با استفاده از نقطه توقف k سعی داریم کرانی را برای تابع رشد $m_{\mathcal{H}}(N)$ برای تمامی مقادیر N استخراج کنیم. برای مثال این واقعیت که هیچ ۴ نقطه‌ای را نمی‌توان توسط پرسپترون دوبعدی خرد کرد، محدودیت زیادی را بر روی تعداد دوبخشی‌هایی که روی ۵ یا بیشتر نقطه توسط پرسپترون قابل‌تولید هستند ایجاد می‌کند. ما از این ایده برای ایجاد یک کران روی $m_{\mathcal{H}}(N)$ در حالت عمومی استفاده خواهیم کرد.

نکته مهم در مورد تابع رشد این است که اگر شرط $m_{\mathcal{H}}(N) = 2^N$ در یک نقطه متوقف شود، در آن صورت می‌توان بر اساس آن نقطه توقف، برای تمامی مقادیر N، کرانی را با استفاده از یک چنده جمله‌ای ساده برای $m_{\mathcal{H}}(N)$ ایجاد کرد. این موضوع که این کران چندجمله‌ای است بسیار حیاتی است. چنانچه نقطه توقفی وجود نداشته باشد، در آن صورت برای تمام N ها $m_{\mathcal{H}}(N) = 2^N$ خواهد شد. اگر این مقدار را در معادله ۲-۱ به جای M قرار دهیم، در آن صورت صرف‌نظر از تعداد مثال‌های آموزشی، کران $\sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$ هیچ‌گاه به صفر میل نخواهد کرد. اما چنانچه $m_{\mathcal{H}}(N)$ به‌وسیله یک چندجمله‌ای (هر چندجمله‌ای) کران‌دار شود، زمانی که N به سمت بی‌نهایت میل می‌کند، خطای تعمیم به سمت صفر میل می‌کند. این مطلب نشان می‌دهد که به شرط استفاده از تعداد کافی از مثال‌ها، تعمیم خوبی خواهیم داشت.

قضیه ۲-۱: اگر $m_{\mathcal{H}}(k) < 2^k$ برای برخی از مقادیر k برقرار باشد، در آن صورت برای تمام N ها خواهیم داشت:

$$m_{\mathcal{H}}(N) = \sum_{i=0}^{k-1} \binom{N}{i} \qquad (۲-۳)$$

[۱] break point

که در آن قسمت سمت راست یک چندجمله‌ای بر حسب N از درجه $k-1$ است. این کران چندجمله‌ای را می‌توان توسط استقرای ریاضی اثبات کرد، اما در اینجا ما به تعریف آن بسنده می‌کنیم. نتیجه عملی این معادله این است که چنانچه $\mathcal{H}$ داری یک نقطه توقف باشد، در آن صورت می‌توانیم از داشتن یک تعمیم خوب مطمئن باشیم. با این مقدمه و ارائه تعریف تابع رشد، در اینجا قصد داریم یک تعریف رایج از کران تعمیم را ارائه کنیم.

۲-۱-۳ بعد VC

قضیه ۲-۱ کل تابع رشد را برحسب یک نقطه توقف کران‌دار می‌کند. می‌توان هر نقطه توقفی را به این منظور انتخاب کرد، اما هر چه این مقدار کوچک‌تر باشد، کران به‌دست‌آمده نیز بهتر خواهد بود. این مطلب ما را به سمت انتخاب یک پارامتر یکتا جهت مشخص کردن تابع رشد رهنمون می‌کند.

تعریف ۲-۵: بُعد وپنیک-چروُننکیس[۱] مجموعه فرضیات $\mathcal{H}$ که با $d_{\text{vc}}(\mathcal{H})$ یا به شکل ساده‌تر d_{vc} نمایش داده می‌شود برابر با بزرگ‌ترین Nی است که برای آن $m_{\mathcal{H}}(N) = 2^N$ است. اگر برای تمام N ها، $m_{\mathcal{H}}(N) = 2^N$ باشد، در آن صورت $d_{\text{vc}} = \infty$.

اگر d_{vc} بعد VC مجموعه فرضیات $\mathcal{H}$ باشد، $k = d_{\text{vc}} + 1$ یک نقطه توقف برای $m_{\mathcal{H}}(N)$ خواهد بود. زیرا طبق تعریف، برای $N > d_{\text{vc}}$ ، $m_{\mathcal{H}}(N)$ نمی‌تواند برابر با 2^N باشد. به‌سادگی می‌توان نشان داد که نقطه توقف کوچک‌تری برای $\mathcal{H}$ نیز وجود ندارد. ازآنجاکه $\mathcal{H}$ می‌تواند d_{vc} نقطه را خرد کند، درنتیجه هر زیرمجموعه از آنها را نیز می‌تواند خرد کند.

با توجه به اینکه $k = d_{\text{vc}} + 1$ یک نقطه توقف برای $m_{\mathcal{H}}(N)$ است، می‌توان قضیه ۲-۱ را برحسب d_{vc} بازنویسی کرد:

$$m_{\mathcal{H}}(N) \leq \sum_{i=0}^{d_{\text{vc}}} \binom{N}{i} \qquad (۴-۲)$$

همانطور که مشاهده می‌شود در واقع بعد VC درجه کران چندجمله‌ای روی $m_{\mathcal{H}}(N)$ است.

[۱] Vapnik-Chervonenkis dimension

با توجه به دنباله استدلال‌های ارائه شده، این بهترین کرانی است که می‌توانستیم به دست بیاوریم، زیرا نقطه توقف $k = d_{\text{vc}} + 1$ کوچک‌ترین نقطه توقف ممکن است. این کران می‌تواند به شکل ساده‌تری نیز بیان شود که وابستگی آن به d_{vc} را روشن‌تر می‌سازد:

$$m_{\mathcal{H}}(N) \leq N_{d_{\text{vc}}} + 1 \tag{۵-۲}$$

این شکل جدید نیز با روش استقرا قابل‌اثبات است که ما از ارائه آن در اینجا صرف نظر می‌کنیم.

۲-۱-۴ کران تعمیم VC

اکنون که تابع رشد توسط ترم بعد VC کران‌دار شد، تنها یک گام تا تحلیلمان فاصله داریم و آن جایگزینی M با تابع رشد $m_{\mathcal{H}}(N)$ در کران تعمیم است. در آن صورت بعد VC نقشی محوری در پرسش تعمیم بازی خواهد کرد.

اگر تابع رشد را به‌عنوان تعداد مؤثر فرضیات در نظر بگیریم و M در کران تعمیم را با $m_{\mathcal{H}}(N)$ جایگزین کنیم فرمول زیر حاصل می‌شود:

$$E_{out} \overset{?}{\leq} E_{in} + \sqrt{\frac{1}{2N} \ln \frac{2m_{\mathcal{H}}(N)}{\delta}} \tag{۶-۲}$$

البته به دلایلی نمی‌توان $m_{\mathcal{H}}(N)$ را مستقیماً جایگزین نمود و برخی اصلاحات مورد نیاز است. در نتیجه فرمول واقعی اندکی با این فرمول تفاوت دارد که در زیر بدون اثبات این اصلاحات ارائه می‌شود:

$$E_{out} \leq E_{in} + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \tag{۷-۲}$$

مگر اینکه $d_{\text{vc}}(\mathcal{H}) = \infty$ باشد، ما مطمئن هستیم که $m_{\mathcal{H}}(N)$ با یک چند جمله‌ای بر حسب N کران‌دار است و بنابراین $\ln m_{\mathcal{H}}(2N)$ صرف نظر از درجه چندجمله‌ای به شکل لگاریتمی

بر حسب N رشد می‌کند. درنتیجه با توجه به ضریب معکوس $\frac{8}{N}$ که رشد بیشتری دارد، کران تعمیم با افزایش N کوچک می‌شود و برای یک ضریب تحمل ثابت δ، E_{out} به E_{in} نزدیک می‌شود.

تنها زمانی که $d_{\text{vc}}(\mathcal{H}) = \infty$ باشد، کران بالا صدق نمی‌کند و در این حالت تابع رشد بر حسب N نمایی است. اما برای یک مقدار متناهی d_{vc} خطای تعمیم با سرعتی که توسط d_{vc} تعیین می‌شود به سمت صفر همگرا می‌شود. از آنجا که d_{vc} درجه چندجمله‌ای است، در نتیجه هر چه d_{vc} کوچکتر باشد، همگرایی به صفر نیز سریع‌تر است.

یکی از پیامدهای این بحث این است که می‌توان مدل‌های مختلف را به دو کلاس مدل‌های خوب و مدل‌های بد تقسیم کرد. مدل‌های خوب دارای d_{vc} متناهی هستند و برای یک N به اندازه کافی بزرگ E_{in} به E_{out} نزدیک خواهد بود. برای این مدل‌ها کارایی درون-نمونه به‌خوبی به کارایی خارج-از-نمونه تعمیم پیدا می‌کند. در مقابل، مدل‌های بد دارای d_{vc} نامتناهی هستند. برای یک مدل بد، فارغ از اینکه مجموعه داده چقدر بزرگ باشد، نمی‌توان بر اساس تحلیل VC خطای E_{in} را به E_{out} تعمیم داد. هرچند باید ذکر کنیم که در برخی مدل‌ها با d_{vc} های نامتناهی مانند مجموعه‌های محدب، تحلیل جایگزینی بر اساس متوسط تابع رشد وجود دارد که رفتار تعمیمی خوبی را از خود نشان می‌دهد.

به علت نقش مهم بعد VC، در اینجا سعی می‌کنیم بینش جدیدی را در مورد آن به‌دست آوریم. یک راه برای کسب این بینش این است که سعی کنیم بعد VC را برای مدل‌های یادگیری که تاکنون با آنها آشنا شدیم محاسبه کنیم. پرسپترون‌ها یکی از مواردی هستند که قادریم d_{vc} را به شکل دقیق برای آنها محاسبه کنیم. این عمل در دو مرحله انجام می‌شود. ابتدا نشان خواهیم داد که d_{vc} حداقل یک مقدار مشخص است و سپس نشان خواهیم داد که حداکثر نیز همان مقدار است. تفاوتی منطقی میان گزاره d_{vc} حداقل مقدار مشخصی است و گزاره d_{vc} حداکثر مقدار مشخصی است وجود دارد. دلیل این امر این است که:

$d_{\text{vc}} \geq N \iff$ یک مجموعه $\mathcal{D}$ در اندازه N وجود دارد که $\mathcal{H}$ می‌تواند آن را خرد کند.

با توجه به این رابطه، موارد زیر قابل نتیجه‌گیری است.

۱. مجموعه‌ای از N نقطه وجود دارد که توسط $\mathcal{H}$ خرد می‌شود. در این مورد می‌توانیم نتیجه بگیریم $d_{\text{vc}} \geq N$ است.

۲. هر مجموعه از N نقطه توسط $\mathcal{H}$ خرد می‌شود. در این مورد اطلاعات بیشتری وجود

دارد که نتیجه بگیریم $d_{\text{vc}} \geq N$ است.

۳. یک مجموعه از N نقطه وجود دارد که توسط $\mathcal{H}$ خرد نمی‌شود. در این حالت، تنها بر اساس این اطلاعات چیزی در مورد مقدار d_{vc} را نمی‌توانیم نتیجه‌گیری کنیم.

۴. هیچ N نقطه‌ای نمی‌تواند توسط $\mathcal{H}$ خرد شود. در این حالت می‌توانیم نتیجه بگیریم $d_{\text{vc}} < N$ است.

همان‌طور که ملاحظه می‌شود اثبات اینکه d_{vc} حداقل برابر با N است، بسیار ساده‌تر از اثبات این است که d_{vc} حداکثر برابر با N است. برای این کار تنها کافی است مجموعه‌ای از N نقطه را پیدا کنیم که توسط $\mathcal{H}$ خرد می‌شوند.

بعد VC یک پرسپترون $\mathcal{D}$ بعدی برابر با $d+1$ است. طبق شکل ۲-۱ (ج) این مطلب در مورد $d = 2$ که در آن $d_{\text{vc}} = 3$ است صادق است. ما اثبات این مطلب را به خواننده واگذار می‌کنیم و تنها به ذکر نتایج آن می‌پردازیم. مورد پرسپترون بینش جدیدی را در مورد بعد VC فراهم می‌کند. ازآنجاکه $d+1$ برابر با تعداد پارامترهای این مدل است، می‌توان به بعد VC به‌عنوان مقیاسی از تعداد مؤثر پارامترها نگاه کرد. هراندازه تعداد پارامترهای یک مدل بیشتر باشد، مجموعه فرضیات آن نیز متنوع‌تر است، که این موضوع خود را در قالب مقدار بالاتر برای تابع رشد $m_{\mathcal{H}}(N)$ نشان می‌دهد. در مورد پرسپترون ها، تعداد پارامترهای مؤثر متناظر است با تعداد پارامترهایی که به شکل صریح در مدل مشخص شده‌اند و با $w_0, w_1 \ldots, w_d$ نمایش داده می‌شوند. در بقیه مدل‌ها پارامترهای مؤثر به این وضوح قابل‌تفکیک نیستند. بعد VC نشانگر تعداد مؤثر پارامترها یا "درجه آزادی" [۱] یک مدل است که مدل را قادر به نمایش مجموعه متنوعی از فرضیات می‌کند.

تنوع در حوزه تعمیم لزوماً چیز خوبی نیست. برای مثال اگر فرضیه‌ها بتوانند تا حد ممکن متنوع باشند، در آن صورت برای تمام Nها $m_{\mathcal{H}}(N) = 2^N$ و $d_{\text{vc}} = \infty$ خواهد شد، در این حالت نمی‌توان انتظار هیچ تعمیمی داشت.

[۱] degrees of freedom

۲-۲ تفسیر کران تعمیم

کران تعمیم VC (۲-۷) یک نتیجه عام است. به این معنی که به تمام مجموعه فرضیات، الگوریتم‌های یادگیری، فضاهای ورودی، توزیع‌های احتمال و توابع هدف دودویی قابل‌اعمال است. همین‌طور به انواع دیگر توابع هدف نیز به‌خوبی بسط پیدا می‌کند. با توجه به این عمومیت، ممکن است به نظر برسد که این کران برای یک کاربرد ویژه به اندازه کافی تنگ[۱] نباشد، زیرا باید ملاحظات مختلفی را پوشش دهد.

در حقیقت این کران بسیار سست[۲] است. این لقی در کران VC می‌تواند به عوامل فنی مختلفی مربوط شود. از آن جمله می‌توان به موارد زیر اشاره می‌شود:

۱. نامعادله هافدینگ که ما از آن به‌عنوان پایه اثبات کران VC استفاده کردیم، خود دارای لقی است. این نامعادله، فارغ از اینکه E_{out} به 0.5 نزدیک باشد یا به صفر، کران یکسانی را ایجاد می‌کند. هرچند واریانس E_{in} در این دو حالت کاملاً متفاوت است و داشتن کرانی که هر دو مورد را پوشش دهد، منجر به یک لقی می‌شود.

۲. استفاده از تابع رشد $m_{\mathcal{H}}(N)$ به‌عنوان حداکثر تعداد دوبخشی‌ها روی N نقطه، فارغ از اینکه مجموعه داده واقعی شامل کدام N نقطه است، یک تخمین بدترین حالت را فراهم می‌کند. این امر اجازه می‌دهد که کران حاصل مستقل از توزیع احتمال P روی $\mathcal{X}$ باشد. هرچند چنانچه ما یک مجموعه خاص $\mathbf{x}_1, \ldots, \mathbf{x}_N$ را در نظر بگیریم و از اندازه $|\mathcal{H}(\mathbf{x}_1, \ldots, \mathbf{x}_N)|$ یا مقدار مورد انتظارش به جای کران بالاتر $m_{\mathcal{H}}(N)$ استفاده کنیم، می‌توانیم کران مناسب‌تری را فراهم کنیم. برای مثال در مورد مجموعه‌های محدب[۳] در دو بعد، اگر N نقطه به شکل تصادفی در صفحه انتخاب شوند، با احتمال زیاد تعداد دوبخشی‌ها روی این نقاط بسیار کمتر از 2^N خواهد بود، درحالی‌که در این حالت به طور کلی $m_{\mathcal{H}}(N) = 2^N$ فرض می‌شود.

۳. کران‌دار کردن $m_{\mathcal{H}}(N)$ با یک چندجمله‌ای ساده‌شده از درجه d_{vc} که در ۲-۵ داده‌شده است، لقی دیگری را به کران VC اضافه می‌کند.

---

[۱] tight

[۲] loose

[۳] convex set

برخی تلاش‌ها برای تنگ‌تر کردن کران VC صورت گرفته است. اما این تلاش‌ها تنها نتایج مختصری را در پی داشته‌اند. حقیقت این است که مسیر تحلیلی VC منتهی به یک کران سست می‌شود. ممکن است این سؤال پیش آید که در این صورت چرا اصلاً ما باید از چنین تحلیلی استفاده کنیم. دو دلیل عمده وجود دارد. اول اینکه تحلیل VC ابزاری است که امکان‌پذیری یادگیری را برای مجموعه داده‌های نامتناهی فراهم می‌کند و تنها روشی است که ما در عمل استفاده می‌کنیم. دلیل دوم این است که هرچند این کران بسیار لق است، اما این لقی برای مدل‌های مختلف یادگیری به یک نسبت است، بنابراین این کران همچنان می‌تواند معیاری را برای مقایسه این مدل‌ها فراهم کند. البته این تنها یک مشاهده تجربی است و نه یک قضیه ریاضی. در کاربردهای واقعی مدل‌هایی که d_{vc} کوچک‌تری دارند نسبت به مدل‌هایی که d_{vc} بالاتری دارند، بهتر تعمیم پیدا می‌کنند. با توجه به این مشاهده، تحلیل VC در عمل می‌تواند مفید واقع شود و در این زمینه برخی قوانین سرانگشتی نیز برحسب بعد VC به وجود آمده‌اند. برای مثال به‌عنوان یک قانون سرانگشتی، برای اینکه به یک تعمیم خوب برسیم، اندازه نمونه N حداقل باید برابر با $10 \times d_{\text{vc}}$ باشد.

به این ترتیب کران VC می‌تواند به شکل نسبی و نه مطلق به عنوان یک راهنما برای مسئله تعمیم استفاده شود. با این نگرش به این کران اجازه دهید به نحوه استفاده از آن در عمل نگاهی بیندازیم.

۲-۲-۱ پیچیدگی نمونه

پیچیدگی نمونه به این معنی است که ما برای رسیدن به یک کارایی تعمیم مشخص، به چه تعداد مثال آموزشی نیاز داریم. این کارایی با دو پارامتر ϵ و δ نمایش داده می‌شود. میزان تحمل خطای ϵ بیانگر حد مجاز خطای تعمیم است و δ مشخص می‌کند که در چند درصد موارد این حد شکسته می‌شود.

می‌توان از کران VC برای تخمین پیچیدگی نمونه برای یک مدل یادگیری داده‌شده استفاده کرد. یک $\delta > 0$ را انتخاب کنید و فرض کنید قصد داریم خطای تعمیم حداکثر به اندازه ϵ باشد. طبق معادله (۲-۷)، خطای تعمیم توسط $\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$ کران‌دار می‌شود، در نتیجه کافی است $\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}} \leq \epsilon$ شود. با حل این نامعادله برحسب N به نامعادله زیر می‌رسیم:

$$N \geq \frac{8}{\epsilon_2} \ln \left(\frac{4m_{\mathcal{H}}(2N)}{\delta} \right)$$

حال اگر $m_{\mathcal{H}}(N)$ را با کران چندجمله‌ای آن برحسب بعد VC جایگزین کنیم به کران زیر می‌رسیم:

$$N \geq \frac{8}{\epsilon_2} \ln\left(\frac{4((2N)^{d_{\text{vc}}}+1)}{\delta}\right) \qquad (۸-۲)$$

این نامعادله یک کران ضمنی برای N است، به این معنی که N در دو طرف نامعادله قرار دارد و محاسبه آن به شکل صریح ممکن نیست. می‌توانیم از یک روش عددی برای بدست آوردن N استفاده کنیم.

برای مثال فرض کنید در یک مدل یادگیری $d_{\text{vc}} = 3$ است. ما قصد داریم با ضریب اطمینان ۹۰%، خطای تعمیم را حداکثر برابر با 0.1 قرار دهیم ($\epsilon = 0.1$ ،$\delta = 0.1$) اندازه مجموعه داده مورد نیاز چقدر باید باشد؟

با جایگذاری مقادیر داده‌شده در فرمول بالا خواهیم داشت:

$$N \geq \frac{8}{0.1_2} \ln\left(\frac{4((2N)^3+1)}{0.1}\right)$$

برای به دست آوردن N با یک مقدار اولیه N شروع می‌کنیم. با قرار دادن $N = 1000$ در سمت راست نامعادله خواهیم داشت:

$$N \geq \frac{8}{0.1_2} \ln\left(\frac{4((2\times 1000)^3+1)}{0.1}\right) \approx 21.193$$

اکنون مقدار جدید $N = 21193$ را دوباره در سمت راست نامعادله قرار می‌دهیم و در یک چرخه تکراری نهایتاً N به مقدار 30000 همگرا می‌شود. اگر ما این محاسبات را برای $d_{\text{vc}} = 4$ تکرار کنیم به $N = 40000$ خواهیم رسید و به همین ترتیب برای $d_{\text{vc}} = 5$ به $N = 50000$ می‌رسیم. ظاهراً این نامعادله پیشنهاد می‌کند که تعداد مثال‌های مورد نیاز متناسب با بعد VC به شکل خطی افزایش پیدا می‌کند و این ضریب تناسب حدوداً برابر با 10000 است. البته این تخمین، تخمین بسیار بالایی است و بر اساس مشاهدات عملی، ضریب این نسبت حدوداً برابر با 10 است.

۲-۲-۲ جریمه پیچیدگی مدل

نامعادله ارائه‌شده برای پیچیدگی نمونه با فرض اینکه مقادیر ϵ (خطای تعمیم) و δ داده شده‌اند، سعی می‌کند تعداد مثال‌های که برای رسیدن به این کارایی مورد نیاز است را تخمین

بزنند. اما در اکثر کاربردهای عملی، مجموعه داده از قبل داده‌شده است و درنتیجه مقدار N ثابت است. در این وضعیت، سؤال مرتبط این است که با توجه به N داده‌شده، کارایی تعمیم مورد انتظار مدل چقدر است؟ کران ۲-۷ پاسخی به این سؤال نیز فراهم می‌کند:

با احتمال حداقل $1-\delta$ داریم:

$$E_{out} \leq E_{in} + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

اگر $m_{\mathcal{H}}(2N)$ را با کران چندجمله‌ای برحسب d_{vc} جایگزین کنیم به کران زیر خواهیم رسید.

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \Big(\frac{4((2N)^{d_{\text{vc}}}+1)}{\delta}\Big)} \qquad (۹-۲)$$

برای مثال فرض کنید $N = 100$ و ضریب اطمینان موردنظر ۹۰% است ($\delta = 0.1$). از شما خواسته شده است که خطای تعمیم را با توجه به این ضریب اطمینان برای یک مدل یادگیری با $d_{\text{vc}} = 1$ به دست آورید. با استفاده از کران بالا خواهیم داشت:

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\tfrac{8}{100} \ln \Big(\tfrac{4(201)}{0.1}\Big)} \approx E_{in}(g) + 0.848$$

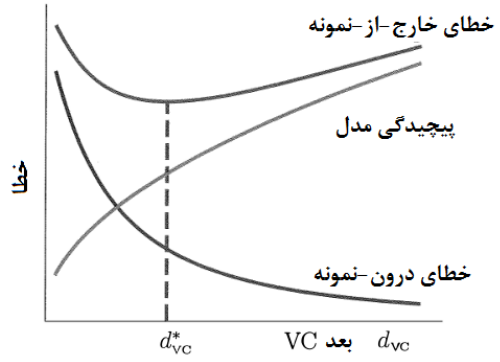
برای ضریب اطمینان ۹۰%، این نتیجه کران بسیار ضعیفی برای E_{out} است. حتی اگر $E_{in} = 0$ باشد، E_{out} همچنان ممکن است به 1 نزدیک باشد. اگر $N = 1000$ باشد، در آن صورت ما خواهیم داشت $E_{out}(g) \leq E_{in}(g) + 0.301$ که کران نسبتاً بهتری است.

اجازه دهید نگاه نزدیک‌تری به دو بخش اصلی که کران تعمیم را تشکیل می‌دهند بیندازیم. بخش اول E_{in} است و بخش دوم ترمی است که با افزایش بعد VC مجموعه فرضیات $\mathcal{H}$ افزایش می‌یابد:

$$E_{out}(g) \leq E_{in}(g) + \Omega(N, \mathcal{H}, \delta), \qquad (۱۰-۲)$$

که در آن

$$\Omega(N, \mathcal{H}, \delta) = \sqrt{\tfrac{8}{N} \ln \tfrac{4m_{\mathcal{H}}(2N)}{\delta}} \leq \sqrt{\tfrac{8}{N} \ln \Big(\tfrac{4((2N)^{d_{\text{vc}}}+1)}{\delta}\Big)}$$



شکل ۲-۲: زمانی که از یک مدل یادگیری پیچیده‌تر استفاده می‌کنیم (مدلی که دارای بعد VC بالاتری است)، بااحتمال‌زیاد قادر خواهیم بود داده‌ها را بهتر برازش کنیم و به خطای درون-نمونه پایین‌تری دست پیدا کنیم، اما بهای پیچیدگی مدل را باید در قالب ترم جریمه بالاتر بپردازیم. بنابراین ترکیبی از این دو که خطای خارج-از-نمونه را تخمین می‌زنند، می‌تواند در یک بعد VC میانی d^*_{vc} به یک مقدار بهینه برسد.

می‌توان $\Omega(N, \mathcal{H}, \delta)$ را به‌عنوان جریمه پیچیدگی مدل در نظر گرفت. هرچه از $\mathcal{H}$ پیچیده‌تری استفاده کنیم (d_{vc} بزرگ‌تر)، این مقدار جریمه بزرگتر شده و کران E_{out} را ضعیف‌تر می‌کند. اگر شخصی برای همان مجموعه آموزشی از مدل ساده‌تری استفاده کند، به تخمین مطلوب‌تری از E_{out} دست پیدا خواهد کرد. همان‌طور که انتظار می‌رود، اصرار بر وجود ضریب اطمینان بالا (δ کمتر) مقدار جریمه $\Omega(N, \mathcal{H}, \delta)$ را افزایش خواهد داد. درمقابل با افزایش تعداد مثال‌های مجموعه آموزشی (N بزرگتر) مقدار جریمه کمتر خواهد شد.

هرچند ترم جریمه $\Omega(N, \mathcal{H}, \delta)$ زمانی که $\mathcal{H}$ بعد VC بالایی دارد افزایش می‌یابد، اما احتمالاً E_{in} به علت بعد بالاتر VC پایین می‌آید، زیرا ما انتخاب‌های بیشتری در $\mathcal{H}$ برای برازش بهتر داده‌ها داریم. بنابراین، باید در اینجا موازنه‌ای صورت گیرد. مدل‌های پیچیده‌تر به E_{in} کمک کرده و به ترم جریمه $\Omega(N, \mathcal{H}, \delta)$ ضربه می‌زنند. مدل بهینه، مدلی است که ترکیب این دو ترم را کاهش دهد. این موضوع به شکل غیررسمی در شکل ۲-۲ نمایش داده‌شده است.

۲-۲-۳ مجموعه آزمایشی

همان‌طور که دیدیم، کران تعمیم یک تخمین سُست از E_{out} را بر اساس E_{in} فراهم می‌کند. با اینکه در فرایند آموزش این تخمین می‌تواند به‌عنوان یک راهنمای مفید مورد استفاده قرار گیرد، اما به‌عنوان یک پیش‌بینی از E_{out}، تخمین ناکارآمدی است. اگر شما سامانه‌ای را برای یک مشتری توسعه دهید، برای اینکه مشتری بداند کارایی مورد انتظار سامانه چقدر است به

تخمین دقیق‌تری نیاز دارید.

یک روش جایگزین برای دست‌یابی به تخمین مناسبی از E_{out} استفاده از مجموعه داده آزمایشی است. همانطور که در ابتدای فصل اشاره شد، این مجموعه در فرایند آموزش استفاده نمی‌شود. پس از اتمام یادگیری، فرضیه انتخاب‌شده g روی مجموعه آزمایشی ارزیابی می‌شود و نتیجه به‌عنوان تخمینی از E_{out} گزارش می‌شود. در اینجا قصد داریم نگاه دقیق‌تری به این رویکرد داشته باشیم.

اجازه دهید خطا روی مجموعه آزمایشی را با E_{test} نمایش دهیم. زمانی که ما E_{test} را به‌عنوان تخمینی از E_{out} گزارش می‌کنیم، درواقع ادعا می‌کنیم که E_{test} به‌خوبی به E_{out} تعمیم پیدا می‌کند. اما E_{test} نیز همانند E_{in} خطای موجود در یک نمونه مشخص است، پس چگونه می‌توان ادعا کرد که E_{test} به‌خوبی تعمیم پیدا می‌کند؟ این سؤال را می‌توان با همان نظریه‌ای که برای خطای تعمیم ارائه شد، پاسخ داد.

تعداد مؤثر فرضیات در مورد E_{test} برابر با یک است. تا زمانی که مجموعه آزمایشی مطرح باشد، تنها یک فرضیه وجود دارد و آن فرضیه نیز همان فرضیه نهایی g است که از مرحله آموزش به‌دست‌آمده است. این فرضیه به مجموعه آموزشی وابسته است، اما در صورت تغییر مجموعه آزمایشی همچنان ثابت باقی می‌ماند. به این معنی که این فرضیه قبل از ایجاد مجموعه آزمایشی تثبیت شده است. درنتیجه نامعادله هافدینگ را می‌توان برای مجموعه آزمایشی بکار برد. بنابراین کران تعمیمی که روی مجموعه آزمایشی اعمال می‌شود همان نامعادله هافدینگ است و این کران نسبت به کران VC کران تنگ‌تری است. برای مثال اگر مجموعه داده آزمایشی شامل 1000 نقطه داده باشد، با احتمال بزرگتر از ۹۸ %، E_{test} حدوداً میان ± 5 باقی خواهد ماند. هر چه این مجموعه آزمایشی بزرگ‌تر باشد E_{test} تخمین بهتری از E_{out} فراهم خواهد کرد.

جنبه دیگری که مجموعه آزمایشی را از مجموعه آموزشی تفکیک می‌کند این است که مجموعه آزمایشی دارای بایاس نیست (بی‌طرف است). هر دو مجموعه آزمایشی و آموزشی مجموعه‌های محدودی هستند و به علت اندازه محدود نمونه دارای واریانس هستند، بااین‌حال مجموعه آزمایشی هیچ بایاس بدبینانه یا خوش‌بینانه‌ای در تخمین E_{out} ندارد.

مجموعه داده آموزشی دارای یک بایاس خوش‌بینانه است زیرا فرضیه‌ای را انتخاب می‌کند که روی آن مجموعه داده به‌خوبی جواب داده است. کران تعمیم VC این بایاس را در نظر

می‌گیرید و به همین علت کران بالایی را ارائه می‌کند. اما مجموعه داده آزمایشی تنها دارای واریانس مجموعه-متناهی[۱] است، اما بایاسی ندارد. زمانی که شما مقدار E_{test} را به مشتری گزارش می‌دهید و مشتری سیستم را روی داده‌های جدید به کار می‌برد، عملکرد سیستم به پیش‌بینی نزدیک است و احتمال کمی وجود دارد که این عملکرد بسیار بهتر و یا بسیار بدتر از پیش‌بینی شما باشد.

البته برای داشتن مجموعه آزمایشی بهایی نیز باید پرداخته شود. مجموعه آزمایشی در فرایند یادگیری دخالتی ندارد و تنها مشخص می‌کند که یادگیری به چه صورت انجام شده است. بنابراین چنانچه شما مجموعه محدودی از مثال‌ها را در اختیار دارید و بخشی از آنها را به مجموعه آزمایشی اختصاص دهید، مثال‌های کمتری برای مرحله آموزش باقی خواهند ماند. ازآنجاکه مجموعه آموزشی برای انتخاب یکی از فرضیات مجموعه $\mathcal{H}$ استفاده می‌شود، مثال‌های بیشتر برای یافتن یک فرضیه خوب حیاتی هستند. اگر قسمت اعظمی از داده‌ها را برای آزمایش اختصاص دهیم، مثال‌های کمی برای آموزش در اختیار خواهیم داشت و درنتیجه ممکن است به یک فرضیه خوب نرسیم. تنها مزیتی که وجود دارد این است که می‌توانیم با اعتماد بالایی کارایی فرضیه به‌دست‌آمده را روی داده‌های آزمایشی ارزیابی کنیم. در این حالت، ممکن است نهایتاً به مشتری گزارش دهیم که با اطمینان زیاد فرضیه به‌دست‌آمده وحشتناک است!

بنابراین باید همواره موازنه‌ای میان مجموعه آزمایشی و مجموعه آموزشی برقرار سازیم. این موازنه نیاز به جزئیات بیشتری دارد که ما در قسمت‌های بعد به آنها خواهیم پرداخت.

۲-۲-۴ توابع هدف دیگر

هرچند تحلیل VC بر اساس توابع دودویی بنا شد، اما این تحلیل می‌تواند به توابع حقیقی و دیگر انواع توابع نیز بسط پیدا کند. اما اثبات این موارد خیلی تخصصی است و از طرفی بینش بیشتری را نیز فراهم نمی‌کند. به همین دلیل قصد داریم رویکرد جایگزینی را برای توابع حقیقی معرفی کنیم که بینش جدیدی را نسبت به مسئله تعمیم به ما نمی‌دهد. این رویکرد بر اساس تحلیل بایاس و واریانس است.

برای کار با توابع حقیقی لازم است تعریفی که از E_{in} و E_{out} برای توابع دودویی داشتیم را تغییر دهیم. ما قبلاً E_{in} و E_{out} را برحسب خطای دودویی تعریف کردیم. در این تعریف

[۱] finite-set variance

یا $h(\mathsf{x}) = f(\mathsf{x})$ است و یا $h(\mathsf{x}) \neq f(\mathsf{x})$ است که در مورد دوم ما شاهد خطا هستیم. اما اگر f و h توابع حقیقی باشند، به جای اینکه در هر نقطه دقیقاً بررسی کنیم آیا مقدارشان یکی است یا خیر، به مقیاس خطای دیگری نیاز داریم که اختلاف $f(\mathsf{x})$ و $h(\mathsf{x})$ را اندازه‌گیری کند.

مقیاس خطایی که معمولاً به این منظور استفاده می‌شود مربع خطا است:

$$\mathsf{e}(f(\mathsf{x}), h(\mathsf{x})) = (h(\mathsf{x}) - f(\mathsf{x}))^2$$

می‌توان خطای درون-نمونه و خارج-از-نمونه را بر اساس این مقیاس جدید تعریف کرد. خطای خارج از نمونه $E_{out}(h)$ براساس مقدار مورد انتظار خطا (امید ریاضی) روی کل فضای ورودی تعریف می‌شود:

$$E_{out}(h) = \mathbb{E}[(h(\mathsf{x}) - f(\mathsf{x}))^2]$$

خطای درون-نمونه بر اساس میانگین خطا روی نقاط داخل مجموعه قرار دارد و می‌توان این مقدار را محاسبه کرد:

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} (h(\mathsf{x}) - f(\mathsf{x}))^2$$

در این حالت نیز مانند حالت دودویی E_{in} تخمینی از E_{out} را فراهم می‌کند. در حقیقت خطای دودویی را نیز می‌توان بر اساس مربع خطا تحلیل کرد. نامعادله هافدینگ در اینجا نیز برقرار است و متوسط مربعات خطای نقاط داخل نمونه به مقدار مورد انتظار خطا در خارج از نمونه همگرا می‌شود. در این تحلیل نیز همانند تحلیل VC پیامدهای ناشی از اندازه مجموعه داده و پیچیدگی مجموعه فرضیات نقش مهمی بازی می‌کنند.

۲-۳ موازنه تقریب-تعمیم

تحلیل VC نشان داد که انتخاب $\mathcal{H}$ نیاز به ایجاد موازنه‌ای میان تقریب f روی مجموعه داده‌های آموزشی و تعمیم آن روی داده‌های جدید دارد. به شکل ایدئال $\mathcal{H}$ باید تنها شامل یک فرضیه که همان تابع هدف است باشد، اما بهتر است یک بلیت بخت‌آزمایی بخرید تا اینکه امیدوار باشید چنین $\mathcal{H}$ی را در اختیار خواهید داشت.

ازآنجاکه تابع هدف برای ما ناشناخته است، از یک مدل بزرگ استفاده می‌کنیم به این امید که یک مدل بزرگتر حاوی فرضیه خوبی است و داده‌های موجود آن فرضیه را پیدا خواهند کرد. زمانی که مجموعه فرضیات را انتخاب می‌کنیم باید موازنه‌ای را میان این دو هدف متناقض برقرار سازیم: یعنی از یک طرف داشتن فرضیاتی در مجموعه $\mathcal{H}$ که بتوانند f را به خوبی تقریب بزنند و از طرف دیگر توانایی سوق دادن داده‌ها به سمت انتخاب فرضیه درست.

یک راه برای نگاه به این موازنه کران تعمیم VC است. اگر $\mathcal{H}$ زیادی ساده باشد ممکن است نتوانیم در آن تقریب خوبی برای f پیدا کنیم و به خطای درون-نمونه بالایی برسیم. اگر $\mathcal{H}$ زیادی پیچیده باشد، در آن صورت ممکن است نتوانیم فرضیه به‌دست‌آمده را تعمیم دهیم زیرا مقدار ترم پیچیدگی مدل در کران تعمیم بالا می‌رود.

راه دیگری نیز برای نگاه به موازنه تقریب و تعمیم وجود دارد که در قسمت بعد توضیح داده خواهد شد. به طور مشخص این روش به جای خطای دودویی که در تحلیل VC استفاده شد بر اساس مقیاس مربع خطا قرار دارد. در این روش جدید به جای اینکه E_{out} را با E_{in} بعلاوه ترم جریمه Ω کران‌دار کنیم، آن را به دو ترم خطای دیگر تحت عنوان بایاس و واریانس تجزیه می‌کنیم.

۲-۳-۱ تحلیل بایاس-واریانس

تجزیه خطای خارج-از-نمونه به دو ترم بایاس و واریانس بر اساس مقیاس مربع خطا قرار دارد. خطای خارج-از-نمونه برابر است با:

$$E_{out}(g^{(\mathcal{D})}) = \mathbb{E}_{\mathsf{x}}\Big[(g^{(\mathcal{D})}(\mathsf{x}) - f(\mathsf{x}))^2\Big], \qquad (۲-۱۱)$$

که در آن $\mathbb{E}_{\mathsf{x}}$ نشان‌دهنده مقدار مورد انتظار برحسب ورودی x است (بر اساس توزیع احتمال روی فضای ورودی $\mathcal{X}$). در این معادله وابستگی فرضیه نهایی g به مجموعه داده آموزشی $\mathcal{D}$ را به شکل صریح نشان داده‌ایم و این امر نقشی کلیدی در تحلیل پیش رو بازی خواهد کرد.

برای اینکه از وابستگی به یک مجموعه داده خاص رها شویم، می‌توان مقدار مورد انتظار را برحسب تمام مجموعه داده‌ها به دست آورد. در آن صورت خطای مورد انتظار خارج-از-

نمونه برای یک مدل یادگیری را می‌توان مستقل از هر تحقق خاص مجموعه داده به دست آورد. درنتیجه خواهیم داشت:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\Big[E_{out}(g^{(\mathcal{D})})\Big] &= \mathbb{E}_{\mathcal{D}}\Big[\mathbb{E}_{\mathbf{x}}[(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}))^2]\Big] \\ &= \mathbb{E}_{\mathbf{x}}\Big[\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x}))^2]\Big] \\ &= \mathbb{E}_{\mathbf{x}}\Big[\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})^2] - 2\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})]f(\mathbf{x}) + f(\mathbf{x})^2\Big].\end{aligned}$$

ترم $\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})]$ یک "تابع متوسط" را می‌دهد که ما آن را با $\bar{g}(\mathbf{x})$ نمایش می‌دهیم و به شکل زیر تفسیر می‌شود . مجموعه داده‌های زیادی به شکل $\mathcal{D}_1, \ldots, \mathcal{D}_K$ را تولید کنید و الگوریتم یادگیری را روی هر یک از آنها اعمال کنید تا به فرضیات نهایی $g_1, \ldots, g_K$ برسید. اکنون برای هر $\mathbf{x}$ می‌توان تخمینی از تابع متوسط را به شکل زیر به دست آورد:

$$\bar{g}(\mathbf{x}) \approx \tfrac{1}{K}\sum_{k=1}^{K} g_k(\mathbf{x})$$

به طور اساسی ما تابع $g(\mathbf{x})$ را به عنوان یک متغیر تصادفی در نظر می‌گیریم که تصادفی بودن آن از تصادفی بودن مجموعه داده ناشی می‌شود. $\bar{g}(\mathbf{x})$ مقدار مورد انتظار این متغیر تصادفی برای یک $\mathbf{x}$ مشخص است و $\bar{g}$ تابعی است (تابع متوسط) که از این مقادیر مورد انتظار تشکیل شده است. دقت کنید بااینکه $\bar{g}$ متوسط توابعی است که همگی در مجموعه فرضیات مدل وجود دارند، خود تابع $\bar{g}$ لزوماً در مجموعه فرضیات وجود ندارد.

ما اکنون خطای خارج-از-نمونه مورد انتظار را برحسب $\bar{g}$ بازنویسی می‌کنیم:

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}\Big[E_{out}(g^{(\mathcal{D})})\Big] \\ &= \mathbb{E}_{\mathbf{x}}\Big[\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})^2] - 2\bar{g}(\mathbf{x})f(\mathbf{x}) + f(\mathbf{x})^2\Big] \\ &= \mathbb{E}_{\mathbf{x}}\Big[\underbrace{\mathbb{E}_{\mathcal{D}}[g^{(\mathcal{D})}(\mathbf{x})^2] - \bar{g}(\mathbf{x})^2}_{\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}))^2]} + \underbrace{\bar{g}(\mathbf{x})^2 - 2\bar{g}(\mathbf{x})f(\mathbf{x}) + f(\mathbf{x})^2}_{(\bar{g}(\mathbf{x}) - f(\mathbf{x}))^2}\Big]\end{aligned}$$

که در آن ثابت بودن $\bar{g}(\mathsf{x})$ نسبت به مجموعه داده $\mathcal{D}$ به ما امکان کاهش جملات به دو بخش آخر را فراهم می‌کند. ترم $(\bar{g}(\mathsf{x}) - f(\mathsf{x}))^2$ نشان می‌دهد به چه میزان تابع متوسطی که از طریق مجموعه داده‌های مختلف $\mathcal{D}$ یادگیری شده است از تابع هدفی که این مجموعه داده‌ها را تولید کرده است منحرف شده است. این ترم "بایاس[۱]" خوانده می‌شود:

$$\mathsf{bias}(\mathsf{x}) = (\bar{g}(\mathsf{x}) - f(\mathsf{x}))^2$$

بایاس درواقع نشان می‌دهد که چقدر مدل یادگیری از تابع هدف فاصله گرفته است. دلیل این مطلب این است که $\bar{g}$ از روی تعداد نامتناهی مجموعه داده یادگیری می‌شود، درنتیجه تنها چیزی که توانایی این تابع در تقریب زدن f را محدود می‌کند، محدودیتی است که از خود مدل یادگیری ناشی می‌شود.

ترم $\mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(\mathsf{x}) - \bar{g}(\mathsf{x}))^2]$ واریانس متغیر تصادفی $g^{(\mathcal{D})}(\mathsf{x})$ نامیده می‌شود:

$$\mathsf{var}(\mathsf{x}) = \mathbb{E}_{\mathcal{D}}[(g^{(\mathcal{D})}(\mathsf{x}) - \bar{g}(\mathsf{x}))^2]$$

که میزان تنوع فرضیه نهایی برحسب مجموعه داده را اندازه می‌گیرد. با این توضیحات به تجزیه بایاس و واریانس برای خطای خارج-از-نمونه خواهیم رسید :

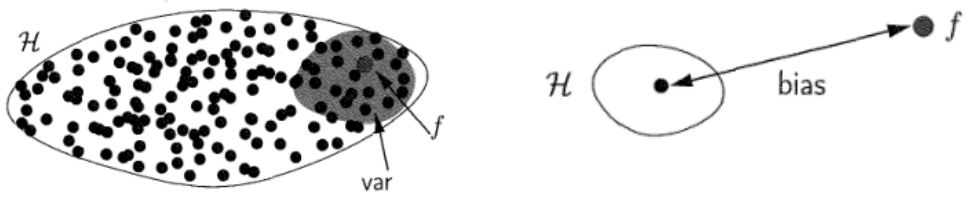
$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] &= \mathbb{E}_{\mathsf{x}}[\mathsf{bias}(\mathsf{x}) + \mathsf{var}(\mathsf{x})] \\ &= \mathsf{bias} + \mathsf{var}\end{aligned}$$

که در آن بایاس به شکل $\mathsf{bias} = \mathbb{E}_{\mathsf{x}}[\mathsf{bias}(\mathsf{x})]$ و واریانس به شکل $\mathsf{var} = \mathbb{E}_{\mathsf{x}}[\mathsf{var}(\mathsf{x})]$ است. فرض این استخراج این است که داده‌ها بدون نویز هستند. در شرایط نویزی، یک ترم نویز هم به این ترکیب اضافه خواهد شد. ترم نویز غیرقابل‌اجتناب است و نمی‌توانیم از آن جلوگیری کنیم. ترم‌هایی که بیشتر در اختیار ما هستند ترم بایاس و واریانس هستند و ما در ادامه به تشریح آنها می‌پردازیم.

موازنه تقریب-تعمیم که هم‌اینک بحث شد توسط تجزیه بایاس-واریانس نیز پوشش داده می‌شود. برای نمایش این مطلب اجازه دهید دو مورد خیلی افراطی را بررسی کنیم: مورد اول

[۱] bias

زمانی است که یک مدل کوچک تنها شامل یک فرضیه داریم و مورد دوم زمانی است که یک مدل خیلی بزرگ شامل تمامی فرضیات ممکن را داریم. این دو مورد در شکل زیر نشان داده شده‌اند.



(ب) مدل خیلی بزرگ (الف) مدل کوچک

ازآنجاکه مدل کوچک تنها شامل یک فرضیه است، هر دو تابع $\bar{g}$ متوسط و $g^{(\mathcal{D})}$ نهایی برای تمام مجموعه داده‌ها یکسان خواهند بود. درنتیجه واریانس صفر خواهد شد و بایاس تنها به توانایی این فرضیه یکتا در تقریب تابع هدف وابسته است. در این حالت مگر اینکه خیلی خوش‌شانس باشیم، باید انتظار بایاس بالایی را داشته باشیم.

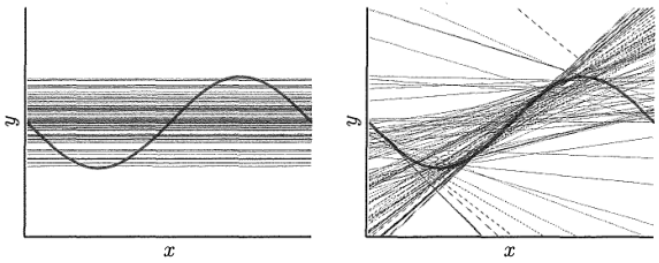
در مدل خیلی بزرگ، تابع هدف داخل مجموعه $\mathcal{H}$ است. مجموعه داده‌های متفاوت منتهی به فرضیات متفاوتی می‌شوند که هر یک روی آن مجموعه داده با f همخوانی دارند و در حوالی f پراکنده هستند (قسمت تیره‌رنگ در شکل). بنابراین بایاس تقریباً صفر است، زیرا با احتمال بالا $\bar{g}(\mathbf{x})$ متوسط نزدیک به f است؛ اما در این حالت مقدار واریانس بالا است (تقریباً معادل اندازه شعاع ناحیه تیره‌رنگ در شکل). می‌توان به واریانس به‌عنوان مقیاسی از میزان بی‌ثباتی در مدل یادگیری نگاه کرد. این بی‌ثباتی خود را به شکل واکنش تند به تغییرات کوچک یا غیرمعمول در داده‌ها نشان می‌دهد و منتهی به فرضیات بسیار متفاوتی خواهد شد.

مثال ۳-۲: تابع هدف $f(\mathbf{x}) = \sin(\pi x)$ و یک مجموعه داده با اندازه $N = 2$ را در نظر بگیرید. برای ایجاد مجموعه داده، به‌طور یکنواخت از x در بازه $[-1, 1]$ نمونه‌برداری می‌کنیم تا نقاط $(x_1, y_1), (x_2, y_2)$ حاصل شوند. این داده‌ها را با استفاده از دو مدل برازش می‌کنیم:

$\mathcal{H}_0$: مجموعه تمام خطها به شکل $h(x) = b$

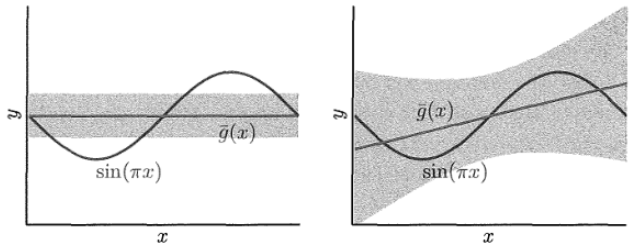
$\mathcal{H}_1$: مجموعه تمام خطها به شکل $h(x) = ax + b$

برای $\mathcal{H}_0$، خط افقی که از نقطه وسط این دو نقطه با عرض $b = \frac{(y_1+y_2)}{2}$ عبور می‌کند را به‌عنوان بهترین برازش انتخاب می‌کنیم. برای $\mathcal{H}_1$، ، این بهترین برازش توسط خطی حاصل می‌شود که از این دو نقطه (x_1, y_1) و (x_2, y_2) عبور می‌کند. اگر این روند را با تعداد زیادی مجموعه داده تکرار کنیم، ما می‌توانیم بایاس و واریانس را تخمین بزنیم. شکل‌های زیر برازش‌های به‌دست‌آمده روی مجموعه‌های تصادفی یکسان را برای هر دو مدل نشان می‌دهند.



$\mathcal{H}_0$ $\mathcal{H}_1$

در مورد $\mathcal{H}_1$، فرضیه یادگیری‌شده تندتر است و بسته به تغییر مجموعه داده تغییرات شدیدی دارد. تجزیه بایاس-واریانس این دو مدل در شکل زیر خلاصه شده است:



$\mathcal{H}_0$
bias $= 0.50$;
var $= 0.25$.

$\mathcal{H}_1$
bias $= 0.21$;
var $= 1.69$.

شکل ۲-۳: فرضیه متوسط $\bar{g}$ به شکل خط ممتد و $\text{var}(x)$ با نواحی تیره رنگ با اندازه $\bar{g}(\mathsf{x}) \pm \sqrt{\text{var}(x)}$ نشان داده شده است.

برای $\mathcal{H}_1$ فرضیه متوسط $\bar{g}$ (خط پررنگ)، یک برازش معقول با یک بایاس کم برابر با 0.21

است. اما تغییرات زیاد فرضیه‌ها منجر به مقدار بالای 1.69 برای واریانس شده است و این مقدار باعث افزایش خطای خارج-از-نمونه مورد انتظار به عدد 1.90 (جمع بایاس و واریانس) گردیده است. با مدل ساده‌تر $\mathcal{H}_0$ برازش‌ها دارای تغییرات کمتری هستند و ما واریانس به‌مراتب کمتری به میزان 0.25 را خواهیم داشت که با ناحیه سایه‌دار مشخص شده است. در مقابل در این حالت، برازش متوسط برابر با تابع ثابت صفر است که منجر به مقدار بالای 0.5 برای بایاس شده است. با جمع این دو مقدار خطای خارج-از-نمونه به میزان 0.75 حاصل می‌شود که نسبت به مدل قبلی مقدار بسیار کمتری است.

مدل ساده‌تر با کاهش قابل‌توجه واریانس به قیمت افزایش کمی در بایاس برنده این رقابت است. دقت کنید که در اینجا هدف ما مقایسه اینکه کدام‌یک از این فرضیات متوسط تقریب بهتری را برای تابع سینوس فراهم می‌کند نیست. در حقیقت این منحنی‌ها تنها ابزارهایی مفهومی هستند، زیرا در یادگیری واقعی ما به این تعداد مجموعه داده که بتوانند چنین منحنی‌هایی تولید کنند دسترسی نداریم. ما تنها یک مجموعه داده در اختیار داریم و این تحلیل نشان می‌دهد که با استفاده از یک مجموعه داده، مدل ساده‌تر به‌طور متوسط دارای خطای خارج-از-نمونه کمتری است. البته اندازه کم مجموعه داده نیز در این قضیه دخیل است و با افزایش N، ترم واریانس کاهش پیدا می‌کند و چنانچه از مجموعه داده خیلی بزرگ‌تری استفاده شود، بایاس بخش غالب در معادله E_{out} خواهد بود و $\mathcal{H}_1$ برنده رقابت خواهد شد.

الگوریتم یادگیری در تحلیل بایاس-واریانس نقشی را ایفاء می‌کند که در تحلیل VC ایفاء نمی‌کرد. در این رابطه دو نکته قابل‌توجه است:

۱. اساساً تحلیل VC کاملاً بر محور مجموعه فرضیات $\mathcal{H}$ قرار دارد و الگوریتم یادگیری $\mathcal{A}$ در آن نقشی ندارد. اما در مقابل، در تحلیل بایاس-واریانس هم $\mathcal{H}$ و هم الگوریتم $\mathcal{A}$ هر دو مهم هستند. با یک $\mathcal{H}$ مشخص، استفاده از الگوریتم‌های یادگیری مختلف منتهی به تولید $g^{(\mathcal{D})}$ متفاوتی خواهد شد. ازآنجاکه $g^{(\mathcal{D})}$ اساس تحلیل بایاس-واریانس را تشکیل می‌دهد، این امر می‌تواند منتهی به ترم‌های بایاس و واریانس متفاوتی شود.

۲. هرچند تحلیل بایاس-واریانس بر اساس مقیاس مربع خطا بنا شده است، اما لزومی ندارد الگوریتم یادگیری بر اساس کمینه کردن این تابع خطا باشد. این الگوریتم می‌تواند از هر مقیاس خطایی برای تولید $g^{(\mathcal{D})}$ بر اساس $\mathcal{D}$ استفاده کند. هرچند زمانی که $g^{(\mathcal{D})}$ تولید شد، ما بایاس و واریانس را با استفاده از مقیاس مربع خطا محاسبه می‌کنیم.

متأسفانه بایاس و واریانس در عمل قابل‌محاسبه نیستند، زیرا هر دو به تابع هدف و توزیع احتمال ورودی وابسته هستند که هر دوی آنها ناشناخته‌اند. درنتیجه تجزیه بایاس-واریانس تنها ابزاری تحلیلی است و معمولاً زمانی استفاده می‌شود که قصد داریم مدل جدیدی را ایجاد کنیم. زمانی که بایاس و واریانس را مورد ملاحظه قرار می‌دهیم، دو هدف نوعی را می‌توان دنبال کرد. یک هدف این است که سعی کنیم واریانس را بدون افزایش قابل‌توجه بایاس کاهش دهیم. از سوی دیگر می‌توانیم بایاس را بدون افزایش قابل‌توجه واریانس کاهش دهیم. این اهداف به‌وسیله تکنیک‌های متفاوتی قابل‌دستیابی هستند. برخی از این تکنیک‌ها مبتنی بر اصول نظری و برخی نیز ابتکاری هستند. "منظم‌سازی" [۱] یکی از این تکنیک‌ها است که ما در فصل بعد در مورد آن به‌تفصیل بحث خواهیم کرد. کاهش بایاس بدون افزایش واریانس نیاز به برخی اطلاعات قبلی در مورد تابع هدف به‌منظور انتخاب h در راستای f دارد و این عمل بسیار وابسته به کاربرد است؛ اما کاهش واریانس بدون مصالحه بیش‌ازحد بایاس می‌تواند از طریق یک سری تکنیک‌های عمومی انجام پذیرد.

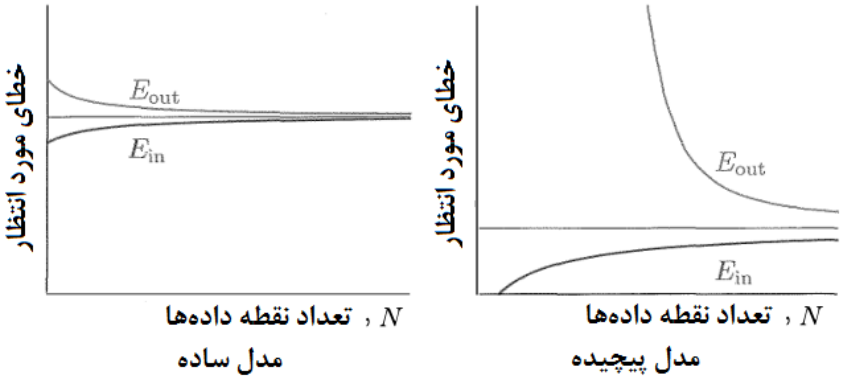
۲-۳-۲ منحنی یادگیری

این فصل را با ارائه یک نمودار مفهومی مهم خاتمه می‌دهیم. این نمودار که "منحنی یادگیری [۲]" خوانده می‌شود موازنه‌ای که ما تاکنون در مورد آن بحث کردیم را به تصویر می‌کشد. منحنی یادگیری رفتار خطاهای درون-نمونه و خارج-از-نمونه را نسبت به تغییر اندازه مجموعه داده آموزشی نشان می‌دهد.

بعد از یادگیری روی یک مجموعه داده خاص $\mathcal{D}$ با اندازه N، نظریه نهایی $g^{(\mathcal{D})}$ دارای خطای درون-نمونه $E_{in}(g^{(\mathcal{D})})$ و خطای خارج-از-نمونه $E_{out}(g^{(\mathcal{D})})$ خواهد بود که هر دوی آنها به مجموعه داده $\mathcal{D}$ وابسته هستند. همان‌طور که در تحلیل بایاس-واریانس دیدیم، مقادیر مورد انتظار خطا با در نظر گرفتن تمام مجموعه داده‌های با اندازه N، به شکل $\mathbb{E}_{\mathcal{D}}[E_{in}(g^{(\mathcal{D})})]$ و $\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})]$ حاصل می‌شوند. این خطاهای مورد انتظار توابعی از N هستند و منحنی‌های یادگیری مدل خوانده می‌شوند. در شکل بالا منحنی‌های یادگیری یک مدل ساده و یک مدل پیچیده یادگیری بر اساس آزمایش‌های واقعی آورده شده‌اند.

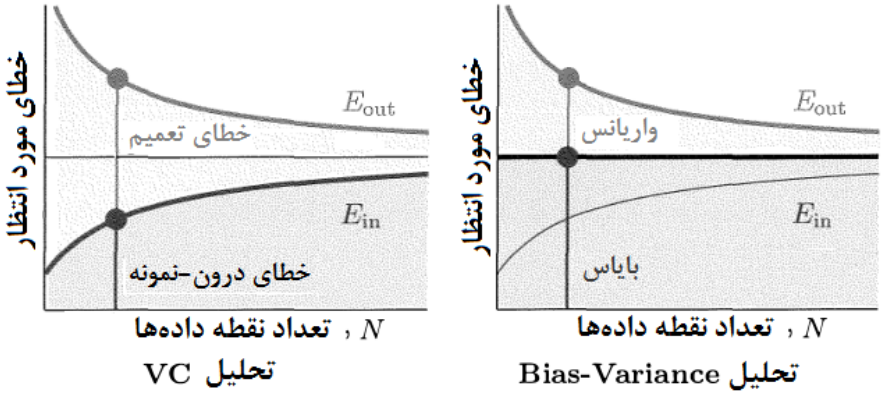
توجه کنید برای مدل ساده، منحنی‌های یادگیری سریع‌تر همگرا می‌شوند، اما کارایی نهایی

[۱] Regularization
[۲] Learning curve



بدتر از مدل‌های پیچیده است. این رفتار در عمل بسیار معمول است. برای هر دو مدل‌های ساده و پیچیده با افزایش N منحنی یادگیری خارج-از-نمونه کاهش پیدا می‌کند، اما در سوی دیگر منحنی یادگیری داخل-نمونه افزایش پیدا می‌کند.

اجازه دهید نگاه دقیق‌تری به این منحنی‌ها بیاندازیم و آنها را در قالب رویکردهای متفاوتی که به تعمیم داشتیم تفسیر کنیم. در تحلیل VC، E_{out} به صورت مجموع E_{in} و خطای تعمیم بیان شد که در آن خطای تعمیم توسط ترم Ω به‌عنوان جریمه پیچیدگی مدل کران‌دار می‌شود. در تحلیل بایاس-واریانس، E_{out} در قالب مجموع دو ترم بایاس و واریانس بیان شد. منحنی‌های یادگیری زیر این دو رویکرد را در کنار یکدیگر نشان می‌دهند.



تحلیل VC در سمت چپ نشان داده‌شده است. کرانی که این تحلیل برای خطای تعمیم ارائه می‌کند به شکل اختلاف میان E_{in} و E_{out} نشان داده‌شده است. تحلیل بایاس-واریانس در سمت راست قرار دارد. این نمودار تا حدی به شکل ایدئال رسم شده است، زیرا برای هر N، فرض می‌کند که کارایی فرضیه متوسط یادگیری $\bar{g}$ به اندازه کارایی بهترین تقریب f در مدل یادگیری است (خط وسط).

زمانی که تعداد نقطه‌داده‌ها افزایش می‌یابد ما در جهت راست منحنی یادگیری حرکت می‌کنیم و همان‌طور که انتظار داریم هر دو خطای تعمیم و ترم واریانس کاهش پیدا می‌کنند. منحنی یادگیری نکته مهم دیگری را در مورد E_{in} نشان می‌دهد. هر چه N افزایش می‌یابد، E_{in} به سمت کوچک‌ترین خطایی که مدل یادگیری در تقریب f قادر به دستیابی است نزدیک می‌شود (خط وسط). برای N های کوچک، مقدار E_{in} بسیار پایین‌تر از این کوچک‌ترین خطای ممکن است. علت این است که در N های کوچک، مدل یادگیری کار آسانی دارد و تنها لازم است f را فارغ از اتفاقات خارج از مجموعه روی این تعداد نقاط محدود تقریب بزند. بنابراین قادر است به یک برازش عالی روی این نقاط دست پیدا کند. هرچند این موضوع به قیمت یک برازش بد روی بقیه نقاط که با E_{out} متناظر نمایش داده‌شده است ختم می‌شود.

فصل سوم

مدل خطی

ما اغلب با مواردی برخورد می‌کنیم که باید خطی میان دو گروه رسم کنیم، درست در مقابل غلط، زندگی شخصی در مقابل زندگی حرفه‌ای، ایمیل مفید در مقابل هرزنامه. رسم یک خط معمولاً اولین انتخاب ما برای ایجاد یک مرز تصمیم‌گیری است. در یادگیری نیز همانند زندگی، خط می‌تواند یک انتخاب اول مناسب باشد.

در فصل اول ما الگوریتم پرسپترون را برای رسم یک خط میان دو طبقه بر اساس یک مجموعه داده ارائه کردیم. ما ابتدا مجموعه فرضیاتی شامل تمام خط‌های ممکن (درواقع ابر صفحه‌های ممکن) را در نظر گرفتیم. سپس با هدف یافتن یک خط مناسب، الگوریتم در یک چرخه تکراری خطاهایی که توسط خط فعلی ایجاد می‌شد را در راستای کم کردن خطای E_{in} تصحیح می‌کرد. همان‌طور که در فصل ۲ مشاهده شد مدل خطی به‌عنوان مجموعه‌ای از خطوط، بعد VC کوچکی دارد و به همین دلیل E_{in} به‌خوبی به E_{out} تعمیم پیدا می‌کند.

هدف این فصل توسعه این مدل پایه برای ایجاد ابزاری قوی در یادگیری ماشین است. ما در اینجا سه مسئله را مطرح خواهیم کرد. مسئله اول مسئله طبقه‌بندی است که ما قبلاً با آن آشنا شدیم. در ادامه دو مسئله دیگر به نام رگرسیون و برآورد احتمال[۱] را ارائه خواهیم کرد. این سه مسئله هر کدام دارای الگوریتم‌های متفاوت اما مرتبطی هستند و حوزه وسیعی‌ای را در قلمرو یادگیری ماشین پوشش می‌دهند. به‌عنوان یک قاعده سرانگشتی، "زمانی که با یک مسئله یادگیری روبرو هستید، یک راهبرد برنده اغلب این است که با یک مدل خطی شروع کنید".

[۱] probability estimation

۳-۱ طبقه‌بندی خطی

مدل خطی برای تفکیک داده‌ها به دو طبقه از یک مجموعه فرضیات $\mathcal{H}$ شامل طبقه‌بندهای[۱] خطی در قالب کلی زیر استفاده می‌کند.

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\text{T}}\mathbf{x})$$

که در آن بردارهای ستونی $\mathbf{w} \in \mathbb{R}^{d+1}$ هستند (d ابعاد فضای ورودی است و مختصه اضافی $x_0 = 1$ متناظر با وزن بایاس $\mathbf{w}_0$ است). دقت کنید فضای ورودی یک فضای d بعدی است و مختصه اضافه‌شده به شکل ثابت برابر با یک است. ما برای رجوع به یک فرضیه، از h و $\mathbf{w}$ به شکل مترادف استفاده خواهیم کرد.

زمانی که فصل ۱ را ترک می‌کردیم دو معیار اساسی را برای یادگیری معرفی کردیم:

۱. آیا می‌توان مطمئن بود که $E_{out}(g)$ به $E_{in}(g)$ نزدیک است؟ این موضوع به ما اطمینان می‌دهد آنچه در داخل نمونه یاد گرفته‌ایم به خارج از نمونه نیز قابل‌تعمیم است.

۲. آیا می‌توان $E_{in}(g)$ را کوچک کرد؟ پاسخ این سؤال به ما اطمینان می‌دهد آنچه در داخل نمونه یاد گرفتیم یک فرضیه خوب است.

شرط اول در فصل ۲ مطالعه شد. بعد VC مدل خطی مشخصاً تنها برابر با $d + 1$ است. با استفاده از کران تعمیم VC در ۲-۷ و کران ۲-۵ روی تابع رشد برحسب بعد VC به نتیجه خلاصه زیر رسیدیم، با احتمال بالا:

$$E_{out}(g) = E_{in}(g) + O\left(\sqrt{\frac{d}{N}\ln N}\right) \qquad (۳-۱)$$

طبق این معادله زمانی که N به اندازه کافی بزرگ باشد، E_{in} و E_{out} به یکدیگر نزدیک خواهند بود. به‌این‌ترتیب شرط اول یادگیری برقرار است.

شرط دوم مبنی بر اطمینان از کوچک بودن E_{in} نیازمند وجود یک یا چندین فرضیه خطی با E_{in} کوچک است. اگر چنین فرضیه‌های خطی وجود نداشته باشند، مسلماً یادگیری نیز

[۱] classifier

نمی‌تواند یکی را پیدا کند. اجازه دهید فعلاً فرض کنیم که یک مدل خطی با E_{in} کم وجود دارد. به بیان بهتر، فرض کنید داده‌ها به شکل خطی تفکیک‌پذیر هستند که به معنی وجود برخی فرضیه‌های w^* با $E_{in}(\mathsf{w}^*) = 0$ است. حالتی که این شرط برقرار نیست را به زودی بررسی خواهیم کرد.

در فصل ۱ الگوریتم یادگیری پرسپترون را ارائه کردیم. در این الگوریتم با یک بردار $\mathsf{w}(0)$ دلخواه شروع می‌کنیم و در هر گام $t \geq 0$ یک نقطه‌داده $(\mathsf{x}(t), y(t))$ که بد طبقه‌بندی شده است را انتخاب کرده و با استفاده از آن $\mathsf{w}(t)$ را طبق فرمول زیر به‌روزرسانی می‌کنیم.

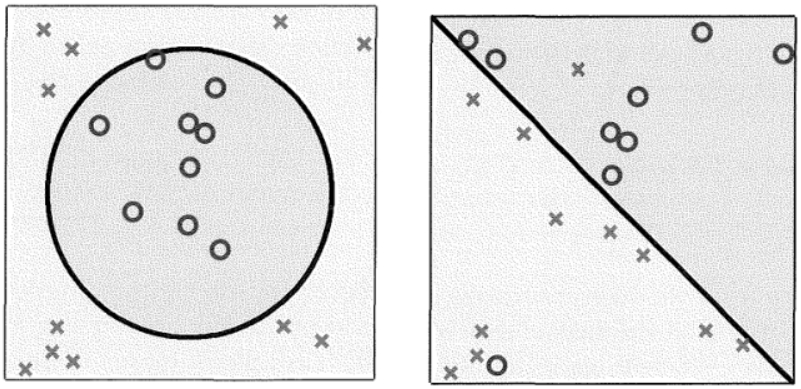
$$\mathsf{w}(t+1) = \mathsf{w}(t) + y(t)\mathsf{x}(t).$$

ایده اصلی این فرمول اصلاح خطایی است که به علت طبقه‌بندی نادرست $\mathsf{x}(t)$ ایجاد شده است. نکته جالب این است که این روند افزایشی به شکل "یک نقطه‌داده در یک‌زمان" در عمل کار می‌کند. می‌توان ثابت کرد که PLA با انجام این روند نهایتاً یک بردار $\mathsf{w}_{\mathsf{PLA}}$ با $E_{in}(\mathsf{w}_{\mathsf{PLA}}) = 0$ را پیدا خواهد کرد. هرچند این نتیجه تنها برای مجموعه محدودی (داده‌های تفکیک‌پذیر به شکل خطی) صادق است، اما همچنان گام مهمی محسوب می‌شود. الگوریتم PLA هوشمند است، به این معنی که برای یافتن فرضیه‌ای که داده‌ها را تفکیک کند هر فرضیه خطی را امتحان نمی‌کند. اگر این چنین بود به بی‌نهایت زمان احتیاج داشتیم. این الگوریتم موفق شد با استفاده از یک رویه تکراری ساده مجموعه‌ای نامتناهی از فرضیات را جستجو کرده و در زمانی متناهی یک طبقه‌بند خطی را تولید کند.

در مورد PLA، تفکیک‌پذیری خطی ویژگی داده‌ها است نه تابع هدف. یک مجموعه داده آموزشی $\mathcal{D}$ که به شکل خطی تفکیک‌پذیر است، ممکن است توسط تابع هدفی با همین ویژگی تولید شده باشد و یا به شکل تصادفی از تابع هدفی که به شکل خطی تفکیک‌ناپذیر است تولید شده باشد. اثبات همگرایی PLA تضمین می‌کند که در هر دو مورد PLA یک فرضیه نهایی با $E_{in} = 0$ را تولید خواهد کرد. همچنین طبق کران VC، در هر دو مورد می‌توان مطمئن بود که این کارایی به‌خوبی به خارج از نمونه تعمیم پیدا می‌کند.

۳-۱-۱ داده‌هایی که به شکل خطی تفکیک‌پذیر نیستند

شکل ۳-۱ دو مجموعه داده را نشان می‌دهد که به صورت خطی تفکیک‌پذیر نیستند. در قسمت (الف) پس از حذف دو مثال که می‌توان آنها را نقاط حاشیه‌ای و یا مثال‌های نویزی



(الف) مقدار کمی نویز (ب) تفکیک‌ناپذیری خطی

شکل ۳-۱: مجموعه داده‌هایی که به شکل خطی تفکی پذیر نیستند اما (الف) با حذف چند نقطه‌داده نویزی تفکیک پذیر می‌شوند. (ب) با استفاده از یک منحنی پیچیده‌تر تفکیک می‌شوند.

محسوب کرد، داده‌ها با یک خط تفکیک می‌شوند. در قسمت (ب)، می‌توان به جای خط از یک دایره برای تفکیک داده‌ها استفاده کرد. در هر دو مورد در صورت اصرار بر یافتن یک فرضیه خطی، همواره یک یا چند مثال آموزشی وجود خواهند داشت که به درستی تفکیک نشده‌اند و درنتیجه PLA هرگز خاتمه پیدا نخواهد کرد. در این حالت رفتار PLA شدیداً بی‌ثبات خواهد شد. به این معنی که ممکن است تنها با یک به‌روزرسانی از یک پرسپترون خوب به یک پرسپترون بد جهش کند و به‌این‌ترتیب کیفیت E_{in} نهایی قابل تضمین نخواهد بود.

در شکل ۳-۱ (الف) ظاهراً بهتر است ما مقدار کم نویز را تحمل کرده و همچنان از یک خط برای تفکیک داده‌ها استفاده کنیم و به جای $E_{in} = 0$ به فرضیه‌ای با E_{in} کوچک رضایت دهیم. در قسمت (ب) در نگاه اول به نظر می‌رسد مدل خطی مدل مناسبی نیست، هرچند در قسمت ۳-۴ ما تکنیکی را معرفی خواهیم کرد که با اعمال یک نگاشت غیرخطی می‌توان این گونه مسائل را نیز با همان مدل خطی پوشش داد.

وضعیتی که در شکل ۳-۱ (الف) نمایش داده‌شده است در عمل بسیار رایج است. با اینکه مدل خطی مدل مناسبی به نظر می‌رسد، اما به علت وجود مقدار کمی نویز یا مثال‌های حاشیه‌ای، داده‌ها به شکل خطی تفکیک‌پذیر نیستند. در این حالت برای اینکه فرضیه‌ای با کمترین E_{in} را پیدا کنیم، باید یک مسئله بهینه‌سازی ترکیباتی را حل کنیم.

$$\min_{\mathsf{w}\in\mathbb{R}^{d+1}} \underbrace{\frac{1}{N}\sum_{n=1}^{N}[\![\text{sign}(\mathsf{w}^{\mathrm{T}}\mathsf{x}_n)\neq y_n]\!]}_{E_{in}(\mathsf{w})} \qquad (۲-۳)$$

حل این مسئله به علت ذات گسسته هر دو توابع $\text{sign}(.)$ و $[\![.]\!]$ دشوار است. در واقع کمینه‌سازی $E_{in}(\mathsf{w})$ در ۳-۲ در حالت کلی یک مسئله (NP)-سخت محسوب می‌شود. به این معنی که هیچ الگوریتم کارآمد مناسبی برای حل آن وجود ندارد. شما اگر بتوانید چنین الگوریتمی پیدا کنید، مطمئن باشید بسیار مشهور خواهید شد. درنتیجه باید به دنبال روشی برای کمینه‌سازی E_{in} به شکل تقریبی بود.

یک رویکرد برای یافتن یک راه‌حل تقریبی این است که PLA را با انجام یک سری اصلاحات ساده که الگوریتم "پاکت[۱]" نامیده می‌شود توسعه دهیم. به‌طورکلی الگوریتم پاکت بهترین بردار وزنی که تا تکرار t ام به‌دست‌آمده است را در یک پاکت نگه داشته و در انتها بهترین بردار وزن به‌دست‌آمده را به‌عنوان فرضیه نهایی گزارش می‌کند. این الگوریتم در زیر نمایش داده‌شده است.

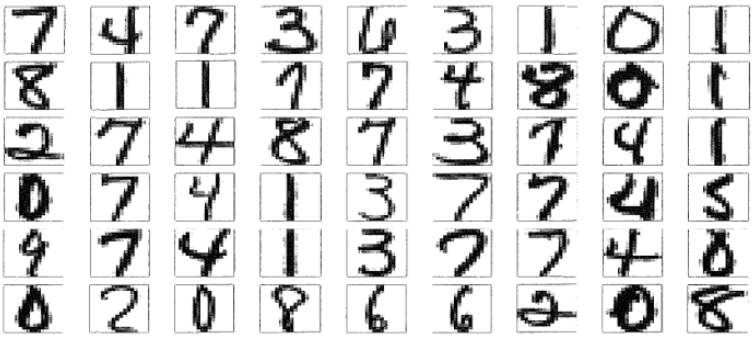
The pocket algorithm:

1: Set the pocket weight vector $\hat{\mathbf{w}}$ to $\mathbf{w}(0)$ of PLA.
2: **for** $t = 0, \ldots, T$ 1 **do**
3: Run PLA for one update to obtain $\mathbf{w}(t+1)$.
4: Evaluate $E_{\text{in}}(\mathbf{w}(t+1))$.
5: If $\mathbf{w}(t+1)$ is better than $\hat{\mathbf{w}}$ in terms of E_{in}, set $\hat{\mathbf{w}}$ to $\mathbf{w}(t+1)$.
6: Return $\hat{\mathbf{w}}$.

در PLA اصلی، برای به دست آوردن $(\mathsf{x}(t), y(t))$ در هر تکرار، تنها برخی از مثال‌ها را با جایگذاری $\mathsf{w}(t)$ بررسی می‌کردیم، درحالی‌که در الگوریتم پاکت برای محاسبه $E_{in}(\mathsf{w}(t+1))$ نیاز به یک گام اضافه‌تر برای ارزیابی همه مثال‌ها با جایگذاری $\mathsf{w}(t+1)$ داریم. این گام اضافی الگوریتم پاکت را بسیار کندتر از PLA می‌کند. همین‌طور تضمینی برای میزان همگرایی الگوریتم پاکت به یک E_{in} خوب وجود ندارد. بااین‌حال این الگوریتم به دلیل سادگی می‌تواند

[۱] pocket

یک الگوریتم دم‌دستی مفید باشد. برخی رویکردهای کاراتر بر اساس تکنیک‌های بهینه‌سازی متفاوت برای رسیدن به یک راه‌حل تقریبی بهینه توسعه داده‌شده‌اند که در قسمت‌های بعد مورد بحث قرار خواهند گرفت.

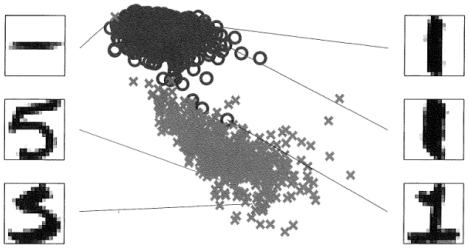


مثال تشخیص ارقام دست‌نویس: ما مجموعه‌ای از ارقام دست‌نویس مربوط به کد پستی را از پایگاه داده یک اداره پست نمونه‌برداری کردیم. این مجموعه ارقام به شکل تصاویر ۱۶ در ۱۶ پیکسل اسکن شده و مورد پیش‌پردازش قرار گرفته‌اند. هدف این است که رقم موجود در هر تصویر را شناسایی کنیم. مثال‌هایی از این ارقام در شکل بالا نمایش داده شده‌اند. یک نگاه کلی به این تصاویر نشان می‌دهد که این عمل حتی برای یک انسان کار دشواری است. تفکیک برخی ارقام مانند رقم‌های 4 از 9 و یا 2 از 7 گاهی اوقات گیج‌کننده است. E_{out} یک انسان نوعی در حدود 2.5% است. یک فرضیه یادگیری که بتواند به چنین نرخ خطایی نزدیک شود، فرضیه بسیار مطلوبی محسوب می‌شود.

اجازه دهید تکلیف سنگین تفکیک ده رقم از یکدیگر را به تکالیفی کوچک‌تر شامل تفکیک دو رقم از یکدیگر تجزیه کنیم. چنین رویکردی برای تجزیه طبقه‌بندی چند کلاسی به طبقه‌بندی دودویی بسیار رایج است و در بسیاری از الگوریتم‌های یادگیری به کار گرفته می‌شود. ما فعلاً روی ارقام 1 و 5 متمرکز می‌شویم. رویکرد معمول انسان‌ها برای تشخیص رقم موجود در هر تصویر به این شکل است که به نمای کلی یا برخی عناصر بارز در تصویر نگاه می‌کنند و

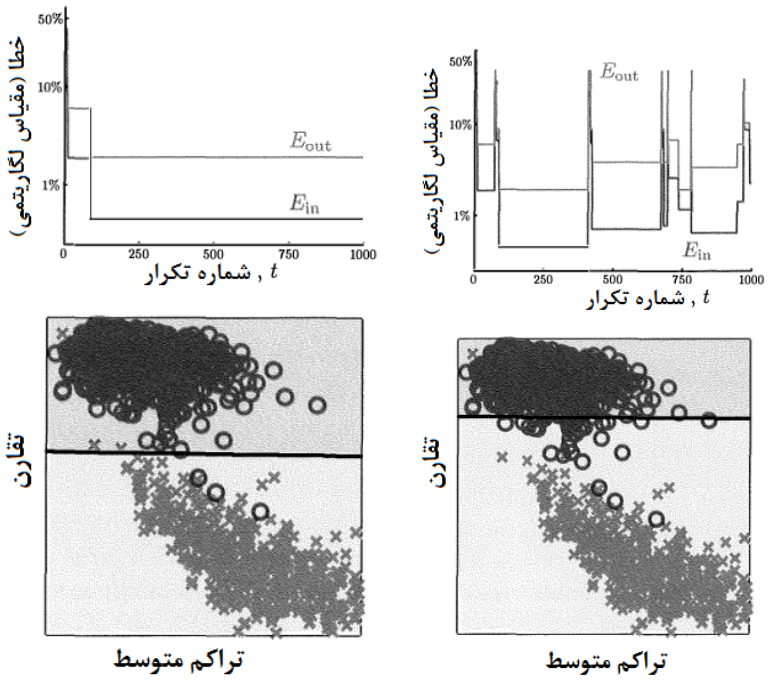
رقم مربوط را تشخیص می‌دهند. بنابراین به جای اینکه از کلیه اطلاعات مربوط به ۲۵۶ پیکسل استفاده کنیم، منطقی‌تر است که این اطلاعات را در قالب تعداد محدودی ویژگی شاخص خلاصه کنیم.

اجازه دهید به دو ویژگی مهم نگاهی بیندازیم. تراکم و تقارن. معمولاً نسبت نقاط تیره به کل نقاط تصویر برای رقم 5 بیشتر از رقم 1 است. به‌عبارت‌دیگر تراکم نقطه‌ای متوسط رقم 5 بیشتر است. به‌عنوان تفاوتی دیگر، رقم 1 متقارن است درحالی‌که رقم 5 متقارن نیست. اگر عدم تقارن را به‌عنوان متوسط اختلاف مطلق میان یک تصویر و نسخه واژگون شده آن در نظر بگیریم، تقارن مکمل این مقدار است. طبق این تعریف رقم یک دارای مقدار تقارن بالاتری است.



نمودار پراکندگی تعدادی رقم بر اساس ویژگی‌های تراکم و تقارن آنها در شکل بعد نشان داده‌شده است. درحالی‌که می‌توان این ارقام را در صفحه حاصل از این دو ویژگی به‌وسیله یک خط تا حد قابل قبولی تفکیک کرد، اما همچنان ارقامی مانند 5 در سمت چپ بالا وجود دارند که به علت بدخط بودن زیاد مانع یک تفکیک خطی ایدئال می‌شوند.

در اینجا الگوریتم‌های PLA و پاکت را روی این مجموعه داده اعمال می‌کنیم تا ببینیم چه اتفاقی می‌افتد. ازآنجاکه مجموعه داده به شکل خطی تفکیک‌پذیر نیست، به‌روزرسانی بردار وزن در PLA هیچ‌گاه متوقف نخواهد شد. در این حالت همان‌طور که در شکل ۳-۲ (الف) مشاهده می‌شود، رفتار الگوریتم ممکن است کاملاً بی‌ثبات باشد. زمانی که آن را در تکرار هزارم به اجبار متوقف می‌کنیم، خط حاصل دارای E_{in} ضعیفی برابر با 2.24% و E_{out}ی برابر با 6.37% است. چنانچه الگوریتم پاکت را به همان مجموعه داده اعمال کنیم، همان‌طور که در شکل ۳-۲ب (ب) نشان داده‌شده است، به خطی خواهیم رسید که دارای E_{in} بهتری برابر با 0.45% و E_{out} بسیار بهتری برابر با 1.89% است.



شکل ۳-۲: مقایسه الگوریتم PLA و الگوریتم pocket برای تفکیک رقم 1 و 5.

۳-۲ رگرسیون خطی

رگرسیون خطی مدل خطی مفید دیگری است که به توابع هدف مقدار-حقیقی [۱] اعمال می‌شود. این روش دارای یک تاریخچه طولانی در علم آمار است و در آنجا با جزئیات کامل مطالعه می‌شود. همین‌طور دارای کاربردهای فراوانی در علوم رفتاری و اجتماعی است. ما در اینجا رگرسیون خطی را از منظر یادگیری مورد بحث قرار می‌دهیم و با در نظر گرفتن کمترین فرضیات نتایج اصلی را استخراج می‌کنیم.

اجازه دهید یک بار دیگر مثال کارت اعتباری را مرور کنیم. این بار به جای یک مسئله طبقه‌بندی، یک مسئله رگرسیونی را در نظر بگیرید. همان‌طور که قبلاً گفته شد بانک پرونده سوابق هر مشتری شامل برخی فقره‌های اطلاعاتی در مورد اعتبار اشخاص مانند حقوق سالانه،

[۱] real-valued

سوابق کار، وام‌های عمده و غیره را در اختیار دارد. این اطلاعات می‌توانند برای یادگیری یک طبقه‌بند خطی برای تصمیم‌گیری در مورد تائید اعتبار استفاده شوند. فرض کنید علاوه بر یک تصمیم دودویی مبنی بر تائید یا عدم تائید یک درخواست، بانک قصد دارد برای هر مشتری که تائید می‌شود یک حد مجاز اعتبار در نظر بگیرد. به شکل سنتی این حد توسط تعدادی فرد خبره محاسبه می‌شود. بانک قصد دارد در کنار خودکارسازی فرایند تائید اعتبار، این عمل را نیز خودکار کند.

این مسئله یک مسئله یادگیری رگرسیون است. بانک با استفاده از سوابق مشتریان یک مجموعه داده از مثال‌هایی به شکل $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ ایجاد می‌کند که در آن $\mathbf{x}_n$ اطلاعات یک مشتری و y_n حد اعتبار تعیین شده برای آن مشتری توسط یکی از خبرگان بانک است. دقت کنید که در اینجا y_n به جای مقادیر دودویی ± 1، یک عدد حقیقی مثبت است. بانک قصد دارد با استفاده از یادگیری، یک فرضیه g را پیدا کند که روش تعیین حد اعتبار توسط خبره‌های انسانی را تقلید کند.

ازآنجاکه بیشتر از یک خبره وجود دارد و هر خبره نیز ممکن است کاملاً باثبات رفتار نکند، تابع هدف یک تابع قطعی $y = f(\mathbf{x})$ نیست، بلکه هدفی نویزی است که در قالب توزیعی از متغیر تصادفی y که از دیدگاه‌های متغیر و متفاوت خبرگان مختلف بر می‌آید فرموله می‌شود. به‌عبارت‌دیگر برچسب y_n به جای نتیجه یک تابع قطعی $f(\mathbf{x})$، نتیجه یک توزیع $P(y \mid \mathbf{x})$ است. بااین‌وجود همان‌طور که در فصل ۱ تشریح شد کلیت مسئله فرقی نمی‌کند. یک توزیع ناشناخته $P(\mathbf{x}, y)$ وجود دارد که نقاط (x_n, y_n) را تولید کرده است و ما قصد داریم یک فرضیه g پیدا کنیم که خطای میان $g(\mathbf{x})$ و y را با توجه به آن توزیع کمینه کند.

پیش‌فرض ما برای انتخاب یک مدل خطی برای این مسئله این است که یک ترکیب خطی از فیلدهای اطلاعات مشتریان وجود دارد که حد اعتبار را همان‌گونه که خبره‌های انسانی تعیین می‌کنند، به شکل مناسبی تقریب می‌زند. اگر چنین فرضی برقرار نباشد، نمی‌توان با استفاده از یک مدل خطی به یک خطای پایین دست پیدا کرد. زمانی که نگاشت‌های غیرخطی را توضیح می‌دهیم، به این موضوع بیشتر خواهیم پرداخت.

۳-۲-۱ الگوریتم

الگوریتم رگرسیون خطی بر اساس کمینه‌سازی خطای مربع میان $h(\mathbf{x})$ و y قرار دارد.

$$E_{out}(h) = \mathbb{E}\left[(h(\mathbf{x}) - y)^2\right]،$$

که در آن مقدار مورد انتظار طبق توزیع احتمال توأم $P(\mathsf{x}, y)$ به دست می‌آید. هدف یافتن فرضیه‌ای است که کمترین $E_{out}(h)$ را دارا باشد. ازآنجاکه توزیع $P(\mathsf{x}, y)$ ناشناخته است، $E_{out}(h)$ نیز قابل محاسبه نیست. در انیجا نیز همانند مسئله طبقه‌بندی، ما به نسخه درون-نمونه این خطا رجوع می‌کنیم.

$$E_{in}(h) = \frac{1}{N} \sum_{n=1} N(h(\mathsf{x}_n) - y_n)^2$$

در رگرسیون خطی h به شکل یک ترکیب خطی از مؤلفه‌های x است:

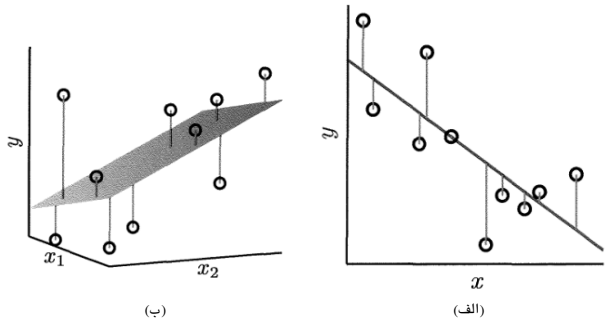
$$h(\mathsf{x}) = \sum_{i=0}^{d} w_i x_i = \mathsf{w}^{\mathrm{T}}\mathsf{x}$$

که در آن $x_0 = 1$ و مطابق معمول $\mathsf{x} \in \{1\} \times \mathbb{R}^d$ و $\mathsf{w} \in \mathbb{R}^{d+1}$ است. برای مورد ویژه h خطی، ارائه یک نمایش ماتریسی از $E_{in}(h)$ می‌تواند بسیار مفید باشد. ابتدا ماتریس داده $\mathsf{X} \in \mathbb{R}^{N \times (d+1)}$ را به شکل یک ماتریس $N \times d + 1$ بعدی در نظر بگیرید. در این ماتریس هر سطر نماینده یک مثال از مجموعه داده $\mathcal{D}$ است. همین‌طور بردار هدف $y \in \mathbb{R}^N$ را به شکل یک بردار ستونی که عناصر آن مقادیر هدف y_n هستند تعریف کنید. با توجه به این نمایش، خطای درون-نمونه به‌عنوان تابعی از w و داده‌های ماتریس‌های X و y قابل‌محاسبه است.

$$\begin{aligned} E_{in}(\mathsf{w}) &= \frac{1}{N} \sum_{n=1}^{N} (\mathsf{w}^{\mathrm{T}}\mathsf{x}_n - y_n)^2 \\ &= \frac{1}{N} \|\mathsf{X}\mathsf{w} - \mathsf{y}\|^2 \\ &= \frac{1}{N} (\mathsf{w}^{\mathrm{T}}\mathsf{X}^{\mathrm{T}}\mathsf{X}\mathsf{w} - 2\mathsf{w}^{\mathrm{T}}\mathsf{X}^{\mathrm{T}}\mathsf{y} + \mathsf{y}^{\mathrm{T}}\mathsf{y}), \end{aligned}$$

الگوریتم رگرسیون خطی با کمینه کردن $E_{in}(\mathsf{w})$ روی کلیه $\mathsf{w} \in \mathbb{R}^{d+1}$ استخراج می‌شود. این مسئله بهینه‌سازی به شکل زیر خواهد بود.

$$\mathsf{w}_{\mathsf{lin}} = \underset{\mathsf{w} \in \mathbb{R}^{d+1}}{\arg\min}\, E_{in}(\mathsf{w}) \qquad (۳-۳)$$



شکل ۳-۳: فرضیه نهایی حاصل از رگراسیون خطی در فضای یک بعدی و ۲ بعدی. حاصل‌جمع خطاهای مربعی کمینه شده است.

شکل ۳-۳ نتیجه حل این مسئله را در ابعاد یک‌بعدی و دوبعدی نشان می‌دهد. راه‌حل این مسئله در آمار به طور کامل توضیح داده می‌شود و ما در اینجا قصد نداریم وارد جزئیات آن شویم. به‌طور خلاصه ابتدا گرادیان ماتریس خطا را محاسبه کرده و برابر با صفر قرار می‌دهیم. با حل این معادله، بردار w به شکل زیر حاصل خواهد شد:

$$\mathsf{w}_{\text{lin}} = ((\mathsf{X}^{\mathrm{T}}\mathsf{X})^{-1}\mathsf{X}^{\mathrm{T}})\mathsf{y}. \tag{۴-۳}$$

در مقایسه با الگوریتم یادگیری پرسپترون، رگرسیون خطی خیلی شبیه یادگیری به نظر نمی‌رسد. از این جهت که به جای یک فرایند تکراری، فرضیه w_{lin} مستقیماً با استفاده از یک راه‌حل تحلیلی قابل‌محاسبه است. بااین‌وجود تا زمانی که فرضیه w_{lin} دارای یک خطا خارج-از-نمونه مطلوب باشد، یادگیری صورت گرفته است. در حقیقت، رگرسیون خطی از موارد نادری است که ما برای حل آن یک فرمول تحلیلی در اختیار داریم و این یکی از دلایل اصلی رایج بودن این تکنیک است.

رگرسیون خطی با جزئیات بیشتر در آمار بحث می‌شود. در این روش، بردار وزن w_{lin} تلاشی برای نگاشت ورودی x به خروجی y است، هرچند w_{lin} دقیقاً y را تولید نمی‌کند اما تقریبی از آن را فراهم می‌کند. این تقریب در فرمول زیر با کلاه نمایش داده شده است که اختلاف آن با y ناشی از وجود خطای درون نمونه است:

$$\hat{\mathsf{y}} = \mathsf{X}\mathsf{w}_{\text{lin}}$$

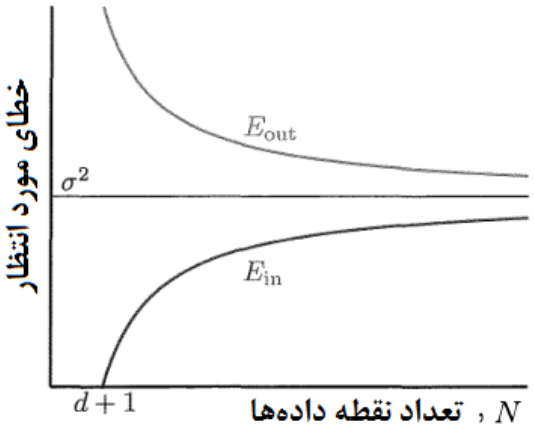
۳-۲-۲ پیامدهای تعمیم

همان‌طور که مشاهده شد رگرسیون خطی به دنبال یافتن بردار وزن بهینه برحسب خطای درون نمونه E_{in} است. این مطلب سؤال معمول تعمیم را در پی دارد. آیا این راه‌حل می‌تواند تضمین‌کننده یک E_{out} مناسب باشد یا خیر؟ جواب کوتاه بله است. یک نسخه رگرسیونی از کران تعمیم VC وجود دارد که به‌طور مشابه کرانی برای E_{out} فراهم می‌کند. به‌طور مشخص، در مورد رگرسیون خطی فرمولهای دقیقی برای محاسبه مقادیر مورد انتظار E_{in} و E_{out} وجود دارد که تحت برخی فرضیات ساده کننده استخراج می‌شوند. فرم کلی نتیجه به شکل زیر است:

$$E_{out}(g) = E_{in}(g) + O\left(\frac{d}{N}\right)$$

که $E_{out}(g)$ و $E_{in}(g)$ مقادیر مورد انتظار هستند. این معادله با کران طبقه‌بندی در ۳-۱ قابل قیاس است.

شکل ۳-۴ منحنی یادگیری رگرسیون خطی را تحت برخی فرضیات نشان می‌دهد. در این شکل، از یک تابع هدف نویزی به شکل $y = \mathbf{w}^{*\mathrm{T}}\mathbf{x} + \epsilon$ برای تولید داده‌ها استفاده شده است. در این تابع، ϵ ترم نویزی با میانگین صفر و واریانس σ^2 است و به شکل مستقل برای هر $\mathbf{x}$ و y تولید می‌شود. مقدار مورد انتظار خطا برای بهترین برازش خطی ممکن همان σ^2 است.



شکل ۳-۴: منحنی یادگیریِ رگراسیون خطی

۳-۳ رگرسیون لجستیک

هسته مدل خطی، سیگنال $s = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ است که متغیرهای ورودی را به شکل خطی ترکیب می‌کند. تاکنون دو مدل بر اساس این سیگنال ارائه کردیم و اکنون قصد داریم سومین مدل را معرفی کنیم.

در رگرسیون خطی خود سیگنال به‌عنوان خروجی استفاده می‌شود. این حالت زمانی استفاده می‌شود که شما قصد دارید یک پاسخ مقدار-حقیقی و بدون محدودیت را پیش‌بینی کنید. در طبقه‌بندی خطی این سیگنال با استفاده از مقدار صفر آستانه‌گذاری می‌شود تا یک خروجی ± 1 را تولید کند. این روش برای ایجاد تصمیمات دودویی مناسب است.

امکان سومی نیز وجود دارد که در عمل بسیار کاربردی است و آن تولید یک احتمال به‌عنوان عددی بین صفر و یک است. این مدل جدید "رگرسیون لجستیک" [۱] خوانده می‌شود و به هر دو مدل قبلی شباهت‌هایی دارد. از یک جهت همانند رگرسیون، خروجی آن عددی حقیقی است و از جهت دیگر همانند طبقه‌بندی مقداری محدود شده است (بین صفر و یک).

برای مثال فرض کنید قصد داریم احتمال حمله قلبی یک فرد را بر اساس سطح کلسترول، فشارخون، وزن و بقیه عوامل دخیل پیش‌بینی کنیم. مسلماً نمی‌توان چنین اتفاقی را با قطعیت پیش‌بینی کرد. اما ممکن است بتوان میزان احتمال آن را بر اساس این عوامل پیش‌بینی کرد. بنابراین در این مسئله تولید یک خروجی پیوسته میان صفر و یک، نسبت به یک تصمیم دودویی مدل مناسب‌تری خواهد بود. هرچه y به ۱ نزدیک‌تر باشد احتمال حمله قلبی بیشتر است.

۳-۳-۱ پیش‌بینی یک احتمال

طبقه‌بندی خطی از یک آستانه سخت روی سیگنال $s = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ استفاده می‌کرد:

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

در مقابل رگرسیون خطی از هیچ آستانه‌ای استفاده نمی‌کرد.

$$h(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

در مدل جدید ما به چیزی بین این دو حالت نیاز داریم که خروجی را به شکل ملایمی در بازه صفر و یک محدود کند.

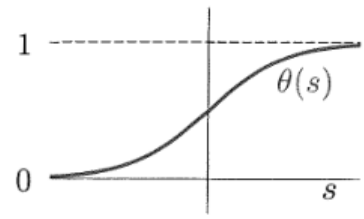
[۱] logistic regression

یک راه برای رسیدن به این هدف مدل رگرسیون لجستیک است:

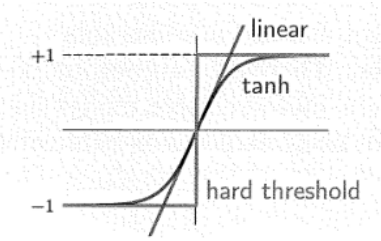
$$h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

که در آن θ تابع لجستیک خوانده می‌شود $\theta(s) = \frac{e^s}{1+e^s}$ و مقداری بین صفر و یک تولید می‌کند.

می‌توان از این خروجی به‌عنوان احتمال یک پیشامد دودویی استفاده کرد (برای مثال حمله قلبی یا عدم حمله قلبی، رقم 1 یا رقم 5). در طبقه‌بندی خطی نیز ما با یک پیشامد دودویی مواجه بودیم. اما تفاوت در این است که در رگرسیون لجستیک 'طبقه‌بندی' غیرقطعی است و اجازه دارد مقداری بین صفر و یک که نشان‌دهنده میزان این عدم قطعیت است را اختیار کند. در مقایسه با آستانه سخت در طبقه‌بندی، تابع لجستیک θ یک "آستانه نرم" خوانده می‌شود.



(الف) تابع لجستیک



(ب) مقایسه آستانه طبقه‌بندی، رگرسیون لجستیک و رگرسیون خطی

فرمول ویژه $\theta(s)$ به ما اجازه می‌دهد مقیاس خطایی را برای یادگیری تعریف کنیم که دارای مزایای محاسباتی و تحلیلی بسیاری است و به زودی به آنها اشاره خواهیم کرد. اجازه دهید ابتدا به هدفی که رگرسیون لجستیک قصد دارد یاد بگیرد نگاهی بیندازیم. هدف یک احتمال است؛ برای مثال آیا یک بیمار در خطر حمله قلبی است یا خیر. این احتمال وابسته به ورودی $\mathbf{x}$ به‌عنوان خصوصیات بیمار است. به شکل رسمی ما قصد داریم تابع هدفی به شکل زیر را یاد بگیریم:

$$f(\mathbf{x}) = \mathbb{P}[y = +1 \mid \mathbf{x}]$$

داده‌ها قادر نیستند f را به شکل صریح مشخص کنند، بلکه برخی نمونه‌ها که به‌وسیله این احتمال تولید شده است را به ما می‌دهند. بیمارانی با یک سری خصوصیات که حمله قلبی

داشتند و بیمارانی که حمله قلبی نداشتند. درنتیجه این داده‌ها توسط یک هدف نویزی $P(y \mid \mathbf{x})$ تولید می‌شوند:

$$P(y \mid \mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1; \\ 1 - f(\mathbf{x}) & \text{for } y = -1. \end{cases} \tag{۵-۳}$$

برای یادگیری از چنین مجموعه داده‌ای ما احتیاج به یک مقیاس خطای مناسب داریم که میزان نزدیکی یک فرضیه مشخص h به f را برحسب این مثال‌های نویزی ± 1 اندازه بگیرد.

مقیاس خطا: مقیاس خطای استاندارد $\mathrm{e}(h(\mathbf{x}), y)$ که در رگرسیون لجستیک استفاده می‌شود بر اساس مفهوم "درستنمایی"[۱] قرار دارد. به این معنی که اگر توزیع هدف $P(y \mid \mathbf{x})$ به‌طور کامل به‌وسیله فرضیه $h(\mathbf{x})$ پوشش داده شود، چقدر احتمال دارد خروجی y از ورودی $\mathbf{x}$ بدست آید.

بر اساس ۳-۵ این درستنمایی به شکل زیر است:

$$P(y \mid \mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1; \\ 1 - h(\mathbf{x}) & \text{for } y = -1. \end{cases}$$

ما $h(\mathbf{x})$ را با $\theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ جایگزین می‌کنیم و از این واقعیت که $1 - \theta(s) = \theta(-s)$ استفاده می‌کنیم تا به جمله زیر برسیم:

$$P(y \mid \mathbf{x}) = \theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x}) \tag{۶-۳}$$

یکی از دلایلی که ما فرم ریاضی $\theta(s) = \frac{e^s}{1+e^s}$ را انتخاب کردیم این است که این تابع به عبارت ساده‌ای برای $P(y \mid \mathbf{x})$ منتهی می‌شود.

ازآنجاکه نقطه‌داده‌های $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ به شکل مستقل تولید شده‌اند. احتمال اینکه بتوانیم کلیه y_n هایی که در مجموعه داده هستند را از $\mathbf{x}_n$ متناظرشان به دست بیاوریم به

[۱] likelihood

شکل زیر خواهد بود.

$$\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n).$$

روش "بیشینه‌سازی درستنمایی" [۱]، فرضیه‌ای را انتخاب می‌کند که این احتمال را بیشینه کند. برای این کار به شکل معادل می‌توانیم یک کمیت ساده‌تر را کمینه کنیم.

$$-\frac{1}{N}\ln\left(\prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{P(y_n|\mathbf{x}_n)}\right)$$

که در آن $-\frac{1}{N}\ln(.)$ یک تابع نزولی یکنواخت است. چنانچه معادله ۳-۶ را در این فرمول جایگزین کنیم، به مسئله کمینه‌سازی کمیت زیر برحسب بردار وزن $\mathbf{w}$ می‌رسیم:

$$\frac{1}{N}\sum_{n=1}^{N}\ln\left(\frac{1}{\theta(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)}\right)$$

این موضوع که ما این کمیت را کمینه می‌کنیم به ما اجازه می‌دهد آن را به‌عنوان مقیاس خطا در نظر بگیریم. با جایگزین کردن تابع $\theta(y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$، خطای درون-نمونه برای رگرسیون لجستیک به شکل زیر حاصل می‌شود:

$$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e_{-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right) \qquad (۳-۷)$$

مقیاس خطای نقطه‌به‌نقطه‌ای که نتیجه می‌شود عبارت است از :

$$\mathsf{e}(h(x_n), y_n) = \ln(1+e^{-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n})$$

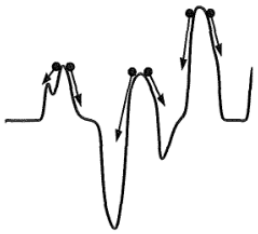
دقت کنید که این مقیاس خطا زمانی کوچک است که $y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n$ بزرگ و مثبت باشد. این مطلب به این معنی است که $y_n = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$ است. همان‌طور که انتظار داشتیم، این مقیاس خطا باعث می‌شود $\mathbf{w}$ هر $\mathbf{x}_n$ را به‌درستی طبقه‌بندی کند.

روش‌هایی که برای طبقه‌بندی خطی و رگرسیون خطی توسعه داده شدند را نمی‌توان در مورد رگرسیون لجستیک بکار گرفت. در این حالت در رویکردی مشابه رگرسیون خطی سعی

[۱] maximum likelihood

می‌کنیم گرادیان E_{in} را برابر با صفر قرار داده و معادله را حل کنیم. متأسفانه برخلاف رگرسیون خطی، حل این معادله به شکل تحلیلی ساده نیست. به جای استفاده از یک روش تحلیلی ما از یک روش تکراری استفاده می‌کنیم. به این منظور الگوریتم "گرادیان نزولی[۱]" را معرفی می‌کنیم.

۳-۳-۲ گرادیان نزولی



گرادیان نزولی یک الگوریتم عمومی است که می‌توان از آن برای آموزش بسیاری از مدل‌های یادگیری با یک تابع خطای هموار (دو بار مشتق‌پذیر) استفاده کرد.

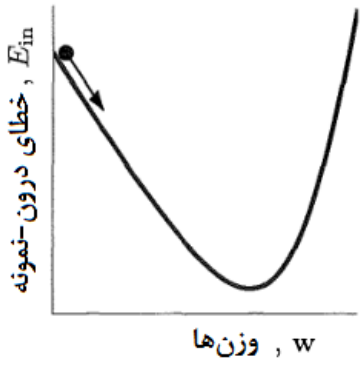
برای تصور تحلیل گرادیان، فرض کنید یک توپ را در یک سطح تپه‌ای قل می‌دهید. چنانچه توپ روی یک تپه رها شود به سمت پایین حرکت خواهد کرد و در کف یک دره ساکن می‌شود. همین ایده را می‌توان در مورد گرادیان نزولی بکار برد. $E_{in}(\mathbf{w})$ متناظر با چنین سطحی در یک فضایی چندبعدی با ابعاد زیاد است. در گام صفر ما روی نقطه‌ای از این سطح قرار داریم و سعی می‌کنیم با حرکت به سمت پایین E_{in} را کاهش دهیم.

نکته‌ای که با توجه به تشبیه فیزیکی گرادیان نزولی به ذهن می‌رسد این است که توپ لزوماً در پایین‌ترین نقطه در کل صفحه قرار نمی‌گیرد، بلکه بسته به اینکه در ابتدا در چه نقطه‌ای رها شود، نهایتاً در پایین یکی از دره‌ها متوقف خواهد شد که به آن "کمینه محلی[۲]" گفته می‌شود. به‌طور مشابه این موضوع در مورد گرادیان نزولی نیز صادق است و بسته به وزن‌های ابتدایی، شما نهایتاً در یک کمینه محلی در صفحه خطا متوقف خواهید شد.

اما در مورد رگرسیون لجستیک با خطای آنتروپی متقابل[۳] مزیت ویژه‌ای وجود دارد و آن این است که تصویر سطح ازآنچه توصیف شد بسیار بهتر است. در این حالت تنها یک دره وجود دارد. درنتیجه فارغ از اینکه توپ را در ابتدا کجا رها کنید، همواره به سمت کمینه سراسری[۴]

[۱] gradient descent
[۲] local minimum
[۳] cross-entropy error
[۴] global minimum

قل خواهد خورد. این موضوع نتیجه این واقعیت است که $E_{in}(\mathbf{w})$ یک تابع محدب از $\mathbf{w}$ است. تحدب یک ویژگی ریاضی است که تضمین کننده وجود یک دره یکتا است (همانند شکل زیر). بنابراین گرادیان نزولی هرگز در یک کمینه محلی به دام نخواهد افتاد.



به طور خلاصه در این الگوریتم، در یک چرخه تکراری شما گام به گام در جهت شیب‌دارترین بردار رو به پایین حرکت کرده و در هر گام بردار $\mathbf{w}$ را بروز رسانی می‌کنید. جزئیات کامل این الگوریتم در ریاضیات وجود دارد و ما در اینجا تنها به همین توضیح کلی بسنده می‌کنیم.

مثال ۱-۳: در اینجا قصد داریم با استفاده از مثال کارت اعتباری، مدل‌های خطی متفاوتی که تاکنون دیدیم را خلاصه کنیم. اگر هدف تائید یا رد درخواست کارت اعتباری باشد، در آن صورت در قلمرو طبقه‌بندی هستیم. اگر هدف تعیین حد اعتبار مشخصی برای یک درخواست باشد، در آن صورت رگرسیون خطی گزینه مناسب است. در آخر چنانچه هدف پیش‌بینی احتمال بدحسابی یک متقاضی باشد، از رگرسیون لجستیک استفاده کنید.

هر یک از این سه مدل خطی اهداف متفاوتی را دنبال می‌کنند و دارای مقیاس‌های خطا و الگوریتم‌های متناظر خود هستند. بااین‌حال نه‌تنها در مجموعه فرضیات خطی مشابهی مشترک هستند، بلکه از جهات دیگری نیز مرتبط هستند. در اینجا قصد داریم به یک ارتباط مهم اشاره کنیم. هر دوی رگرسیون لجستیک و رگرسیون خطی می‌توانند برای طبقه‌بندی خطی استفاده شوند. در ادامه این مطلب را توضیح می‌دهیم.

رگرسیون لجستیک یک فرضیه نهایی $g(\mathbf{x})$ را تولید می‌کند که نشان‌دهنده تخمین ما از $P(y = +1 \mid \mathbf{x})$ است. به‌آسانی می‌توان با انتساب یک آستانه روی $g(\mathbf{x})$ از این تخمین برای طبقه‌بندی استفاده کرد. یک آستانه طبیعی برای این عمل 0.5 است، به این معنی که احتمال بالای 0.5 به معنی طبقه‌بندی $+1$ و احتمال زیر آن به معنی طبقه‌بندی -1 است. این عمل معادل استفاده از وزن‌های رگرسیون لجستیک به‌عنوان وزن‌های موردنیاز در پرسپترون برای طبقه‌بندی است.

نه‌تنها می‌توان از وزن‌های به‌دست‌آمده از رگرسیون لجستیک برای طبقه‌بندی به این شکل استفاده کرد، بلکه می‌توان این روش را به‌عنوان روشی برای آموزش مدل پرسپترون به‌کاربرد. در حقیقت مسئله یادگیری پرسپترون یک مسئله بهینه‌سازی ترکیباتی است که حل آن دشوار است. در عوض تحدب E_{in} در رگرسیون لجستیک مسئله بهینه‌سازی را بسیار آسان‌تر خواهد کرد. ازآنجاکه تابع لجستیک یک نسخه نرم از یک آستانه سخت است، وزن‌های رگرسیون لجستیک می‌توانند وزن‌های خوبی برای طبقه‌بندی با استفاده از پرسپترون باشند.

ارتباط مشابهی میان طبقه‌بندی و رگرسیون خطی وجود دارد. رگرسیون خطی می‌تواند برای هر تابع هدف مقدار-حقیقی که شامل مقادیر حقیقی ± 1 است استفاده شود. چنانچه $\mathbf{w}_{\text{lin}}^{\mathrm{T}}\mathbf{x}$ با این مقادیر ± 1 به‌خوبی منطبق شود، $sign(\mathbf{w}_{\text{lin}}^{\mathrm{T}}\mathbf{x})$ نیز احتمالاً با این مقادیر منطبق خواهد شد و یک پیش‌بینی طبقه‌بندی خوب را فراهم خواهد کرد. به بیان دیگر وزن‌های رگرسیون خطی $\mathbf{w}_{\text{lin}}$ که با استفاده از وارون ماتریس به‌آسانی قابل‌محاسبه هستند می‌توانند راه‌حل تقریبی خوبی برای مدل پرسپترون باشند. می‌توان به شکل مستقیم از این وزن‌ها برای طبقه‌بندی استفاده کرد و یا از آنها به‌عنوان شرایط اولیه برای اجرای الگوریتم پاکت استفاده کرد.

۳-۴ نگاشت غیرخطی

کلیه فرمول‌هایی که تا کنون برای مدل خطی ارائه‌شدند از حاصل جمع زیر به‌عنوان اصلی‌ترین کمیت در محاسبه فرضیه خروجی استفاده می‌کنند.

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{i=0}^{d} w_i x_i \qquad (۸-۳)$$

این کمیت نه‌تنها بر حسب x_i ها بلکه بر حسب w_i ها نیز کمیتی خطی است. بررسی دقیق‌تر الگوریتم‌های یادگیری متناظر نشان می‌دهد پیش‌نیاز اصلی استخراج این الگوریتم‌ها، خطی بودن در w_i ها است. از دیدگاه این الگوریتم‌ها x_i ها تنها یک سری ثوابت به شمار می‌آیند. این مطلب به ما امکان استفاده از نسخه‌های غیرخطی x_i ها را فراهم می‌کند درحالی‌که همچنان می‌توانیم در حوزه تحلیلی مدل خطی باقی بمانیم. زیرا معادله ۲-۷ در پارامترهای w_i همچنان خطی باقی خواهد ماند.

برای مثال مسئله تعیین حد اعتبار را در نظر بگیرید. این که فیلد سابقه کار بر روی این حد اعتبار تأثیر مثبت داشته باشد منطقی به نظر می‌رسد، زیرا نشان‌دهنده پایداری و ثبات وضعیت فرد است. اما افزایش خطی آن به نسبت سال‌های اشتغال ایده جالبی نیست. بهتر است یک آستانه پایه وجود داشته باشد (برای مثال یک سال) که چنانچه سابقه کار کمتر از آن بود، بر روی حد اعتبار تأثیر منفی بگذارد و آستانه دیگری (برای مثال ۵ سال) وجود داشته باشد که سابقه بالاتر از آن تأثیر مثبتی بر روی این حد بگذارد. اگر x_i متغیری ورودی باشد که نشان‌دهنده سال‌های اشتغال فرد است، دو 'ویژگی' [۱] غیرخطی می‌توان از آن استخراج کرد: یکی بازه $[\![x_i < 1]\!]$ و دیگری بازه $[\![x_i > 5]\!]$ که به یک فرمول خطی اجازه می‌دهند حد اعتبار را به شکل بهتری منعکس کند.

ما اخیراً نحوه استفاده از ویژگی‌ها را در طبقه‌بندی ارقام دست‌نویس نشان دادیم که در آن دو ویژگی تراکم و تقارن برای تصاویر ورودی استخراج شدند. می‌توانیم تبدیل غیرخطی را به این ویژگی‌ها اعمال کرده و ویژگی‌های جزئی‌تری را تولید کنیم و به‌این‌ترتیب کارایی را افزایش دهیم. در صورتی که بتوانیم ورودی را با ویژگی‌های مناسب‌تری نمایش دهیم، دامنه کاربرد روش‌های خطی به‌شدت افزایش پیدا می‌کند.

۳-۴-۱ فضای Z

وضعیتی که در شکل ۳-۱ (ب) نشان داده‌شده است را در نظر بگیرید که در آن هیچ طبقه‌بند خطی قادر به برازش داده‌ها نیست. در اینجا می‌توانیم با تبدیل ورودی x_1 و x_2 به یک فرم غیرخطی، داده‌ها را با مرزهای پیچیده‌تری تفکیک کنیم، درحالی‌که همچنان از همان الگوریتم PLA استفاده کنیم.

اجازه دهید به دایره‌ای که در شکل ۳-۵ (الف) رسم شده است و تکرار همان مورد

[۱] feature

غیرقابل‌تفکیک در شکل ۳-۱ (ب) است، نگاه دقیق‌تری بیندازیم. این دایره با فرمول زیر مشخص می‌شود:

$$x_1^2 + x_2^2 = 0.6$$

در اینجا فرضیه غیرخطی $h(\mathbf{x}) = \text{sign}(-0.6 + x_1^2 + x_2^2)$ این مجموعه داده را به شکل ایدئال تفکیک می‌کند. می‌توان با اعمال یک تبدیل غیرخطی روی $\mathbf{x}$، این فرضیه را به قالب یک فرضیه خطی ببریم. به‌طور مشخص فرض کنید $z_0 = 1$، $z_1 = x_1^2$ و $z_2 = x_2^2$

$$\begin{aligned} h(\mathbf{x}) &= \text{sign}\left(\underbrace{-0.6}_{\tilde{w}_0}\cdot\underbrace{1}_{z_0} + \underbrace{1}_{\tilde{w}_1}\cdot\underbrace{x_1^2}_{z_1} + \underbrace{1}_{\tilde{w}_2}\cdot\underbrace{x_2^2}_{z_2}\right) \\ &= \text{sign}\left(\underbrace{[\tilde{w}_0, \tilde{w}_1, \tilde{w}_2]}_{\tilde{\mathbf{w}}^{\mathrm{T}}}\underbrace{\begin{bmatrix}1\\ z_1\\ z_2\end{bmatrix}}_{\mathbf{z}}\right) \end{aligned}$$

که در آن بردار $\mathbf{z}$ با اعمال یک تبدیل غیرخطی Φ روی $\mathbf{x}$ به دست می‌آید.

همان‌طور که در شکل ۳-۵ (ب) نشان داده‌شده است، می‌توانیم داده‌ها را به جای $\mathbf{x}$ برحسب $\mathbf{z}$ رسم کنیم. برای مثال در این شکل نقطه $\mathbf{x}_1$ در قسمت(الف) به نقطه $\mathbf{z}_1$ در قسمت (ب) و نقطه $\mathbf{x}_2$ به $\mathbf{z}_2$ نگاشته شده اند. فضای $\mathcal{Z}$ که حاوی بردارهای $\mathbf{z}$ است به‌عنوان "فضای ویژگی[۱]" شناخته می‌شود، زیرا مختصات آن ویژگی‌های سطح بالاتری هستند که از ورودی خام $\mathbf{x}$ استخراج شده‌اند. ما کمیت‌های مرتبط با $\mathcal{Z}$ را با علامت مَد [۲] به هم‌رتبه‌هایشان در $\mathcal{X}$ متناظر می‌کنیم. برای مثال ابعاد $\mathcal{Z}$ را با $\tilde{d}$ و بردار وزن را با $\tilde{\mathbf{w}}$ نشان می‌دهیم. تبدیل Φ که ما را از فضای $\mathcal{X}$ به فضای $\mathcal{Z}$ می‌برد، "نگاشت ویژگی[۳]" خوانده می‌شود که در مورد بالا برابر است با:

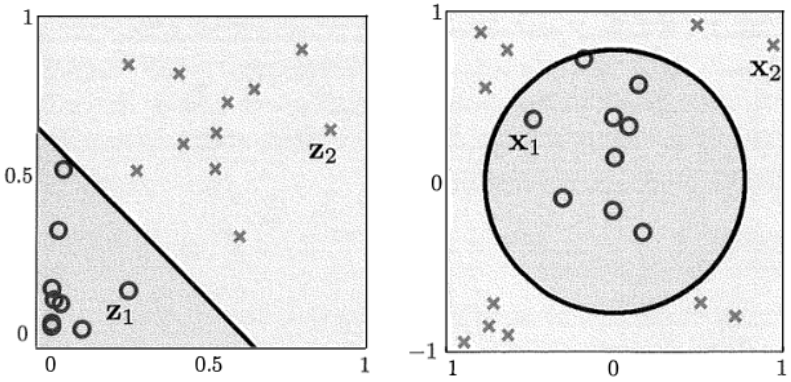
$$\Phi(\mathbf{x}) = (1, x_1^2, x_2^2) \tag{۹-۳}$$

به‌طورکلی برخی نقاط فضای $\mathcal{Z}$ ممکن است نگاشت‌های معتبری برای هیچ نقطه‌ای در

[۱] feature space

[۲] tilde

[۳] feature transform



(ب) داده‌های نگاشته شده به فضای $\mathcal{Z}$ (الف) داده در فضای $\mathcal{X}$

شکل ۳-۵: الف) داده‌های اصلی توسط یک خط قابل تفکیک نیستند اما با یک دایره می‌توان آنها را تفکیک کرد. ب) داده‌های نگاشته شده در فضای $\mathcal{Z}$ به شکل خطی قابل تفکیک هستند. دایره در فضای $\mathcal{X}$ به یک خط در فضای $\mathcal{Z}$ نگاشته می‌شود.

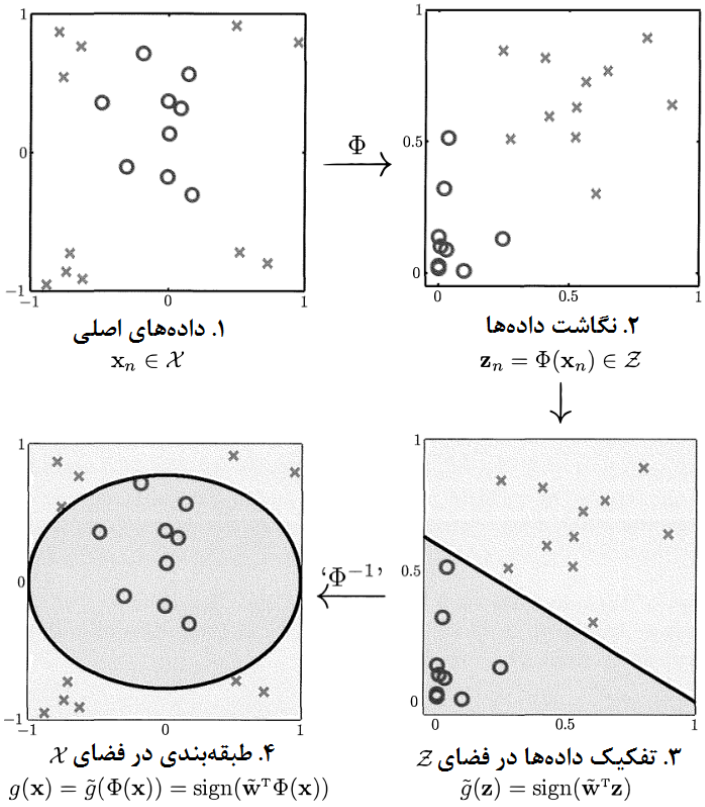
فضای $\mathcal{X}$ نباشند. همین‌طور بسته به نگاشت غیرخطی Φ، ممکن است چندین نقطه در $\mathcal{X}$ به یک نقطه $\mathbf{z}$ یکسان در $\mathcal{Z}$ نگاشته شوند.

مزیت استفاده از نگاشت بالا این است که فرضیه غیرخطی h (دایره) در فضای $\mathcal{X}$ می‌تواند با یک فرضیه خطی (خط) در فضای $\mathcal{Z}$ نمایش داده شود. درواقع هر فرضیه خطی $\tilde{h}$ بر حسب $\mathbf{z}$ متناظر با یک فرضیه (احتمالاً غیرخطی) برحسب $\mathbf{x}$ است که به‌وسیله رابطه زیر تعریف می‌شود:

$$h(\mathbf{x}) = \tilde{h}(\Phi(\mathbf{x}))$$

مجموعه این فرضیات h با $\mathcal{H}_\Phi$ نمایش داده می‌شود. برای مثال زمانی که نگاشت ویژگی در ۳-۹ استفاده می‌شود، هر $h \in \mathcal{H}_\Phi$ یک منحنی درجه‌دو در $\mathcal{X}$ است که متناظر با تعدادی خط $\tilde{h}$ در $\mathcal{Z}$ است. ازآنجاکه مجموعه داده نگاشته شده $(\mathbf{z}_1, y_1), \ldots (\mathbf{z}_N, y_N)$ در شکل ۳-۵ب (ب) به شکل خطی در فضای ویژگی $\mathcal{Z}$ تفکیک‌پذیر است، می‌توانیم الگوریتم PLA را روی این مجموعه داده اعمال کنیم تا $\tilde{\mathbf{w}}_{\text{PLA}}$ را به دست بیاوریم. این راه‌حل فرضیه نهایی $g(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}_{\text{PLA}}^{\text{T}}\mathbf{z})$ را در فضای $\mathcal{X}$ نتیجه می‌دهد که در آن $\mathbf{z} = \Phi(\mathbf{x})$ است. شکل ۳-۶ مراحل اعمال نگاشت ویژگی قبل از اجرای PLA را برای یک طبقه‌بند خطی نشان می‌دهد.

خطای درون-نمونه در فضای ورودی $\mathcal{X}$ با خطای درون-نمونه در فضای ویژگی $\mathcal{Z}$ برابر است و درنتیجه $E_{in}(g) = 0$ است. ابر صفحاتی در فضای $\mathcal{Z}$ که برای آنها $E_{in}(\tilde{\mathbf{w}}_{\text{PLA}}) = 0$ به دست می‌آید، متناظر با منحنی‌های تفکیک‌کننده داده‌ها در فضای ورودی اصلی $\mathcal{X}$ هستند.



شکل ۳-۶: نگاشت غیرخطی برای تفکیک داده‌هایی که به شکل خطی در فضای $\mathcal{X}$ تفکیک‌پذیر نیستند

برای مثال همان‌طور که در شکل ۳-۶ نشان داده‌شده است، PLA ممکن است خط $\tilde{\mathbf{w}}_{\text{PLA}} = (-0.6, 0.6, 1)$ را انتخاب کند که داده‌های نگاشته شده $(\mathbf{z}_1, y_1), \ldots (\mathbf{z}_N, y_N)$ را به شکل ایدئال تفکیک می‌کند. به همین صورت فرضیه متناظر $g(\mathbf{x}) = \text{sign}(-0.6 + 0.6 \cdot x_1^2 + x_2^2)$ نیز داده‌های اصلی $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ را به شکل ایدئال تفکیک می‌کند. در این مورد مرزهای تصمیم‌گیری به شکل یک بیضی در $\mathcal{X}$ هستند.

آیا نگاشت ویژگی روی کران VC اثر می‌گذارد؟ اگر قبل از دیدن داده‌ها در مورد انتخاب نگاشت Φ تصمیم‌گیری شود، در آن صورت با احتمال حداقل $1 - \delta$ کران ۳-۱ با استفاده از $d_{\text{vc}}(\mathcal{H}_\Phi)$ به‌عنوان بعد VC حاصل می‌شود. برای مثال تبدیل ویژگی Φ در ۳-۹ را در نظر بگیرید. می‌دانیم که $\mathcal{Z} = \{1\} \times \mathbb{R}^2$. ازآنجاکه $\mathcal{H}_\Phi$ یک پرسپترون در $\mathcal{Z}$ است، $d_{\text{vc}}(\mathcal{H}_\Phi) \leq 3$ (کوچک‌تر مساوی به‌این‌علت استفاده شده است زیرا برخی نقاط $\mathbf{z} \in \mathcal{Z}$ ممکن است نگاشت‌های

معتبری از هیچ x ای نباشند، درنتیجه برخی از دوبخشی ها ممکن است قابل تحقق نباشند). سپس می‌توان N، $d_{\text{vc}}(\mathcal{H}_\Phi)$ و δ را در کران VC جایگزین کرد. پس از اجرای PLA روی مجموعه داده نگاشته شده اگر ما موفق به دست آوردن یک یا چند g با $E_{in}(g) = 0$ بشویم، می‌توانیم ادعا کنیم که این g خارج از مجموعه هم عملکرد خوبی خواهد داشت. تأکید می‌کنیم اثبات بالا در صورتی معتبر است که شما قبل از دیدن داده یا آزمایش هر الگوریتمی در مورد انتخاب Φ تصمیم بگیرید. چه می‌شود اگر ابتدا سعی کنیم با خطها داده‌ها را تفکیک کنیم و اگر شکست خوردیم به سراغ دایره‌ها برویم. در آن صورت از مدلی استفاده می‌کنیم که به شکل مؤثر شامل هر دوی خطوط و دایره‌ها می‌شود و d_{vc} دیگر 3 باقی نخواهد ماند.

حالت بدتر این است که قبل از تصمیم‌گیری در مورد یک Φ مناسب، به داده‌ها نگاه کنید و بعد تصمیم بگیرید (مثلاً به نقطه‌داده‌های شکل ۳-۵ب نگاه کنید)، در آن صورت بیشتر آنچه در فصل ۲ یاد گرفته‌اید را عملاً زیر پا گذاشته‌اید. شما به شکل ناخودآگاه فضای فرضیات زیادی را در ذهنتان مرور می‌کنید تا به یک Φ مشخص برسید که تنها روی این مجموعه داده مشخص کار خواهد کرد. اگر بخواهیم در این حالت کران تعمیم را محاسبه کنیم، باید بعد VC تمام فضایی که در ذهنتان مرور کرده‌اید را لحاظ کنید و نه‌فقط فضایی که Φ به وجود می‌آورد.

البته این موضوع به این معنی نیست که Φ باید کورکورانه انتخاب شود. برای مثال در مسئله تعیین حد اعتبار ما برخی ویژگی‌های غیرخطی را بر اساس فیلد سابقه کار پیشنهاد دادیم که ممکن است نسبت به ورودی خام برای رگرسیون خطی مناسب‌تر باشند. این پیشنهادها بر اساس درک ما از مسئله بود و نه تجسس در داده‌های آموزشی. درنتیجه از لحاظ تعمیم، هیچ هزینه‌ای پرداخت نمی‌کنیم. در این شرایط به علت انتخاب ویژگی‌های مناسب‌تر می‌توانید به کارایی بهتری دست پیدا کنید.

همین‌طور می‌توان یک تبدیل ویژگی Φ عمومی‌تر را انتخاب کرد، البته باز به شرطی که این انتخاب قبل از دیدن داده‌ها صورت گیرد. برای مثال ممکن است توجه کرده باشید که نگاشت ویژگی در ۲-۷ تنها به ما اجازه استفاده از انواع محدودی از منحنی‌های درجه‌دو را می‌دهد. بیضی‌هایی که مرکزشان در مبدأ $\mathcal{X}$ قرار ندارد، طبق این نگاشت نمی‌توانند به هیچ ابر صفحه‌ای در $\mathcal{Z}$ متناظر شوند. برای به دست آوردن تمام منحنی‌های درجه ۲ در $\mathcal{X}$ می‌توان از یک نگاشت ویژگی عمومی‌تر $\mathsf{z} = \Phi_2(\mathsf{x})$ به شکل زیر استفاده کرد:

$$\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2) \qquad (۱۰-۳)$$

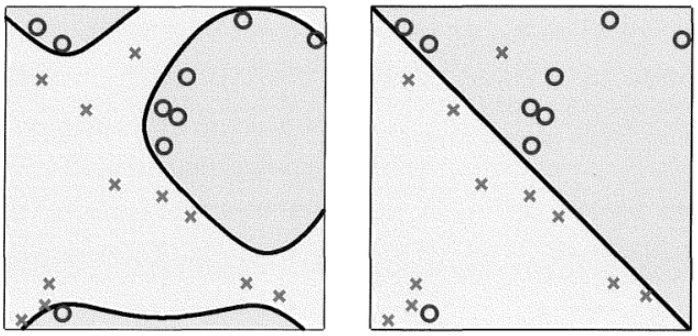
که به ما انعطاف لازم برای نمایش هر نوع منحنی درجه‌دو در $\mathcal{X}$ را با یک ابر صفحه در $\mathcal{Z}$ می‌دهد. پایین‌نویس ۲ در Φ اشاره به چندجمله‌ای درجه ۲ دارد. بهایی که باید برای این انتخاب پرداخته شود این است که اکنون $\mathcal{Z}$ به جای ۲ بعد، ۵ بعدی است و d_{vc} از ۳ به ۶ افزایش می‌یابد. به همین ترتیب می‌توان Φ_2 را به یک فضای ویژگی Φ_3 برای منحنی‌های درجه ۳ در $\mathcal{X}$ بسط داد. حتی به شکل عمومی‌تر می‌توان نگاشت ویژگی Φ_Q را برای منحنی‌های درجه Q در $\mathcal{X}$ تعریف کرد. نگاشت ویژگی Φ_Q، "نگاشت چندجمله‌ای درجه "Q خوانده می‌شود.

باید قدرت نگاشت ویژگی با احتیاط استفاده شود. ممکن است گاهی اوقات اصرار بر روی تفکیک‌پذیری خطی و استفاده از یک سطح خیلی پیچیده برای رسیدن به این هدف ارزش لازم را نداشته باشد. برای مثال مورد شکل ۳-۵ (الف) را در نظر بگیرید. اگر در اینجا ما روی یک نگاشت ویژگی که به شکل خطی کلیه داده‌ها را به درستی طبقه‌بندی کند اصرار کنیم، ممکن است این موضوع منجر به افزایش عمده‌ای در بعد VC شود. همان‌طور که در شکل ۳-۷ مشاهده می‌شود هیچ خطی قادر نیست مثال‌های آموزشی را به شکل ایدئال تفکیک کند. به همین ترتیب هیچ منحنی درجه‌دو یا درجه سه‌ای نیز نمی‌تواند به این هدف دست یابد. درنتیجه نیاز به استفاده از یک چندجمله‌ای درجه چهار خواهیم داشت:

$$\Phi_g(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4)$$

اگر به مرزهای تصمیم ایجادشده توسط این نگاشت در شکل ۳-۷ (ب) نگاهی بیندازید، برای اینکه متوجه شوید که این یک برازش افراطی است و بسیار بعید است که به‌خوبی به داده‌های جدید تعمیم پیدا کند، احتیاج به تحلیل VC ندارید. راه بهتر این است که از دو مثال بدطبقه‌بندی‌شده در شکل ۳-۷ (الف) چشم‌پوشی کنیم و بقیه داده‌ها را به شکل ایدئال با یک خط تفکیک کنیم و خطای غیر صفر اما کوچک E_{in} را بپذیریم. در حقیقت گاهی بهترین شانس ما این است که یک فرضیه ساده‌تر را قبول کرده و خطای E_{in} کوچک ناشی از آن را تحمل کنیم.

درحالی‌که در بحث تبدیل ویژگی تا کنون روی مسئله طبقه‌بندی متمرکز شدیم، این نگاشت



(الف) برازش خطی (ب) برازش چندجمله‌ای درجه ۴

شکل ۳-۷: نگاشت غیرخطی برای مجموعه داده‌ای که به شکل خطی تفکیک پذیر نیست. الف) با حذف چند نقطه‌داده نویزی داده‌ها تفکیک می‌شوند ب) یک چندجمله‌ای درجه ۴ که کلیه نقاط را تفکیک می‌کند.

می‌تواند به‌طور مشابه روی مسائل رگرسیون نیز اعمال شود. می‌توان هر دوی رگرسیون‌های خطی و لجستیک را به جای فضای ورودی $\mathcal{X}$، در فضای ویژگی $\mathcal{Z}$ به‌کار برد. برای مثال رگرسیون خطی اغلب با یک نگاشت ویژگی برای انجام رگرسیون غیرخطی همراه است. در این حالت، ماتریس ورودی X با ابعاد $N \times d+1$ با ماتریس $\mathcal{Z}$ با ابعاد $N \times \tilde{d}+1$ جایگزین می‌شود، درحالی‌که بردار خروجی y همچنان همان باقی می‌ماند.

۳-۴-۲ محاسبه و تعمیم

هرچند استفاده از یک Q بزرگ‌تر انعطاف بیشتری را در شکل مرزهای تصمیم در $\mathcal{X}$ ایجاد می‌کند. اما این کار هزینه‌ای نیز دارد. افزایش محاسبات و کاهش تعمیم از پیامدهای این عمل هستند.

محاسبات یک پیامد است زیرا نگاشت ویژگی Φ_Q یک بردار دوبعدی x را به $\tilde{d} = \frac{Q(Q+3)}{2}$ بعد نگاشت می‌دهد که این امر حافظه و هزینه‌های محاسباتی را افزایش می‌دهد. اگر $\mathcal{X}$ دارای ابعاد بالایی باشد، این پیامد جدی خواهد بود.

پیامد مهم دیگر تعمیم است. اگر Φ_Q نگاشت ویژگی از یک فضای ورودی دوبعدی باشد، ابعاد فضای $\mathcal{Z}$ برابر با $\tilde{d} = \frac{Q(Q+3)}{2}$ خواهد بود و $d_{\text{vc}}(\mathcal{H}_\Phi)$ می‌تواند به اندازه $\frac{Q(Q+3)}{2}+1$ بزرگ باشد. این به این معنی است که ترم دوم در کران VC در ۳-۱ می‌تواند به‌شدت رشد می‌کند. به‌عبارت‌دیگر در این حالت ما تضمین ضعیف‌تری برای کوچک بودن E_{out} خواهیم

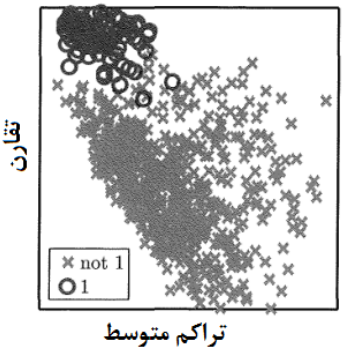
داشت. برای مثال اگر از $\Phi_5 0$ استفاده کنیم، $d_{\text{vc}}(\mathcal{H}_\Phi)$ به جای $d_{\text{vc}} = 3$ به $(50 * 53/2) + 1 =$ 1326 افزایش می‌یابد. با توجه به این قانون سرانگشتی که مقدار داده مورد نیاز متناسب با بعد VC است، برای رسیدن به خطای تعمیمی در حد حالتی که از نگاشت ویژگی استفاده نمی‌شود، به صدها برابر داده بیشتر احتیاج خواهیم داشت.

گاهی اوقات زمانی که به یک فضای با ابعاد بالاتر می‌رویم مشکل تعمیم با مزیتی که در تقریب بهتر تابع هدف به دست می‌آید متعادل می‌شود. برای مثال همان‌طور که در زمان استفاده از منحنی‌های درجه‌دو به جای خطوط مشاهده شد، پس از این عمل داده‌های نگاشته شده به شکل خطی تفکیک‌پذیر شدند و E_{in} به صفر کاهش پیدا کرد. به‌طورکلی زمانی که قرار است بعد مناسبی را برای نگاشت ویژگی انتخاب کنیم باید موازنه تقریب-تعمیم را در نظر داشته باشیم.

$\tilde{d}$ بزرگتر	شانس بیشتر برای تفکیک پذیری خطی	$E_{in} \downarrow$	$d_{\text{vc}} \uparrow$
$\tilde{d}$ کوچکتر	احتمال بیشتر عدم تفکیک پذیری خطی	$E_{in} \uparrow$	$d_{\text{vc}} \downarrow$

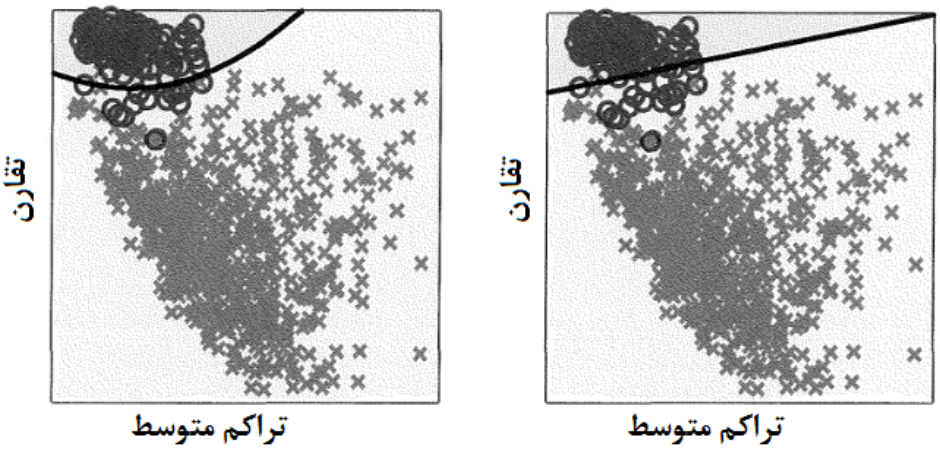
با توجه به این مباحث انتخاب یک نگاشت ویژگی مناسب قبل از دیدن داده‌ها عمل دشواری است. زمانی که قرار است یادگیری را به یک مسئله خاص اعمال کنیم، درک برخی از ابعاد مسئله می‌تواند به انتخاب ویژگی‌هایی مناسب کمک کند. در حالت عمومی‌تر، برخی روش‌های راهنما برای انتخاب یک نگاشت یا یک مدل مناسب وجود دارد که ما در فصل چهار در این مورد بحث خواهیم کرد.

مثال ۲-۳: اجازه دهید یک بار دیگر مثال تشخیص ارقام دست‌نویس را مرور کنیم. همانطور که گفته شد می‌توان تکلیف سنگین تشخیص ده رقم از یکدیگر را به تکالیف کوچک‌تری تجزیه کرد. یک تجزیه موردی این است که سعی کنیم رقم 1 را از بقیه رقم‌ها تفکیک کنیم. گراف پراکندگی بر اساس ویژگی‌های تراکم و تقارن به‌عنوان متغیرهای ورودی در شکل بعد نشان داده‌شده است.



مشاهده می‌شود که تقریباً با یک خط می‌توان رقم 1 را از بقیه رقم‌ها جدا کرد، اما یک منحنی پیچیده‌تر ممکن است بهتر عمل کند.

ما ابتدا از رگرسیون خطی بدون استفاده از هیچ نگاشت ویژگی برای طبقه‌بندی استفاده می‌کنیم. نتیجه این عمل در شکل زیر سمت چپ نشان داده‌شده است. در این حالت مقادیر $E_{in} = 2.13\%$ و $E_{out} = 2.38\%$ را به دست می‌آوریم.



(ب) مدل چندجمله‌ای درجه ۳ (الف) مدل خطی

طبقه‌بندی رقم 1 در مقابل غیر 1 با استفاده از مدل‌های خطی و چندجمله‌ای درجه ۳

زمانی که از رگرسیون خطی با Φ_3 به‌عنوان نگاشت چندجمله‌ای درجه سه استفاده کنیم، برازش بهتری از داده‌ها با E_{in} پایین‌تر برابر با 1.75% را به دست می‌آوریم. این نتیجه در

سمت راست شکل نمایش داده‌شده است. در این حالت برازش درون-نمونه بهتر باعث کارایی خارج-از-نمونه بهتری نیز شده است با $E_{out} = 1.87\%$.

مدل‌های خطی، مرور نهایی: مدل خطی (برای طبقه‌بندی یا رگرسیون) یک ابزار پرکاربرد در حوزه یادگیری ماشین است. الگوریتم‌های کارایی برای یادگیری به‌وسیله مدل‌های خطی وجود دارد. این الگوریتم‌ها سربار کمی داشته و همین‌طور باثبات بوده و دارای ویژگی‌های تعمیم خوبی هستند. زمانی که قرار است از یادگیری ماشین استفاده کنید، یک قانون مفید این است که ابتدا با یک مدل خطی شروع کنید. به دلیل ویژگی‌های خوب مدل‌های خطی مطمئن هستید که چیز زیادی به اشتباه نمی‌رود. چنانچه برازش خوبی از داده‌ها به دست آوردید (خطای درون-نمونه پایین) درنتیجه به هدف رسیده‌اید، اما درصورتی‌که برازش خوبی از داده‌ها حاصل نشد و قصد داشتید به مدل پیچیده‌تری رجوع کنید، باید بهایی را از لحاظ بعد VC بالاتر بپردازید، اما این بها، بهای سنگینی نیست.

فصل چهارم

بیش‌برازش

ترس از نحسی روز سیزده و یا دیگر خرافات مشابه شاید بارزترین نمونه‌های گرایش انسان‌ها به بیش‌برازش هستند. حوادث تلخ معمولاً به یاد می‌مانند و در صورت بروز تعداد کمی از چنین حوادثی، طبیعی است که افراد سعی کنند توضیحی برای آنها پیدا کنند. آیا در آینده واقعاً حوادث ناگوار بیشتری در روز سیزده نسبت به بقیه روزها رخ خواهد داد؟

بیش‌برازش مفهومی است که در آن انطباق بیش‌ازحد با وقایع (داده‌ها) مشاهده‌شده، دیگر نشان‌دهنده یک خطای خارج-از-نمونه خوب نیست و ممکن است حتی به یک نتیجه عکس منتهی شود. شما احتمالاً مواردی از بیش‌برازش را زمانی که مدل به‌کارگرفته‌شده بسیار پیچیده‌تر از آنچه برای نمایش تابع هدف مورد نیاز است را مشاهده کرده باشید. چنین مدلی از درجه آزادی اضافی خود برای برازش ناهنجاری در داده‌ها (برای مثال نویز) استفاده می‌کند و این امر منجر به یک فرضیه نهایی نامطلوب می‌شود. بیش‌برازش حتی زمانی که مجموعه فرضیات حاوی توابعی به‌مراتب ساده‌تر از تابع هدف است نیز مشاهده می‌شود.

توانایی در مواجهه با بیش‌برازش چیزی است که حرفه‌ای‌ها را از غیر حرفه‌ای‌ها در حوزه یادگیری ماشین جدا می‌کند. در این فصل ما سه مسئله را پوشش می‌دهیم:

۱. چه زمانی بیش‌برازش رخ می‌دهد؟

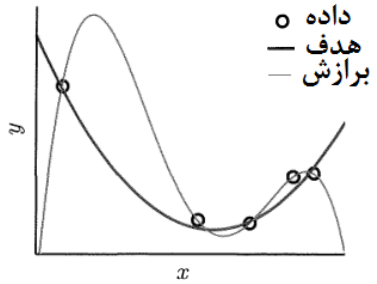
۲. چه ابزارهایی برای مقابله با بیش‌برازش وجود دارد؟

۳. چگونه یک شخص می‌تواند درجه بیش‌برازش را تخمین زده یا "گواهی دهد" که یک مدل خوب است یا از دیگری بهتر است؟

تأکید ما بر روی تکنیک‌هایی است که در عمل خوب جواب داده‌اند.

۴-۱ چه زمانی بیش‌برازش رخ می‌دهد؟

معنی بیش‌برازش از لحاظ لغوی عبارت است از "برازش داده بیش ازآنچه لازم است". بارزترین نمونه بیش‌برازش زمانی رخ می‌دهد که شما فرضیه‌ای با E_{in} پایین‌تر را انتخاب می‌کنید، اما نتیجه E_{out} بالاتری است. این موضوع نشان می‌دهد که E_{in} دیگر به تنهایی یک راهنمای خوب برای یادگیری نیست. اجازه دهید بحث را با شناسایی عوامل بیش‌برازش شروع کنیم.



برازش یک هدف چند جمله‌ای درجه ۲ نویزی با یک مدل چندجمله‌ای درجه ۴

یک مسئله رگرسیون یک‌بعدی با پنج نقطه‌داده در نظر بگیرید. تابع هدف ناشناخته است و بنابراین ما یک مدل عمومی انتخاب می‌کنیم تا شانس پوشش تابع هدف را بیشتر کنیم. ازآنجاکه هر ۵ نقطه‌ای با یک چندجمله‌ای درجه ۴ قابل برازش کامل هستند، ما چندجمله‌ای درجه ۴ را به این منظور انتخاب می‌کنیم.

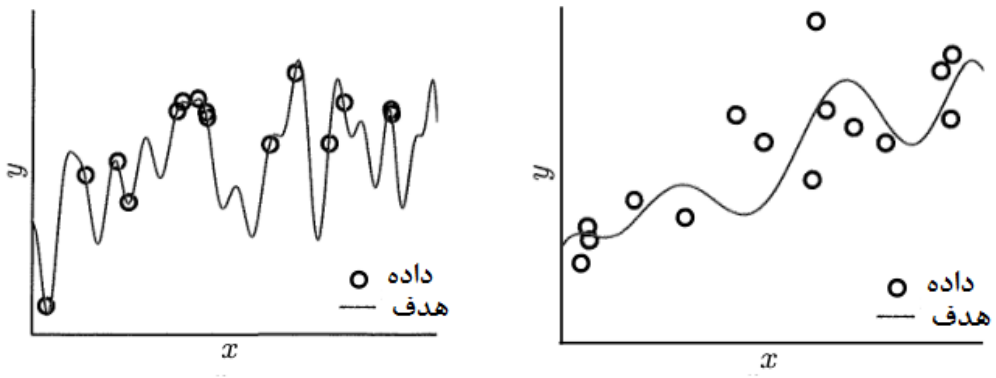
نتیجه برازش در شکل بالا نمایش داده‌شده است. تابع هدف از درجه ۲ است (نمودار سهمی شکل) که دارای مقدار کمی نویز در مجموعه داده است. هرچند که تابع هدف تابع ساده‌ای است، اما الگوریتم یادگیری از تمام توان چندجمله‌ای درجه ۴ برای برازش دقیق کلیه نقاط استفاده کرده است، اما نتیجه هیچ شباهتی به تابع هدف ندارد. در اینجا داده‌ها بیش‌برازش شده‌اند. وجود مقادیر کمی نویز در داده‌ها یادگیری را منحرف کرده است؛ درحالی‌که اگر نویز وجود نداشت، برازش به‌شکل کامل با هدف منطبق می‌شد. این یک سناریوی معمول برای بیش‌برازش است که در آن یک مدل پیچیده از درجه آزادی بیشتر خود برای یادگیری "نویز" استفاده می‌کند.

در مثال بالا، برازش دارای خطای درون-نمونه صفر اما خطای خارج-از-نمونه بالا است،

بنابراین این حالت موردی از تعمیم بد است که در فصل ۲ به آن اشاره شد و اتفاقی است که اغلب در زمان بیش‌برازش رخ می‌دهد. با این وجود، تعریف ما از بیش‌برازش فراتر از تعمیم بد برای یک فرضیه داده‌شده است. از نظر ما، بیش‌برازش در یک فرایند اتفاق می‌افتد: در اینجا فرایند انتخاب یک فرضیه با E_{in} کم و کمتر است که منجر به E_{out} بیشتر و بیشتر می‌شود.

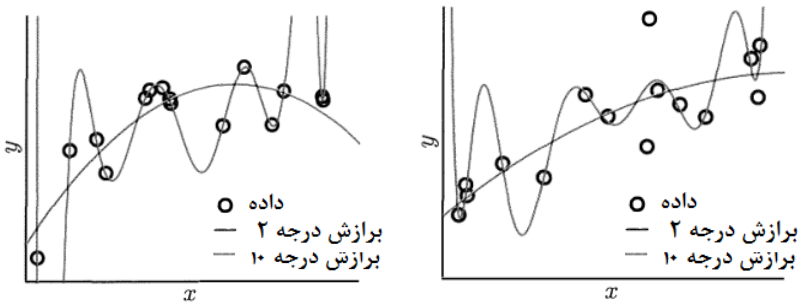
۴-۱-۱ مطالعه موردی: بیش‌برازش با چندجمله‌ای‌ها

برای اینکه درک کنیم بیش‌برازش چه زمانی رخ می‌دهد، اجازه دهید بررسی عمیق‌تری انجام دهیم. ما سعی می‌کنیم مفاهیم اصلی را با استفاده از رگرسیون چندجمله‌ای یک‌بعدی نشان دهیم. این نوع رگرسیون مورد خاصی از مدل خطی است که از نگاشت ویژگی $x \mapsto (1, x, x^2, \dots)$ استفاده می‌کند.



(الف) تابع هدف درجه ۱۰ (ب) تابع هدف درجه ۵۰

دو مسئله رگرسیون در شکل بالا را در نظر بگیرید. در هر دو مسئله تابع هدف یک چندجمله‌ای و مجموعه داده $\mathcal{D}$ شامل ۱۵ نقطه است. در (الف) تابع هدف یک چندجمله‌ای درجه ۱۰ است و داده‌های نمونه نویزی هستند. بنابراین همه داده‌ها روی منحنی تابع هدف قرار نمی‌گیرند. در قسمت (ب) تابع هدف یک چندجمله‌ای درجه ۵۰ است و داده‌ها بدون نویز هستند. بهترین برازش درجه ۲ و ۱۰ در شکل ۴-۱ نمایش داده‌شده‌اند و خطای درون-نمونه و خارج-از-نمونه در جدول زیر آورده شده‌اند.



(ب) هدف بدون نویز با درجه بالا (الف) هدف نویزی با درجه پایین

شکل ۴-۱: برازش‌های چندجمله‌ای درجه ۲ و درجه ۱۰ روی ۱۵ نقطه‌داده. در (الف) داده‌ها نویزی هستند و هدف یک چندجمله‌ای درجه ۱۰ است. در (ب) داده‌ها بدون نویز هستند و هدف چندجمله‌ای درجه ۵۰ است.

	درجه ۲	درجه ۱۰
E_{in}	0.050	0.034
E_{out}	0.127	9.00

(الف) تابع هدف نویزی درجه ۱۰

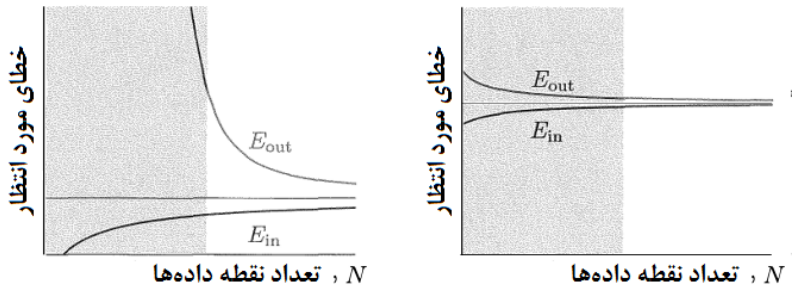
	درجه ۲	درجه ۱۰
E_{in}	0.029	10^{-5}
E_{out}	0.120	7680

(ب) تابع هدف بدون نویز درجه ۵۰

در هر دو مورد چندجمله‌ای درجه ۱۰ داده‌ها را به‌شدت بیش‌برازش کرده است و منجر به تولید یک فرضیه نهایی ناملموس شده است که هیچ شباهتی به تابع هدف ندارد. در هیچ‌کدام از این موارد، برازش‌های درجه ۲ ذات تابع هدف را آشکار نمی‌کنند، اما حداقل سمت و سوی کلی آن را اخذ می‌کنند و درنتیجه دارای خطای خارج-از-نمونه بسیار کمتری هستند.

چیزی که الگوریتم یادگیری می‌بیند داده است نه تابع هدف. برازش‌های درجه ۱۰ دارای خطای درون-نمونه کوچک و همزمان خطای خارج-از-نمونه بسیار بزرگی هستند. این امر نشان‌دهنده موردی از بیش‌برازش است که منجر به یک تعمیم بد می‌شود. این دو مثال نکات جالب‌توجهی را آشکار می‌کنند. ابتدا تابع هدف درجه ۱۰ را بررسی می‌کنیم. سناریو به این قرار است. دو یادگیرنده O (برای بیش‌برازش) و R (برای محدودشده) هر دو اطلاع دارند که تابع هدف یک چندجمله‌ای درجه ۱۰ است و ۱۵ نقطه‌داده نویزی را در اختیار خواهند داشت. یادگیرنده O از مدل $\mathcal{H}_{10}$ استفاده می‌کند، با این توجیه که این مدل حاوی تابع هدف است، و بهترین فرضیه در این مدل که با داده‌ها منطبق می‌شود را پیدا می‌کند. یادگیرنده R از مدل $\mathcal{H}_2$ استفاده می‌کند و به‌طور مشابه بهترین فرضیه‌ای که با داده‌ها منطبق می‌شود را پیدا می‌کند.

نکته عجیب این است که یادگیرنده R با استفاده از یک مدل ساده‌تر به خطای خارج-از-



(ب) منحنی یادگیری برای $\mathcal{H}_{10}$ (الف) منحنی یادگیری برای $\mathcal{H}_2$

شکل ۴-۲: بیش‌برازش برای مقادیر N در ناحیه خاکستری رنگ اتفاق می‌افتد و با انتخاب $\mathcal{H}_{10}$ که دارای E_{in} پایین‌تری است، E_{out} بدتری حاصل می‌شود.

نمونه کمتری دست می‌یابد و بااینکه از عدم توانایی خود در پیاده‌سازی کامل تابع هدف آگاه است، برنده می‌شود. یادگیرنده R خطای درون-نمونه بد را در ازای کسب سود بالا در خطای تعمیم معاوضه می‌کند و نهایتاً به خطای خارج-از-نمونه پایین‌تری می‌رسد. چه چیزی در اینجا عجیب است؟ یک باور عمومی در مورد یادگیری می‌گوید که نتایج بهتر با بهره گرفتن از کلیه اطلاعات موجود در مورد هدف به دست می‌آیند. اما همان‌طور که در اینجا دیدیم حتی اگر ما درجه تابع هدف را بدانیم و با انتخاب مدل متناظر $\mathcal{H}_{10}$ به شکل ساده‌لوحانه سعی کنیم از این اطلاع بهره ببریم، کارایی به‌دست‌آمده خیلی بدتر از آن چیزی است که توسط یک مدل درجه ۲ باثبات‌تر حاصل می‌شود.

منحنی‌های یادگیری متناظر با مدل‌های $\mathcal{H}_2$ و $\mathcal{H}_{10}$ در شکل ۴-۲ نمایش داده‌شده‌اند. اگر به شکل ذهنی این دو نمودار را با هم مقایسه کنید، متوجه می‌شوید که بازه‌ای از N وجود دارد که در آن $\mathcal{H}_{10}$ دارای خطای درون-نمونه کمتر اما خطای خارج-از-نمونه بیشتری نسبت به $\mathcal{H}_2$ است و این همان جایی است که احتمال رخ دادن بیش‌برازش زیاد است.

آیا یادگیرنده R همیشه برنده است؟ مطمئناً خیر، برای مثال اگر داده‌ها بدون نویز بودند، در آن صورت یادگیرنده O با استفاده از ۱۵ نقطه داده شده، تابع هدف را به‌طور کامل بازیابی می‌کرد، درحالی‌که یادگیرنده R امیدی به این کار نداشت. این قضیه ما را به مثال دوم می‌رساند (شکل ۴-۲ب ب) در اینجا داده‌ها بدون نویز هستند، اما تابع هدف خیلی پیچیده است (چندجمله‌ای درجه ۵۰). در این مورد نیز یادگیرنده R برنده می‌شود و باز به همان دلیل که یادگیرنده O داده‌ها را به‌شدت بیش‌برازش کرده است. بیش‌برازش تنها مختص مدل‌های

پیچیده‌ای که سعی در برازش توابع هدف ساده‌تر دارند نیست. درواقع ما در اینجا شاهد عکس این مطلب بودیم و بیش‌برازش به همان شدت مخرب است. نکته‌ای که اهمیت دارد این است که «باید ببینیم پیچیدگی مدل چگونه ممکن است خود را با کمیت و کیفیت داده‌های موجود تطبیق دهد نه اینکه چگونه با تابع هدف منطبق می‌شود».

به‌طور خلاصه بیش‌برازش با افزایش نویز و همین‌طور افزایش پیچیدگی تابع هدف افزایش پیدا می‌کند، و با افزایش نقطه‌داده‌ها کاهش می‌یابد. در ادامه با ارائه یک مفهوم متناظر برای پیچیدگی مدل سعی می‌کنیم این روابط را ساده‌تر کنیم.

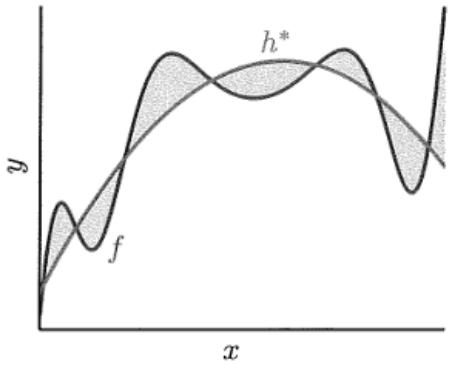
تعداد نقطه‌داده‌ها	↑	بیش‌برازش	↓
نویز	↑	بیش‌برازش	↑
پیچیدگی هدف	↑	بیش‌برازش	↑

نویز قطعی: چرا زمانی که از یک مدل یکسان استفاده می‌کنیم، تابع هدف پیچیده‌تر (درجه ۵۰ در مقایسه با درجه ۱۰ در مثال) منجر به بیش‌برازش بیشتر می‌شود؟ ایده کلی به این شکل است که برای یک مدل یادگیری داده‌شده، یک بهترین تقریب برای تابع هدف وجود دارد که دارای کمترین خطا است. بخشی از تابع هدف که خارج از این بهترین تقریب قرار می‌گیرد مانند نویز در داده عمل می‌کند. ما این نقاط از تابع هدف را "نویز قطعی" [۱] می‌نامیم تا این مفهوم را از "نویز تصادفی" [۲] متمایز کنیم. همانند نویز تصادفی که نمی‌توان آن را مدل کرد، نویز قطعی نیز بخشی از تابع هدف است که مدل نمی‌شود. الگوریتم یادگیری نباید سعی کند نویز را برازش کند، اما نمی‌تواند نویز را از سیگنال تشخیص دهد. بر روی یک مجموعه داده متناهی، الگوریتم ناآگاهانه از درجه‌های آزادی خود برای برازش نویز استفاده می‌کند و این منجر به بیش‌برازش و یک فرضیه نهایی ناکارآمد می‌شود.

شکل ۴-۳ یک مدل درجه‌دو را نشان می‌دهد که یک تابع هدف بسیار پیچیده‌تر از خود را برازش می‌کند. قسمت‌های از تابع که بیرون از این برازش قرار می‌گیرند نویز قطعی محسوب می‌شوند. درحالی‌که نویز قطعی و نویز تصادفی تأثیر مشابهی روی بیش پردازش دارند، دو تفاوت اساسی میان آنها وجود دارد. اولین تفاوت این است که اگر ما همان داده‌ها (مقادیر x)

[۱] deterministic noise

[۲] stochastic noise



شکل ۴-۳: نویز قطعی، h^* بهترین برازش برای f در $\mathcal{H}_2$ است. نواحی خاکستری نویز قطعی را برای این مسئله یادگیری نشان می‌دهند.

را دوباره تولید کنیم، نویز قطعی تغییری نمی‌کند اما نویز تصادفی تغییر می‌کند. تفاوت دوم این است که مدل‌های مختلف بخش‌های متفاوتی از تابع هدف را اخذ می‌کنند؛ درنتیجه بسته به اینکه از چه مدلی استفاده کنیم، یک مجموعه داده یکسان نویزهای قطعی متفاوتی خواهد داشت. در عمل ما در یک زمان با یک مدل کار می‌کنیم و تنها یک مجموعه داده در اختیار داریم؛ بنابراین تنها با یک تحقق خاص از نویز مواجه هستیم و الگوریتم قادر نیست میان این دو نوع از نویز تمایزی قائل شود.

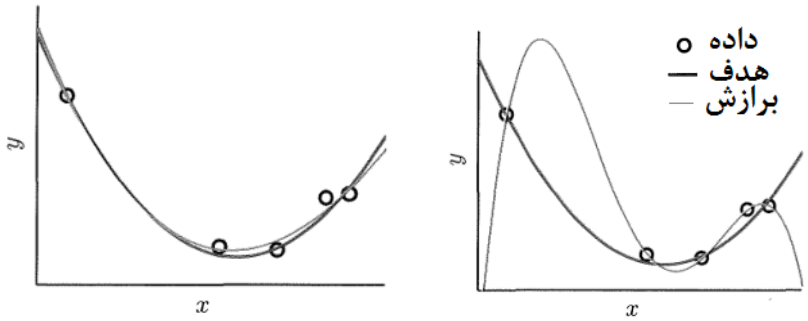
تجزیه بایاس-واریانس که قبلاً در مورد آن بحث کردیم، ابزار مفیدی است برای درک چگونگی تأثیر نویز بر کارایی است.

$$\mathbb{E}_{\mathcal{D}}[E_{out}] = \sigma^2 + \text{bias} + \text{var}$$

دو ترم اول تحت تأثیر مستقیم نویز قطعی و تصادفی هستند. ترم اول σ^2 نشانگر واریانس نویز تصادفی است. بایاس نیز مستقیماً تحت تأثیر نویز قطعی است به این معنی که ناتوانی مدل در تقریب f را نمایش می‌دهد. ترم var به شکل غیرمستقیم تحت تأثیر هر دو نوع نویز است و آسیب‌پذیری مدل در به انحراف کشیده شدن توسط نویز را نشان می‌دهد.

۴-۲ منظم‌سازی

منظم‌سازی اولین سلاح ما برای مبارزه با بیش‌برازش است. این روش، به‌ویژه زمانی که نویز وجود دارد، محدودیت‌هایی را در جهت بهبود خطای خارج-از-نمونه برای الگوریتم یادگیری به وجود می‌آورد.



(الف) بدون منظم‌سازی (ب) با منظم‌سازی

برای اینکه متوجه قدرت این روش شوید، نگاهی به شکل زیر بیندازید که نتیجه منظم‌سازی را بر روی مثال بیش‌برازش که در قسمت ۴-۲ مرور شد نشان می‌دهد. هرچند تنها از میزان کمی منظم‌سازی استفاده شده است، برازش به شکل قابل‌توجهی بهبودیافته است. منظم‌سازی بیشتر یک هنر است تا یک علم. در عمل بیشتر روش‌هایی که با موفقیت استفاده می‌شوند روش‌های اکتشافی[۱] هستند، هرچند ممکن است پایه‌ای در یک چارچوب ریاضی که برای آن موارد خاص توسعه‌یافته است داشته باشند. ما هر دو وجه اکتشافی و ریاضی این روش‌ها را مورد بحث قرار می‌دهیم و سعی می‌کنیم تعادلی را که بازتاب‌دهنده واقعیت این حوزه است را حفظ کنیم.

اگر بخواهیم با دید اکتشافی به مسئله نگاه کنیم، یک نگاه به منظم‌سازی از منظر کران VC است که کرانی را برای E_{out} با استفاده از ترم جریمه پیچیدگی مدل $\Omega(\mathcal{H})$ ایجاد می‌کند:

$$E_{out} \leq E_{in} + \Omega(\mathcal{H}) \quad \text{for all } h \in \mathcal{H} \tag{۱-۴}$$

با توجه به این ترم جریمه، بهتر است ما داده‌ها را با استفاده از یک $\mathcal{H}$ ساده برازش کنیم. اگر بخواهیم یک گام جلوتر برویم بهتر است داده‌ها را با استفاده از یک h ساده از مجموعه $\mathcal{H}$ برازش کنیم. اساس روش منظم‌سازی این است که یک مقیاس $\Omega(h)$ را برای اندازه‌گیری

[۱] heuristics

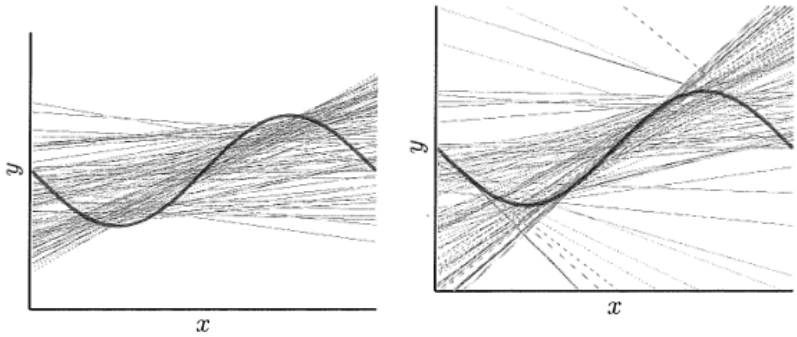
پیچیدگی یک فرضیه خاص توسعه دهد. به جای اینکه $E_{in}(h)$ را به‌تنهایی کاهش دهیم می‌توانیم ترکیبی از $E_{in}(h)$ و $\Omega(h)$ را کاهش دهیم. این عمل با محدود کردن الگوریتم یادگیری به استفاده از یک فرضیه ساده برای برازش داده‌ها مانع بیش‌برازش می‌شود. یکی از تکنیک‌های شناخته‌شده برای منظم‌سازی روش "زوال وزن"[۱] است که پیچیدگی فرضیه h را با توجه به مقدار ضرایبی که برای نمایش h در یک مدل خطی استفاده می‌شود اندازه می‌گیرد. این روش اکتشافی خط‌های هموار با انحراف و انحناهای کم را به خط‌های تند با انحراف و انحناهای زیاد ترجیح می‌دهد. به زودی به توضیح بیشتر نحوه کارکرد این روش باز خواهیم گشت، اما در اینجا اجازه دهید نگاهی به اعمال این روش به مسئله رگرسیون تابع $f(\mathbf{x}) = \sin(\pi x)$ با استفاده از دو نقطه که قبلاً آن را نشان دادیم بیندازیم.

به این منظور، ابتدا از $\mathbf{x}$ به شکل یکنواخت در بازه $[-1, +1]$ نمونه‌برداری می‌کنیم و به این ترتیب یک مجموعه داده تولید می‌کنیم. سپس این داده‌ها را توسط مدل $\mathcal{H}_1$ با یک خط مستقیم برازش می‌کنیم. شکل زیر نتیجه این برازش با استفاده از تعداد زیادی مجموعه داده‌های تصادفی یکسان را در دو حالت بدون منظم‌سازی و با منظم‌سازی نشان می‌دهد. در حالت بدون منظم‌سازی، تابع یادگیری شده نسبت به تغییر مجموعه داده حساس است و به‌شدت تغییر می‌کند. در این حالت همان‌طور که قبلاً در مثال ۲-۳ دیدیم، حتی یک مدل ثابت می‌تواند بهتر از این مدل عمل کند. اما با استفاده از روش زوال وزن، برازش‌ها روی همان مجموعه داده‌ها تغییرات به‌مراتب کمتری دارند و این امر باعث ایجاد E_{out} کمتری نسبت به دو حالت دیگر می‌شود (برابر با 0.56 در مقایسه با 1.9 برای تابع خطی و 0.75 برای تابع ثابت قبل).

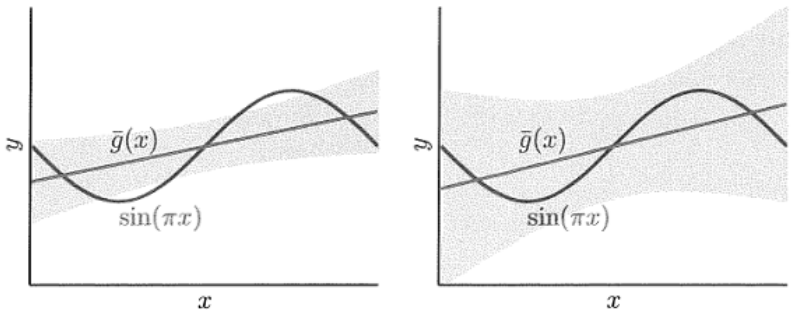
تجزیه بایاس-واریانس می‌تواند در درک اینکه چرا نسخه 'با منظم‌سازی' از هر دو مدل قبلی بهتر عمل کرد به ما کمک کند. همان‌طور که انتظار می‌رفت، منظم‌سازی ترم واریانس را بسیار کاهش داده است(از 1.69 به 0.33). هزینه این کاهش خود را در افزایش ترم بایاس (برازش متوسط) نشان می‌دهد، اما این افزایش بسیار جزئی است و بایاس تنها به اندازه کمی از 0.21 به 0.23 افزایش پیدا می‌کند. نتیجه نهایی کاهش چشمگیر خطای خارج-از-نمونه است که از جمع بایاس و واریانس حاصل می‌شود. این خاصیت درواقع نکته کلیدی منظم‌سازی است. به این معنی که با محدود کردن الگوریتم یادگیری به انتخاب فرضیات ساده در $\mathcal{H}$، افزایش کوچکی در بایاس را متحمل می‌شویم تا به یک کاهش قابل‌توجه در بُعد واریانس دست پیدا کنیم.

---

[۱] weight decay



(ب) با منظم‌سازی (الف) بدون منظم‌سازی



bias $= 0.23$; var $= 0.33$. bias $= 0.21$; var $= 1.69$.

(ب) با منظم‌سازی (الف) بدون منظم‌سازی

$\bar{g}$ به شکل خط ممتد و $\mathsf{var}(x)$ با نواحی تیره رنگ با اندازه $\bar{g}(\mathsf{x}) \pm \sqrt{\mathsf{var}(x)}$ نشان داده شده است.

این مثال نشان می‌دهد که چرا منظم‌سازی در اینجا مورد نیاز بود. مدل خطی برای مقدار داده کمی که در اختیار داشتیم خیلی پیچیده بود، زیرا هر دو نقطه را می‌توان با یک خط به شکل ایدئال برازش کرد. در این حالت حتی اگر تابع هدف، تابع دیگری بود، تا زمانی که نویز قطعی یا تصادفی وجود داشت این ضرورت وجود داشت. در حقیقت نیاز به منظم‌سازی تابع کمیت و کیفیت داده‌های موجود است. زمانی که آن مجموعه داده ناچیز به ما داده شد، دو انتخاب در پیش رو داشتیم، یا باید مدل ساده‌تری مانند مدلی حاوی توابع ثابت را انتخاب می‌کردیم و یا اینکه مدل خطی را محدود می‌کردیم. نهایتاً به این نتیجه رسیدیم که از مدل خطی که مدل پیچیده‌تری است استفاده کنیم، اما الگوریتم را به انتخاب فرضیات ساده‌تر محدود کنیم. این

حالت نسبت به مدل ثابت انعطاف بیشتری در برازش داده‌ها فراهم می‌کند و نهایتاً E_{out} بهتری حاصل می‌شود. این مطلب یک قاعده کلی است و تنها یک استثناء نیست.

۴-۲-۱ روش زوال وزن

در این قسمت با معرفی یک روش شناخته‌شده منظم‌سازی، نشان خواهیم داد که چگونه می‌توان الگوریتم یادگیری را به انتخاب فرضیات ساده‌تر سوق داد. به این منظور ابتدا کمیت جدیدی با عنوان خطای افزوده[۱] را به شکل زیر تعریف می‌کنیم:

$$E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda \mathbf{w}^{\mathrm{T}}\mathbf{w} \qquad (۴-۲)$$

که در آن $\lambda \geq 0$ یک پارامتر آزاد است. این خطا شامل دو ترم است. اولین ترم خطای درون-نمونه است که تاکنون هدف اصلی کمینه‌سازی محسوب می‌شد و ترم دوم یک ترم جریمه است. همانطور که ملاحظه می‌شود این تعریف با نگاه اکتشافی به منظم‌سازی که قبلاً در مورد آن بحث کردیم منطبق است. با این تفاوت جزئی که در آن جریمه پیچیدگی به جای اینکه برای مجموعه کامل فرضیات $\mathcal{H}$ تعریف شود، برای هر h منفرد تعریف می‌شود.

زمانی که $\lambda = 0$ است، تنها با همان خطای درون-نمونه معمول روبرو هستیم. برای $\lambda > 0$، کمینه‌سازی خطای افزوده معادل با کمینه‌سازی یک خطای درون-نمونه جریمه‌شده است. پارامتر λ درجه منظم‌سازی را کنترل می‌کند. ترم جریمه $\mathbf{w}^{\mathrm{T}}\mathbf{w}$ موازنه‌ای میان کوچک کردن خطای درون-نمونه و کوچک کردن وزن‌ها برقرار می‌سازد و به‌عنوان "زوال وزن" شناخته می‌شود. اگر این خطای افزوده را با استفاده از یک روش تکراری مانند گرادیان نزولی کمینه کنیم، شاهد کاهش خطای درون-نمونه به همراه کاهش تدریجی وزن‌ها خواهیم بود و اصطلاح "زوال وزن" از این خاصیت گرفته شده است.

در تعریف قبلی خطای افزوده در قالب معادله ۴-۲، ما تنها وابستگی به $\mathbf{w}$ را برجسته کردیم. دو کمیت دیگر نیز وجود دارند که قابل‌کنترل هستند و عبارت‌اند از درجه منظم‌سازی λ و نوع منظم‌سازی که از آن استفاده می‌کنیم ($\mathbf{w}^{\mathrm{T}}\mathbf{w}$). به شکل کلی، خطای افزوده برای هر $h \in \mathcal{H}$ به شکل زیر است:

[۱] augmented error

$$E_{aug}(h,\lambda,\Omega) = E_{in}(h) + \frac{\lambda}{N}\Omega(h) \qquad (۳-۴)$$

که در مورد روش زوال وزن $\Omega(h) = \mathbf{w}^{\mathrm{T}}\mathbf{w}$ است و وزن‌های بزرگ را جریمه می‌کند. در حالت کلی ترم جریمه از دو جزء تشکیل شده است: منظم‌ساز $\Omega(h)$ (نوع منظم‌سازی) که ویژگی مشخصی از h را جریمه می‌کند و پارامتر منظم‌سازی λ که نشان‌دهنده درجه منظم‌سازی است. طبق معادله بالا با افزایش نقطه‌داده‌ها نیاز به منظم‌سازی کاهش می‌یابد و به همین دلیل ما از ضریب $\frac{1}{N}$ برای λ استفاده کردیم. این موضوع باعث می‌شود انتخاب مقدار بهینه λ حساسیت کمتری به N داشته باشد. این تعریف درواقع تعریف مجددی از پارامتر λ است با این هدف که آن را به پارامتر باثبات‌تری تبدیل کنیم که به‌راحتی قابل تفسیر باشد. همان‌طور که طبق نگاه اکتشافی به منظم‌سازی پیش‌بینی کرده بودیم، معادله ۴-۳ شباهت زیادی به کران VC دارد. به همین علت ما از علامت Ω برای مشخص کردن هر دو جریمه روی فرضیه منفرد $\Omega(h)$ و مجموعه فرضیات $\Omega(\mathcal{H})$ استفاده می‌کنیم.

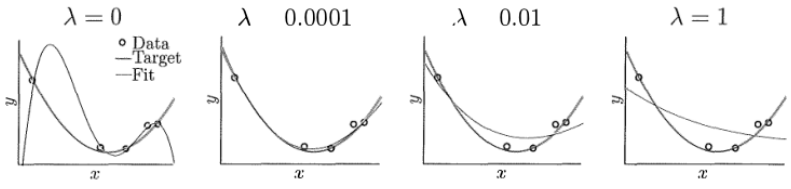
منظم‌ساز Ω اغلب قبل از رؤیت داده‌ها انتخاب و تثبیت می‌شود. هرچند در برخی موارد ممکن است خود مسئله منظم‌ساز خاصی را دیکته کند. در مقابل مقدار بهینه پارامتر منظم‌سازی معمولاً وابسته به داده‌ها است. بدست آوردن مقدار بهینه λ یکی از کاربردهای "اعتبارسنجی"[۱] است که به زودی آن را ارائه خواهیم داد.

شکل زیر اعمال منظم‌سازی زوال وزن به مثال قبل را با استفاده از مقادیر مختلف λ نشان می‌دهد. همان‌طور که مشاهده می‌شود حتی یک منظم‌سازی جزئی می‌تواند تأثیر زیادی داشته باشد، اما استفاده بیش از حد از منظم‌سازی منجر به یک منحنی تقریباً مسطح و برازش درون-نمونه بد می‌شود.

۴-۲-۲ انتخاب منظم‌ساز

در عمل انتخاب Ω عمدتاً اکتشافی است. پیدا کردن یک Ωی ایدئال همانند یافتن یک مجموعه فرضیات ایدئال کار دشواری است. این عمل وابسته به اطلاعاتی است که به دلیل ذات یادگیری در اختیار ما قرار ندارد. هرچند منظم‌سازهایی مانند زوال-وزن وجود دارند که بسیار رایج و

[۱] validation



شکل ۴-۴: روش زوال وزن که با مقادیر متفاوت پارامتر منظم‌سازی λ به مثال ۲-۴ اعمال شده است. با افزایش مقدار λ برازش مسطح‌تر می‌شود.

کاربردی هستند، اما به‌طورکلی در یک کاربرد خاص و یا در مورد داده‌های خاص ممکن است برخی از اشکال منظم‌سازی کار کنند و برخی دیگر کار نکنند. شکل ۴-۴ نشان می‌دهد که حتی درجه منظم‌سازی (پارامتر λ) باید به‌دقت انتخاب شود. مقدار زیاد منظم‌سازی به معنی ایجاد یک محدودیت سخت است و ممکن است انعطاف لازم برای برازش داده‌ها را فراهم نکند و منجر به زیربرازش[۱] شود که به اندازه بیش‌برازش مخرب است.

در برخی شرایط منظم‌سازی ضروری است. چنانچه مدل ما نسبت به داده‌هایی که داریم خیلی پیچیده باشد، ما ناچار به استفاده از منظم‌سازی هستیم. با استفاده از منظم‌سازی این شانس را خواهیم داشت که با انتخاب یک منظم‌ساز مناسب و انتخاب مقادیر بهینه پارامترها به یک برازش مطلوب برسیم. در اینجا اجازه دهید مسئله انتخاب میان دو منظم‌ساز مختلف را برای یک مدل $\mathcal{H}_{15}$ به‌عنوان مجموعه چند جمله‌های درجه ۱۵ بررسی کنیم. فرض کنید تابع هدف یک تابع نویزی درجه ۱۵ با واریانس نویز 0.5 است و مجموعه داده حاوی ۳۰ نقطه‌داده است. در این مسئله از دو منظم‌ساز به شکل زیر استفاده می‌کنیم:

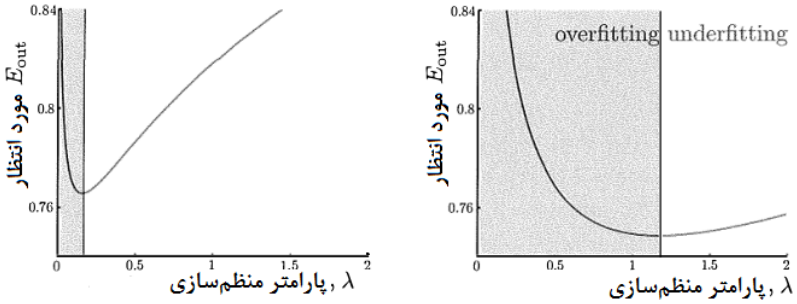
۱. منظم‌ساز یکنواخت $\Omega_{unif}(\mathbf{w}) = \sum_{q=0}^{15} w_q^2$

۲. منظم‌ساز پایین-مرتبه[۲] $\Omega_{low}(\mathbf{w}) = \sum_{q=0}^{15} q w_q^2$

منظم‌ساز اول مشوق کاهش یکنواخت تمام وزن‌ها است. منظم‌ساز دوم توجه بیشتری به کاهش وزن‌های با درجه بالا دارد و برازش‌های با درجه پایین را ترجیح می‌دهد.

شکل ۴-۵ کارایی خارج-از-نمونه مورد انتظار را برحسب مقادیر متفاوت پارامتر منظم‌سازی برای این دو منظم‌ساز نشان می‌دهد. با کم کردن پارامتر λ، الگوریتم بهینه‌سازی

[۱] underfitting
[۲] low-order



(الف) منظم‌ساز یکنواخت (ب) منظم‌ساز پایین مرتبه

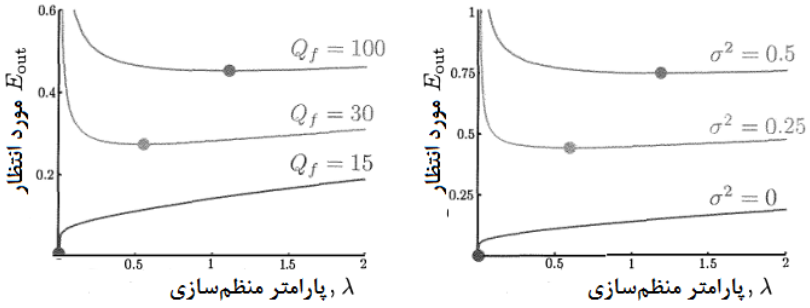
شکل ۴-۵: خطای خارج-از-نمونه برای منظم‌ساز یکنواخت و پایین مرتبه با استفاده از مدل $\mathcal{H}_{15}$ با $\sigma^2 = 0.5$، $Q_f = 15$ و $N = 30$. بیش‌برازش در نواحی خاکستری رنگ اتفاق می‌افتد زیرا E_{in} پایین منجر به E_{out} بالا می‌شود. زمانی که λ خیلی بزرگ میشود زیربرازش اتفاق می‌افتد زیرا الگوریتم یادگیری انعطاف لازم برای برازش داده‌ها را ندارد.

توجه کمتری به ترم جریمه کرده و توجه خود را معطوف به E_{in} می‌کند و درنتیجه E_{in} کاهش پیدا می‌کند. در قسمت سایه‌دار همراه با کاهش E_{in} (کاهش پارامتر λ)، E_{out} متناظر افزایش پیدا می‌کند (از راست به چپ). به عبارت دیگر، در این قسمت مقدار پارامتر منظم‌سازی کوچک است و درنتیجه محدودیت کافی بر روی یادگیری اعمال نمی‌شود. این امر منجر به کارایی کمتر ناشی از بیش‌برازش می‌شود.

در قسمت سفیدرنگ، مقدار پارامتر منظم‌سازی بسیار بزرگ است و محدودیت زیادی بر یادگیری اعمال می‌شود، درنتیجه انعطاف مورد نیاز برای برازش داده‌ها کم شده و این امر باز منجر به کارایی کمتر اما این بار ناشی از زیربرازش می‌شود. همان‌طور که در شکل مشاهده می‌شود، هزینه‌ای که عموماً برای بیش‌برازش داده می‌شود بسیار بیشتر از زیربرازش است. درنتیجه در این مورد، بهتر است محافظه‌کار نباشید.

همان‌طور که در شکل مشاهده می‌شود، مقدار بهینه پارامتر منظم‌سازی برای هر یک از این موارد کاملاً با یک‌دیگر متفاوت است. از سوی دیگر کارایی خارج-از-نمونه حساسیت زیادی به این مقدار دارد. با این حال، هرچند رفتار این دو منظم‌ساز بسیار متفاوت است اما نکته امیدبخش این است که در صورت انتخاب مقدار بهینه برای λ، کارایی به‌دست‌آمده توسط هرکدام نزدیک به هم و قابل‌مقایسه است (هر دو در حدود 0.76 است).

از این آزمایش همینطور می‌توان برای بررسی چگونگی تأثیر نویز بر منظم‌سازی استفاده کرد. در شکل ۴-۶ (الف) زمانی که نویزی وجود ندارد ($\sigma^2 = 0$)، هیچ مقداری از منظم‌سازی



(الف) نویز تصادفی (ب) نویز قطعی

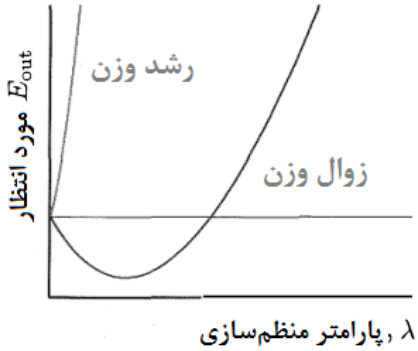
شکل ۴-۶: کارایی منظم‌ساز یکنواخت در سطوح مختلف نویز. مقدار بهینه λ در هر منحنی مشخص شده است.

تأثیر مثبتی بر کارایی مورد انتظار ندارد و پارامتر بهینه منظم‌سازی همان $\lambda = 0$ است. این امر عجیب نیست، زیرا در این مورد هیچ نوع نویز تصادفی و یا قطعی جود ندارد (هم مدل و هم تابع هدف از درجه ۱۵ هستند). با افزودن نویز تصادفی، همان‌طور که انتظار می‌رود کارایی کاهش پیدا می‌کند. توجه کنید که با افزایش نویز، مقدار بهینه پارامتر منظم‌سازی نیز افزایش پیدا می‌کند. بر اساس بحثی که ما در مورد تأثیر معکوس نویز بر بیش‌برازش داشتیم، این امر دور از انتظار نیست. بنابراین در این حالت محدود کردن بیشتر یادگیری مؤثر خواهد بود.

شکل ۴-۶ (ب) حالتی را نشان می‌دهد که ما نویز تصادفی را در مقدار صفر نگاه می‌داریم و نویز قطعی را افزایش می‌دهیم. برای این کار درجه تابع هدف را به شکل پلکانی افزایش می‌دهیم، درنتیجه نویز قطعی را افزایش یافته و بقیه چیزها ثابت نگه داشته می‌شود.

مقایسه بخش (الف) و (ب) شکل ۴-۶ نشان می‌دهد که تأثیر نویز قطعی و تصادفی تا چه اندازه مشابه یکدیگر است. زمانی که هر کدام از آنها وجود داشته باشد، استفاده از منظم‌سازی مفید است و هرچه نویز بیشتر شود، مقدار منظم‌سازی نیز بیشتر موردنیاز است.

اگر نوع منظم‌ساز را به‌درستی انتخاب نکنیم، چه اتفاقی می‌افتد؟ برای نشان دادن این مطلب، ما منظم‌ساز دیگری را انتخاب کردیم که در مقایسه با منظم‌ساز زوال وزن که وزن‌های کمتر را ترجیح می‌داد، وزن‌های بالاتر را ترجیح می‌دهد و آن را منظم‌ساز "رشد وزن" می‌نامیم. همان‌طور که در شکل ۴-۷ مشاهده می‌کنید، این منظم‌ساز هیچ کمکی به بیش‌برازش نمی‌کند. اگر اشتباهاً چنین منظم‌سازی را انتخاب کنیم، تا زمانی که روشی برای انتخاب مقدار مناسب پارامتر منظم‌سازی در اختیار داشته باشیم، مشکلی پیش نخواهد آمد. در این مورد خاص



شکل ۴-۷: مقدار بهینه پارامتر λ برای منظم‌ساز رشد وزن صفر است که به معنی عدم استفاده از این منظم‌ساز است.

مقدار بهینه پارامتر λ همان صفر است و استفاده از این منظم‌ساز بدتر از زمانی که از هیچ منظم‌سازی استفاده نکنیم نیست (معادل یکدیگرند).

ازآنجاکه ما هیچ‌گاه اطلاعات کاملی در اختیار نداریم، عدم استفاده از منظم‌ساز می‌تواند در کلیه شرایط و یا در یک شرایط خاص ممکن است بهترین گزینه باشد. اما چنانچه مقدار پارامتر منظم‌سازی λ در حد مناسبی انتخاب شود، تمام منظم‌سازها با موفقیت‌های نسبی مفید خواهند بود. درنتیجه تمام بار این تکلیف بر روی انتخاب مناسب λ است. این انتخاب می‌تواند توسط تکنیکی به نام اعتبارسنجی صورت گیرد که موضوع بحث ما در قسمت بعدی است.

درسی که از این مبحث یاد گرفتیم این است که استفاده از برخی از اشکال منظم‌سازی ضروری است، زیرا یادگیری به نویز قطعی و تصادفی حساس است. بهترین روش برای محدود کردن یادگیری، سوق دادن یادگیری در راستای تابع هدف است و هر چه نویز به‌عنوان عامل منحرف‌کننده بیشتر باشد، محدودیت بیشتری مورد نیاز است. درحالیکه ما نه از تابع هدف اطلاع داریم و نه از نویز، منظم‌سازی همچنان می‌تواند در کم کردن تأثیر نویز مؤثر باشد. مجموعه فرضیات اکثر مدل‌های رایج دارای پارامترهایی هستند و هر چه مقدار این پارامترها کوچک‌تر باشد، فرضیات هموارتر و ملایم‌تر خواهند بود. بنابراین منظم‌سازی از نوع زوال وزن یادگیری را به سمت انتخاب فرضیات هموارتر سوق می‌دهد. این عمل اغلب مؤثر است زیرا نویز تصادفی بسامد بالایی دارد. به‌طور مشابه نویز قطعی به‌عنوان بخش‌هایی از تابع هدف که توسط مدل مورد استفاده قابل مدل شدن نیستند گرایش به ناهمواری دارند. بنابراین محدود کردن یادگیری به سمت فرضیات هموارتر بیشتر از اینکه به توانایی الگوریتم در برازش

داده‌های مفید ضربه بزند، به پتانسیل آن در بیش‌برازش نویز ضربه می‌زند. هرچند در اینجا باید تأکید کرد که این‌ها بیشتر مشاهدات تجربی هستند تا جملات نظری قابل اثبات.

۴-۳ اعتبارسنجی

تاکنون بیش‌برازش را به‌عنوان مشکل، نویز به شکل تصادفی و یا قطعی را به عنوان دلیل و منظم‌سازی را به‌عنوان راه‌حل معرفی کردیم. در این قسمت راه‌حل دیگری به نام "اعتبارسنجی" [۱] را معرفی می‌کنیم. می‌توان به هر دو روش منظم‌سازی و اعتبارسنجی به شکل تلاشی در جهت کم کردن E_{out} به جای صرفاً کاهش E_{in} نگاه کرد. هرچند، E_{out} در دسترس ما نیست و نیاز به تخمینی از E_{out} با استفاده از اطلاعات موجود در نمونه داریم. منظم‌سازی تلاش می‌کند E_{out} را بر اساس رابطه زیر کمینه کند:

$$E_{out}(h) = E_{in}(h) + \underbrace{\text{جریمه بیش‌برازش}}_{\text{منظم‌سازی این کمیت را تخمین می‌زند}}$$

و یک ترم اکتشافی برای بخش جریمه ارائه کند. اعتبارسنجی از سوی دیگر سعی می‌کند این معادله را دور بزند و مستقیماً تخمینی از خطای خارج-از-نمونه به دست آورد.

$$\underbrace{E_{out}(h)}_{\text{اعتبارسنجی مستقیماً این کمیت را تخمین می‌زند}} = E_{in}(h) + \text{overfit penalty}$$

تخمین خطای خارج-از-نمونه به شکل مستقیم چیز جدیدی نیست. ما قبلاً هم ایده استفاده از مجموعه آزمایشی به‌عنوان زیرمجموعه‌ای از $\mathcal{D}$ که دخالتی در فرایند یادگیری ندارد و تنها برای ارزیابی فرضیه نهایی استفاده می‌شود را معرفی کرده‌ایم. E_{test} برخلاف E_{in} یک تخمین بی‌طرف از E_{out} است.

۴-۳-۱ مجموعه اعتبارسنجی

ایده اصلی مجموعه اعتبارسنجی تقریباً شبیه به مجموعه آزمایشی است. ما بخشی از داده‌ها را کنار می‌گذاریم و از آنها در آموزش استفاده نمی‌کنیم. سپس از این بخش برای تخمین خطای خارج از نمونه استفاده می‌کنیم. مجموعه کنار گذاشته‌شده، خارج از نمونه محسوب می‌شود زیرا در خلال یادگیری استفاده نشده است.

[۱] validation

با این حال تفاوتی میان مجموعه اعتبارسنجی و مجموعه آزمایشی وجود دارد. بااینکه مجموعه اعتبارسنجی به شکل مستقیم در آموزش دخالتی ندارد، اما برای انجام انتخاب‌های مشخصی در فرایند یادگیری استفاده می‌شود. زمانی که یک مجموعه به هر طریق در فرایند یادگیری تأثیر بگذارد، دیگر نمی‌توان اسم آن را مجموعه آزمایشی نامید. هرچند همان‌طور که خواهیم دید، روشی که مجموعه اعتبارسنجی در فرایند یادگیری استفاده می‌شود آن‌قدر بی‌خطر است که تخمین آن از E_{out} تقریباً سالم باقی می‌ماند.

اجازه دهید ابتدا ببینیم مجموعه اعتبارسنجی چگونه ایجاد می‌شود. گام اول این است که مجموعه $\mathcal{D}$ را به دو قسمت مجموعه آموزشی $\mathcal{D}_{train}$ با اندازه $N-K$ و مجموعه اعتبارسنجی $\mathcal{D}_{val}$ با اندازه K تقسیم کنیم. هر روش تقسیم‌بندی که وابسته به مقادیر نقطه‌داده‌ها نباشد برای این عمل مناسب است. برای مثال می‌توان $N-K$ نقطه را به شکل تصادفی برای آموزش انتخاب کنیم و بقیه را برای اعتبارسنجی کنار بگذاریم.

سپس الگوریتم یادگیری را روی $\mathcal{D}_{train}$ اجرا می‌کنیم تا فرضیه نهایی $g^- \in \mathcal{H}$ حاصل شود. بالانویس $^-$ در نماد این فرضیه اشاره به این مطلب دارد که بخشی از نقطه‌داده‌ها از مجموعه آموزشی کنار گذاشته شده‌اند. اکنون با استفاده از مجموعه $\mathcal{D}_{val}$، خطای اعتبارسنجی g^- را به شکل زیر محاسبه می‌کنیم:

$$E_{val}(g^-) = \frac{1}{K} \sum_{\mathbf{x}_n \in \mathcal{D}_{val}} \mathsf{e}(g^-(\mathbf{x}_n), y_n)،$$

که در آن $\mathsf{e}(g^-(\mathbf{x}_n), y_n)$ مقیاس خطای نقطه‌به‌نقطه‌ای است که در فصل اول معرفی شد. برای مسئله طبقه‌بندی، این مقیاس به شکل جمله $\mathsf{e}(g^-(\mathbf{x}_n), y_n) = [\![g^-(\mathbf{x}) \neq y]\!]$ و برای رگرسیون به شکل مربع خطا $\mathsf{e}(g^-(\mathbf{x}_n), y_n) = (g^-(\mathbf{x}) - y)^2$ تعریف می‌شود.

خطای اعتبارسنجی یک تخمین بی‌طرف از E_{out} است، زیرا فرضیه نهایی g^- مستقل از نقطه‌داده‌های مجموعه اعتبارسنجی ایجاد شد. در حقیقت با محاسبه مقدار مورد انتظار E_{val} برحسب نقطه‌داده‌های $\mathcal{D}_{val}$ به خطای خارج-از-نمونه فرضیه g^- می‌رسیم:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}_{val}}[E_{val}(g^-)] &= \frac{1}{K}\sum_{\mathbf{x}_n\in\mathcal{D}_{val}} \mathbb{E}_{\mathcal{D}_{val}}[\mathsf{e}(g^-(\mathbf{x}_n), y_n)],\\ &= \frac{1}{K}\sum_{\mathbf{x}_n\in\mathcal{D}_{val}} E_{out}(g^-), \qquad (۴\text{-}۴)\\ &= E_{out}(g^-)\end{aligned}$$

که قدم اول به دلیل خطی بودن مقدار مورد انتظار نتیجه می‌شود و قدم دوم به این دلیل صورت می‌گیرد که $\mathsf{e}(g^-(\mathbf{x}_n), y_n)$ تنها به $\mathbf{x}_n$ وابسته است و بنابراین:

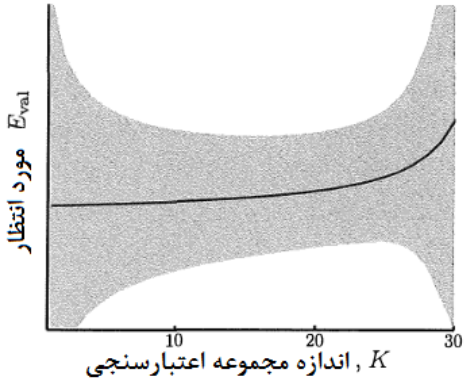
$$\mathbb{E}_{\mathcal{D}_{val}}[\mathsf{e}(g^-(\mathbf{x}_n), y_n)] = \mathbb{E}_{\mathbf{x}_n}[\mathsf{e}(g^-(\mathbf{x}_n), y_n)] = E_{out}(g^-)$$

چقدر تخمین E_{val} از E_{out} قابل‌اتکا است؟ در مورد طبقه‌بندی می‌توان از کران VC برای تحلیل قابلیت اعتماد تخمین خطای اعتبارسنجی از خطای خارج-از-نمونه استفاده کرد. به این منظور می‌توان $\mathcal{D}_{val}$ را همانند یک مجموعه آموزشی در نظر گرفت که قرار است تنها خطای فرضیه یکتای g^- روی آن محاسبه می‌شود. بنابراین می‌توان کران VC را روی این مدل متناهی حاوی یک فرضیه اعمال کرد (کران هافدینگ)، و با احتمال بالا ادعا کرد که:

$$E_{out}(g^-) \leq E_{val}(g^-) + O\big(\frac{1}{\sqrt{K}}\big) \qquad (۵\text{-}۴)$$

نامعادله فوق به توابع هدف دودویی قابل‌اعمال است. در حالت عمومی‌تر، می‌توان از واریانس E_{val} برای ایجاد یک کران مشابه برای توابع حقیقی استفاده کرد. در این حالت می‌توان ثابت کرد که خطای میان $E_{val}(g^-)$ و $E_{out}(g^-)$ حداکثر به اندازه $\frac{\sigma(g^-)}{\sqrt{K}}$ است که $\sigma(g^-)$ در مورد طبقه‌بندی با یک مقدار ثابت برحسب K کران‌دار می‌شود ($\sigma^2_{val} \leq \frac{1}{4K}$).

شکل ۴-۸ نمودار خطای مورد انتظار اعتبارسنجی برحسب K را برای $\mathcal{H}_2$ نمایش می‌دهد. برای رسم این نمودار از یک طرح آزمایشی با تابع هدفی به شکل چندجمله‌ای درجه ۱۰ و اندازه مجموعه آزمایشی $N = 40$ و سطح نویز 0.4 استفاده کردیم. این نمودار به روشنی نشان می‌دهد که برای به دست آوردن یک تخمین بی‌طرف از E_{out} باید بهایی را در قبال کنار گذاشتن K نقطه‌داده بپردازیم. زمانی که داده‌های بیشتری را برای اعتبارسنجی کنار

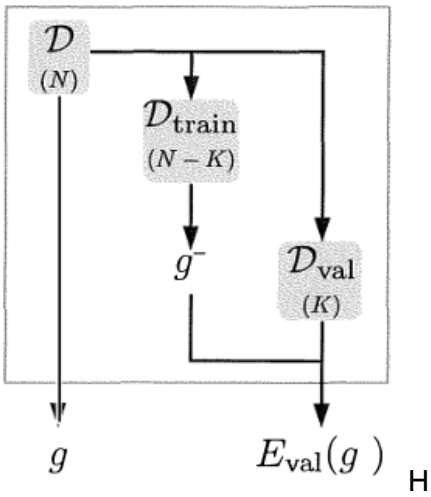


شکل ۴-۸: مقدار مورد انتظار خطا $\mathbb{E}[E_{val}(g^-)]$ به عنوان تابعی از K؛ ناحیه خاکستری برابر است با $\mathbb{E}[E_{val}] \pm \sigma_{val}$

می‌گذاریم، نقطه‌داده‌های آموزشی کمتری خواهیم داشت و درنتیجه فرضیه g^- بدتری حاصل خواهد شد. در پی آن $E_{out}(g^-)$ و بنابراین خطای اعتبارسنجی مورد انتظار افزایش پیدا می‌کنند (منحنی تیره‌رنگ). همان‌طور که انتظار می‌رود، عدم اطمینان نسبت به E_{val} که با σ_{val} اندازه‌گیری می‌شود (ناحیه هاشور خورده) با افزایش K، کاهش پیدا می‌کند (قابلیت اطمینان تخمین افزایش می‌یابد). این روند کاهش ادامه دارد تا اینکه دوباره $\sigma^2(g^-)$ به شکل جهشی افزایش پیدا می‌کند. این نقطه جایی است که تعداد نقطه‌داده‌های آموزشی به‌شدت کاهش پیدا کرده است. چنانچه K را نه خیلی کم و نه خیلی زیاد انتخاب کنیم، E_{val} تخمین خوبی از E_{out} را فراهم می‌کند. به‌عنوان یک قانون سرانگشتی بهتر است $K = \frac{N}{5}$ انتخاب شود؛ یعنی 20% داده‌ها را برای اعتبارسنجی کنار بگذاریم.

ما دو تقاضای متضاد را در مورد K مطرح کردیم. ازیک‌طرف K باید به اندازه کافی بزرگ باشد تا E_{val} یک تخمین قابل‌اعتماد باشد و از طرف دیگر باید به اندازه کافی کوچک باشد تا بتوان توسط $N - K$ نقطه باقیمانده فرضیه g^- مناسبی را استخراج کرد. نامعادله ۴-۵ تقاضای اول را کمّی می‌کند. تقاضای دوم را می‌توان با استفاده از منحنی یادگیری که در فصل ۲ ارائه شد کمّی کرد. منحنی یادگیری (همین‌طور منحنی نشان داده‌شده در شکل ۴-۹) بیان می‌کند که با افزایش نقطه‌داده‌های مجموعه آموزشی خطای خارج-از-نمونه مورد انتظار کاهش پیدا می‌کند. این واقعیت که داده‌های آموزشی بیشتر، فرضیه نهایی بهتری را تولید می‌کنند به طور گسترده‌ای در عمل ثابت شده است، هرچند اثبات آن به شکل نظری چالش‌انگیز است.

بازسازی $\mathcal{D}$: هرچند کنار گذاشتن K نقطه‌داده برای اعتبارسنجی و استفاده از $N - K$



شکل ۴-۹: استفاده از مجموعه اعتبارسنجی برای تخمین E_{out}

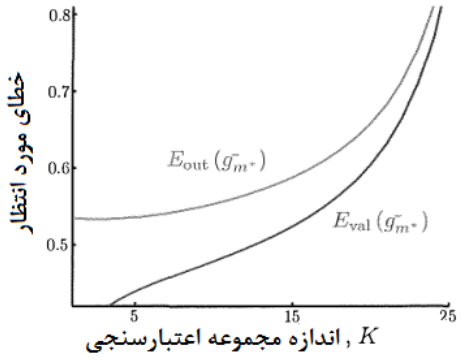
نقطه‌داده باقیمانده برای آموزش، هزینه‌ای در قبال E_{out} را به همراه دارد، اما ما مجبور به پرداخت این هزینه نیستیم. هدف اعتبارسنجی تخمین کارایی خارج-از-نمونه است و E_{val} تخمین نسبتاً خوبی از$(g^-)E_{out}$ را فراهم می‌کند. اما این موضوع به این معنی نیست که ما باید g^- را به‌عنوان فرضیه نهایی گزارش کنیم. هدف اولیه بدست آوردن بهترین فرضیه ممکن است. بنابراین باید فرضیه g که روی تمام مجموعه داده $\mathcal{D}$ آموزش دیده است را گزارش کنیم. هدف ثانویه تخمین E_{out} است که این عمل از طریق اعتبارسنجی میسر می‌شود. با توجه به بحثی که در مورد منحنی یادگیری داشتیم $E_{out}(g) \leq E_{out}(g^-)$

$$E_{out}(g) \leq E_{out}(g^-) \leq E_{val}(g^-) + O\left(\frac{1}{\sqrt{K}}\right) \qquad (۶-۴)$$

این معادله به این معنی است که اگر ما آموزش را با $N-K$ نقطه‌داده انجام داده و از K نقطه‌داده باقیمانده برای اعتبار سنجی استفاده کنیم و سپس همه داده‌ها را برای استخراج g به کار ببریم، خطای اعتبار سنجی که به دست می‌آوریم احتمالاً همچنان از تخمین $E_{out}(g)$ که با استفاده از کران VC و $E_{in}(g)$ حاصل می‌شود، بهتر خواهد بود. این مطلب به‌ویژه برای مجموعه فرضیات پیچیده با d_{vc} بالا با احتمال بیشتری برقرار است.

تا اینجا از مجموعه اعتبارسنجی به‌عنوان روشی برای تخمین E_{out} استفاده کردیم، بدون

اینکه بخواهیم از این تخمین در اتخاذ تصمیمی که یادگیری را تحت تأثیر قرار می‌دهد استفاده کنیم. تخمین E_{out} به‌خودی‌خود دارای کاربرد است. برای مثال، مشتریان اغلب مایل‌اند بدانند که فرضیه یادگیری شده به چه صورت عمل خواهد کرد. اما همان‌طور که در قسمت بعد خواهیم دید وظیفه اصلی مجموعه اعتبارسنجی راهبری فرایند یادگیری است و این چیزی است که مجموعه اعتبارسنجی را از مجموعه آزمایشی متمایز می‌کند.



شکل ۴-۱۰: بایاس خوش‌بینانه از خطای اعتبارسنجی زمانی که اعتبارسنجی برای انتخاب مدل استفاده می‌شود.

۴-۳-۲ انتخاب مدل

مهم‌ترین کاربرد اعتبارسنجی انتخاب مدل است. این انتخاب می‌تواند انتخاب میان یک مدل خطی و یک مدل غیرخطی، انتخاب درجه چندجمله‌ای در یک مدل، انتخاب مقدار یک پارامتر منظم‌سازی، یا هر انتخابی که فرایند یادگیری را تحت تأثیر قرار می‌دهد باشد. تقریباً در هر موقعیت یادگیری، با چندین انتخاب روبرو هستیم و نیاز به روشی قانونمند برای انجام این انتخاب‌ها داریم.

نکته قابل‌توجه در مورد اعتبارسنجی این است که می‌توان از آن برای تخمین خطای خارج-از-نمونه بیش از یک مدل استفاده کرد. فرض کنید ما M مدل به شکل $\mathcal{H}_1, \ldots, \mathcal{H}_M$ داریم و قصد داریم یکی از آنها را انتخاب کنیم. می‌توان از اعتبارسنجی به این منظور استفاده کرد. به این ترتیب با استفاده از مجموعه داده آموزشی $\mathcal{D}_{train}$ یادگیری را برای هر یک از مدل‌ها تکرار کنید تا M فرضیه نهایی g_m^- حاصل شود. اکنون هر کدام از این فرضیات را روی مجموعه اعتبارسنجی ارزیابی کنید تا خطاهای $E_1, \ldots, E_M$ را به دست آورید:

$$E_m = E_{val}(g_m^-) \quad m = 1, \ldots, M$$

این خطاهای اعتبارسنجی خطای خارج-از-نمونه $E_{out}(g_m^-)$ را برای هر مدل $\mathcal{H}_m$ تخمین می‌زنند. با داشتن این خطاها، مدلی که دارای کمترین خطای اعتبارسنجی است انتخاب می‌شود. فرض کنید m^* اندیس این مدل باشد. بنابراین برای $\mathcal{H}_{m^*}$، $E_{m^*} \leq E_m$ برای $m = 1, \ldots, M$ برقرار است. مدل $\mathcal{H}_{m^*}$ مدلی است که بر اساس خطاهای اعتبارسنجی انتخاب شده است. دقت کنید که در این حالت E_{m^*} دیگر یک تخمین بی‌طرف از $E_{out}(g_{m^*}^-)$ نیست. ازآنجاکه ما مدلی با کمترین خطا را انتخاب کردیم، E_{m^*} دارای یک بایاس خوش‌بینانه خواهد بود. این بایاس خوش‌بینانه در موردی که قصد انتخاب میان $\mathcal{H}_2$ و $\mathcal{H}_5$ را داریم در شکل ۴-۱۰ رسم شده است. برای رسم این نمودار از یک طرح آزمایشی با چندجمله‌ای درجه ۳، $\sigma^2 = 0.4$ و $N = 35$ استفاده شده است.

با توجه به فرایند مذکور برای انتخاب مدل با استفاده از اعتبارسنجی، خطای تعمیم تا چه اندازه خوب است؟ مدل جدیدی به نام $\mathcal{H}_{val}$ را به‌عنوان مجموعه‌ای از فرضیات نهایی $g_1^-, \ldots, g_M^-$ که توسط مدل‌های $\mathcal{H}_1, \ldots, \mathcal{H}_M$ روی داده‌های آموزشی تولید شده‌اند در نظر بگیرید.

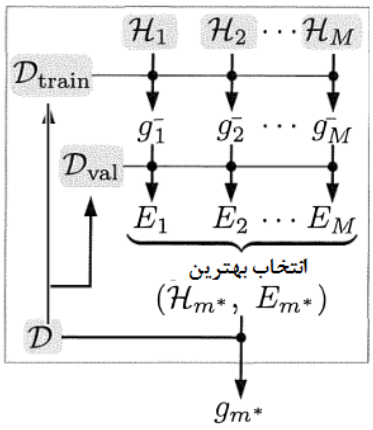
$$\mathcal{H}_{val} = \{g_1^-, \ldots, g_M^-\}$$

فرایند انتخاب مدل یکی از فرضیات مجموعه $\mathcal{H}_{val}$ را بر اساس کارایی آن روی مجموعه اعتبارسنجی $\mathcal{D}_{val}$ انتخاب می‌کند. ازآنجاکه مدل $\mathcal{H}_{val}$ قبل از رؤیت داده‌های مجموعه اعتبارسنجی قابل ایجاد است، فرآیند انتخاب مدل معادل این است که ما فرضیه‌ای از $\mathcal{H}_{val}$ را با استفاده از مجموعه داده $\mathcal{D}_{val}$ یاد بگیریم. خطاهای اعتبارسنجی $E_{val}(g_m^-)$ خطاهای درون-نمونه برای این فرایند یادگیری محسوب می‌شوند، درنتیجه می‌توان کران VC را روی مجموعه فرضیات متناهی $\mathcal{H}_{val}$ با اندازه M اعمال کرد:

$$E_{out}(g_{m^*}^-) \leq E_{val}(g_{m^*}^-) + O\left(\sqrt{\frac{\ln M}{K}}\right) \qquad (۴-۷)$$

چه اتفاقی می‌افتاد اگر از مجموعه اعتبارسنجی برای انتخاب مدل استفاده نمی‌کردیم؟ به‌عنوان یک روش جایگزین، می‌توانستیم از خطای درون-نمونه به‌عنوان معیار انتخاب مدل استفاده

کنیم. به‌عبارت‌دیگر مدلی که فرضیه نهایی آن دارای کمترین خطای درون-نمونه بود را انتخاب می‌کردیم. این عمل معادل این حالت است که فرضیه‌ای با کمترین خطای درون-نمونه را از یک مدل بزرگ‌تر حاوی تمام فرضیات موجود در کلیه M مدل اصلی انتخاب کنیم. اگر قرار بود کرانی را روی خطای خارج-از-نمونه این فرضیه نهایی به دست بیاوریم، می‌بایست ترم جریمه VC را برای این مجموعه بزرگ حاصل از اجتماع M مجموعه فرضیات محاسبه کنیم. ازآنجاکه بعد VC چنین مجموعه‌ای بسیار بالا خواهد بود، کران به‌دست‌آمده از آن بسیار شل‌تر از کرانی است که توسط ۴-۷ ایجاد می‌شود.



شکل ۴-۱۱: استفاده از مجموعه اعتبارسنجی برای انتخاب مدل

هدف از انتخاب مدل این است که بهترین مدل را انتخاب کرده و بهترین فرضیه موجود در آن را به‌عنوان فرضیه نهایی گزارش کنیم. به شکل مشخص ما قصد داریم مدل m را به گونه‌ای انتخاب کنیم که $E_{out}(g^-)$ زمانی که به همه داده‌ها رجوع می‌کنیم کمینه باشد. انتخاب مدل با استفاده از مجموعه اعتبارسنجی بر این اصل استوار است که اگر $E_{out}(g^-)$ کمینه باشد، در آن صورت $E_{out}(g_m^-)$ نیز کمینه است. خطای اعتبارسنجی E_m، $E_{out}(g_m^-)$ را تخمین می‌زند و طبق اصل مذکور مجموعه اعتبارسنجی مدل درست را انتخاب می‌کند. طبق بحثی که در مورد منحنی یادگیری داشتیم، صرف‌نظر از اینکه کدام مدل m^* انتخاب شود، ما نباید $g_{m^*}^-$ را به‌عنوان فرضیه نهایی گزارش کنیم، بلکه باید بار دیگر یادگیری را با استفاده از همه داده‌ها تکرار کنیم و g_{m^*} حاصل را گزارش کنیم که شرط زیر در مورد آن برقرار است:

$$E_{out}(g_{m^*}) \leq E_{out}(g_{m^*}^-) \leq E_{val}(g_{m^*}^-) + O\left(\sqrt{\frac{\ln M}{K}}\right) \qquad (۸-۴)$$

در این فرمول نیز نامعادله اول را بدون اثبات پذیرفتیم.

آزمایش قبل در مورد ارزیابی کارایی خارج-از-نمونه زمانی که از مجموعه اعتبارسنجی برای انتخاب میان مدل $\mathcal{H}_2$ و $\mathcal{H}_5$ استفاده کردیم را در اینجا تکرار می‌کنیم. نتایج در شکل ۴-۱۱ نشان داده‌شده‌اند. استفاده از اعتبارسنجی برای انتخاب مدل به وضوح نسبت به استفاده از E_{in} روش کارآمدتری است.

مثال ۳-۴: می‌توانیم از مجموعه اعتبارسنجی برای انتخاب مقدار بهینه پارامتر منظم‌سازی در خطای افزوده که در معادله ۴-۲ معرفی شد استفاده کنیم. هرچند بخش اصلی هر مدل مجموعه فرضیات آن است، اما متناظر با هر مجموعه فرضیات یک الگوریتم یادگیری نیز وجود دارد که فرضیه نهایی g را انتخاب می‌کند. از این لحاظ ممکن است دو مدل تنها از نظر الگوریتم یادگیری متفاوت باشند درحالی‌که از مجموعه فرضیات یکسانی بهره می‌برند. تغییر مقدار λ در خطای افزوده به نوعی باعث تغییر الگوریتم یادگیری می‌شود (معیاری که با آن g انتخاب می‌شود)، درنتیجه این عمل به شکل مؤثری مدل را تغییر می‌دهد.

بر اساس این بحث، M مدل متفاوت را در نظر بگیرید که همگی دارای مجموعه فرضیات یکسانی هستند، اما هر یک مقدار متفاوتی را برای λ در خطای افزوده انتخاب می‌کنند. بنابراین ما M مدل متفاوت به شکل$(\mathcal{H}, \lambda_1), (\mathcal{H}, \lambda_2), \ldots, (\mathcal{H}, \lambda_M)$ خواهیم داشت. برای مثال ممکن است ما از مقادیر $\lambda_1 = 0, \lambda_2 = 0.01, \lambda_3 = 0.02, \ldots, \lambda_M = 10$ برای این مدل‌ها استفاده کنیم. استفاده از مجموعه اعتبارسنجی برای انتخاب یکی از این M مدل متناظر با انتخاب مقدار λ از میان مقادیری با فاصله 0.01 است.

تاکنون استفاده از اعتبارسنجی برای انتخاب مدل بر اساس مجموعه‌ای متناهی از مدل‌ها را بررسی کردیم. اگر قرار باشد از اعتبارسنجی برای انتخاب مقدار یک پارامتر، برای مثال λ در مثال قبل، استفاده کنیم، در آن صورت مقدار M وابسته به درجه تفکیک‌پذیری مقادیر آن

پارامتر است (برای مثال 0.01 در مثال قبل). در حالت حدی زمانی که این درجه به صفر میل می‌کند، انتخاب میان مجموعه‌ای نامتناهی از مدل‌ها خواهد بود، زیرا مقدار λ می‌تواند هر عدد حقیقی باشد. درنتیجه کران‌های ۴-۷ و ۴-۸ که وابسته به M هستند دیگر قابل‌استفاده نیستند. اما همان‌طور که کران هافدینگ، زمانی که از یک مجموعه فرضیات متناهی به سمت مجموعه‌ای نامتناهی با بعد VC متناهی حرکت کردیم، از بین نرفت؛ به همان شکل کران‌هایی مانند ۴-۷ و ۴-۸ نیز کاملاً از بین نمی‌روند. در اینجا نیز می‌توان کران‌هایی از نوع VC را استخراج کرد، زیرا هرچند تعداد مدل‌ها نامتناهی است، اما این مدل‌ها همه به یک‌دیگر شبیه هستند و تنها به میزان جزئی در مقدار λ با یک‌دیگر تفاوت دارند. به‌عنوان یک قانون سرانگشتی، آنچه اهمیت دارد تعداد پارامترهایی است که قرار است از طریق اعتبارسنجی مقداردهی شوند. اگر تعداد این پارامترها کم باشد، در آن صورت تخمینی که بر اساس یک مجموعه اعتبارسنجی با اندازه معقول صورت می‌گیرد قابل‌اعتماد خواهد بود. هرچه انتخاب‌های بیشتری بر اساس یک مجموعه اعتبارسنجی یکسان صورت گیرد، این مجموعه بیشتر آلوده خواهد شد و تخمینی که فراهم می‌کند غیرقابل‌اعتمادتر خواهد بود. درواقع هر چه از مجموعه اعتبارسنجی برای تنظیم دقیق مدل بیشتر استفاده کنیم، مجموعه اعتبارسنجی بیشتر شبیه به یک مجموعه آموزشی خواهد بود که برای یادگیری مدل مناسب استفاده می‌شود و می‌دانیم که توانایی مجموعه آموزشی در تخمین خطای خارج-از-نمونه بسیار محدود است.

بسیار سخت است که بتوان یک مسئله جدی یادگیری را پیدا کرد که در آن اعتبارسنجی استفاده نشده باشد. اعتبارسنجی روشی است که از لحاظ مفهومی ساده است و در شرایط بسیاری قابل‌استفاده است. همین‌طور به اطلاعات خاصی در مورد جزئیات مدل نیاز ندارد. عیب اصلی این روش این است که باعث کاهش مجموعه آموزشی می‌شود. این مشکل را می‌توان با استفاده یک نسخه اصلاح‌شده که در قسمت بعدی معرفی خواهد شد تا حد زیادی حل کرد.

۴-۳-۳ اعتبارسنجی متقابل

اعتبارسنجی بر پایه دنباله استدلال‌های زیر قرار دارد:

$$E_{out}(g) \underset{(\text{small } K)}{\approx} E_{out}(g^-) \underset{(\text{large } K)}{\approx} E_{val}(g^-),$$

این روابط تناقضی که در انتخاب K وجود دارد را برجسته می‌کنند. هدف ما تولید g است.

زمانی که K بزرگ است، اختلافی میان خطای $E_{out}(g^-)$ (خطای تخمین زده‌شده با E_{val}) و $E_{out}(g)$ (خطای نهایی یادگیری با استفاده از کل مجموعه $\mathcal{D}$) وجود دارد. ما مایلیم که K را تا حد ممکن کوچک انتخاب کنیم تا این اختلاف کم شود (به شکل ایدئال $K = 1$). اما با چنین انتخابی اتکاپذیری تخمین اعتبارسنجی را از دست خواهیم داد، زیرا کرانی که در سمت راست معادله ۴-۵ وجود دارد بسیار بزرگ خواهد شد. در این حالت بااینکه خطای اعتبارسنجی $E_{val}(g^-)$ همچنان یک تخمین بی‌طرف از $E_{out}(g^-)$ است (g^- روی $N-1$ نقطه آموزش می‌بیند)، اما این تخمین آن‌قدر غیرقابل‌اعتماد است که تقریباً می‌توان گفت بی‌فایده است زیرا این تخمین تنها بر اساس یک نقطه‌داده قرار دارد. این مشکل ما را به تخمین "اعتبارسنجی متقابل[۱]" از خطای خارج-از-نمونه می‌رساند. برای توضیح این روش ما روی نسخه ساده‌ای متمرکز می‌شویم که در آن مجموعه اعتبارسنجی تنها شامل یک نقطه‌داده است. این نسخه "کنارگذاری تک نقطه[۲]" نامیده می‌شود و راحت‌ترین مورد برای تحلیل است. البته نسخه‌های رایج‌تر معمولاً از K های بزرگ‌تری استفاده می‌کنند، اما اصل قضیه در همه آنها یکسان است.

N روش مختلف برای تقسیم داده‌ها به یک مجموعه آموزشی به اندازه $N-1$ و یک مجموعه اعتبارسنجی با اندازه ۱ وجود دارد. به‌طور مشخص فرض کنید:

$$\mathcal{D}_n = (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}), \cancel{(x_n, y_n)}, (x_{n+1}, y_{n+1}), \ldots, (x_N, y_N)$$

که در آن $\mathcal{D}_n$ مجموعه داده $\mathcal{D}$ منهای نقطه‌داده (x_n, y_n) است که با خط‌خوردگی مشخص شده است. فرضیه نهایی که با یادگیری روی $\mathcal{D}_n$ تولید می‌شود را g_n^- می‌نامیم. فرض کنید e_n خطایی است که g_n^- روی مجموعه اعتبارسنجی که تنها شامل تک نقطه $\{(x_n, y_n)\}$ است ایجاد می‌کند:

$$\mathsf{e}_n = E_{val}(g_n^-) = \mathsf{e}(g_n^-(\mathbf{x}_n), y_n).$$

تخمین اعتبارسنجی متقابل برابر با میانگین مقادیر e_n ها خواهد بود:

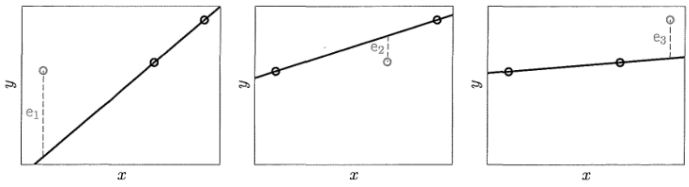
$$E_{\mathsf{cv}} = \frac{1}{N} \sum_{n=1}^{N} \mathsf{e}_n$$

شکل ۴-۱۳ اعتبارسنجی متقابل را روی یک مثال ساده نشان می‌دهد. هر خطای e_n یک تخمین تند[۳] اما بی‌طرف از $E_{out}(g_n^-)$ متناظر است که با جایگذاری $K = 1$ در ۴-۴ به دست

[۱] cross validation

[۲] leave-one-out

[۳] wild estimate



شکل ۴-۱۳: اعتبارسنجی متقابل به روش کنارگذاری تک نقطه برای یک برازش خطی با استفاده از سه نقطه داده. با میانگین‌گیری از سه خطای نمایش داده‌شده مقدار خطای E_{cv} بدست می‌آید.

می‌آید.

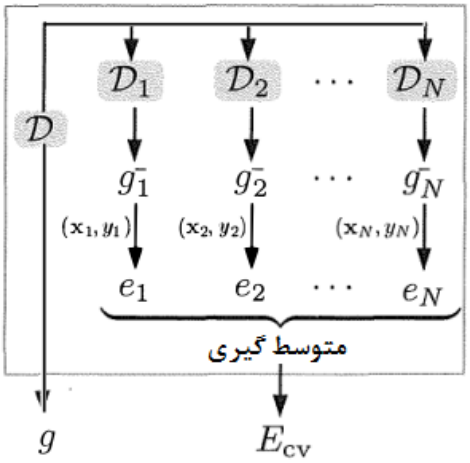
در اعتبارسنجی متقابل N تابع به شکل $g_1^-, \ldots, g_N^-$ و N تخمین خطا به شکل $\mathsf{e}_1, \ldots, \mathsf{e}_N$ وجود دارند. انتظار ما این است که این N خطا همه با هم تقریباً معادل تخمین E_{out} با استفاده از یک مجموعه اعتبارسنجی با اندازه N باشند. در همان حال توانسته‌ایم هر g_n^- را نیز با استفاده از $N-1$ نقطه‌داده آموزشی به دست آوریم. اجازه دهید ببینیم چرا E_{cv} می‌تواند تخمین خوبی از E_{out} باشد.

اولین و اصلی‌ترین نکته این است که E_{cv} یک تخمین بی‌طرف از $E_{out}(g^-)$ است. البته در اینجا باید کمی محتاط باشیم، زیرا در این حالت برخلاف زمانی که از یک مجموعه اعتبارسنجی استفاده می‌کردیم تنها یک فرضیه g^- وجود ندارد. بسته به نقطه (x_n, y_n)ای که کنار گذاشته می‌شود، هر g_n^- ممکن است فرضیه متفاوتی باشد. برای اینکه درک کنیم چگونه E_{cv} در این شرایط می‌تواند E_{out} را تخمین بزند، احتیاج به مرور دوباره مفهوم منحنی یادگیری داریم.

به شکل ایدئال هدف ما دانستن $E_{out}(g)$ است. فرضیه نهایی g از یادگیری روی یک مجموعه تصادفی $\mathcal{D}$ با اندازه N بدست می‌آید. به همین نسبت دانستن کارایی مورد انتظار مدل پس از یادگیری روی مجموعه داده‌ای با اندازه N نیز می‌تواند مفید باشد. فرضیه g تنها یک نمونه از یادگیری روی مجموعه داده‌ای با اندازه N است. کارایی مورد انتظار که حاصل متوسط گیری روی کلیه مجموعه داده‌هایی با اندازه N است، زمانی که به عنوان تابعی از N دیده شود، دقیقاً همان منحنی یادگیری است که در شکل ۴-۸ نمایش داده‌شده است. به شکل رسمی‌تر برای یک مدل داده‌شده فرض کنید

$$\bar{E}_{out}(N) = \mathbb{E}_{\mathcal{D}}[E_{out}(g)]$$

مقدار مورد انتظار خطای خارج- از-نمونه برای مجموعه داده‌های $\mathcal{D}$ با اندازه N باشد که



شکل ۴-۱۴: استفاده از اعتبارسنجی متقابل برای تخمین E_{out}

به‌وسیله مدل تولید می‌شود. در این صورت مقدار مورد انتظار E_{cv} دقیقاً $\bar{E}_{out}(N-1)$ است. این موضوع را می‌توان با استفاده از خطای اعتبارسنجی منفرد e_n به شکل زیر نشان داد:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}[\mathsf{e}_n] &= \mathbb{E}_{\mathcal{D}_n}\mathbb{E}_{(x_n,y_n)}[\mathsf{e}(g_n^-(\mathbf{x}_n), y_n)],\\ &= \mathbb{E}_{\mathcal{D}_n}[E_{out}(g_n^-)],\\ &= \bar{E}_{out}(N-1)\end{aligned}$$

ازآنجاکه این معادله برای هر e_n برقرار است درنتیجه برای متوسط آنها نیز برقرار است. ما این را نتیجه با ارائه قضیه زیر برجسته می‌کنیم:

قضیه ۴-۱: E_{cv} یک تخمین بی‌طرف از $\bar{E}_{out}(N-1)$ است. (مقدار مورد انتظار کارایی مدل،$\mathbb{E}[E_{out}]$، روی مجموعه داده‌هایی با اندازه $N-1$)

اکنون که تخمین اعتبارسنجی متقابل از E_{out} را داریم، نیازی نیست که هیچ‌یک از g_n^- ها را به‌عنوان فرضیه نهایی گزارش کنیم. بلکه می‌توانیم آخرین قطره کارایی را نیز بکشیم و با یادگیری مجدد روی کل داده‌های مجموعه $\mathcal{D}$، فرضیه g حاصل را به‌عنوان فرضیه نهایی گزارش

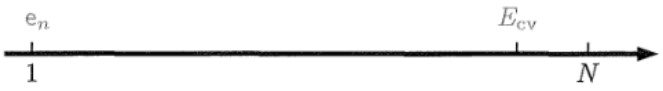
کنیم. به‌این‌ترتیب با رفتن از $N-1$ به N در منحنی یادگیری سودی هرچند ناچیز را به دست خواهیم آورد. در این حالت تخمین اعتبارسنجی متقابل به‌طور متوسط یک تخمین بالاسری برای خطای خارج-از-نمونه خواهد بود $E_{out}(g) \leq E_{\text{cv}}$. درنتیجه کارایی واقعی ممکن است اندکی ازآنچه انتظار داریم بهتر باشد.

زمانی که از اعتبارسنجی ساده با مجموعه‌ای با اندازه $K=1$ استفاده کردیم، می‌دانستیم که تخمین اعتبارسنجی خیلی قابل‌اعتماد نیست. اکنون سؤال این است که چقدر تخمین E_{cv} قابل‌اعتماد است؟ برای اندازه‌گیری این اتکاپذیری می‌توان از واریانس E_{cv} استفاده کرد. اما متأسفانه درحالی‌که می‌توان مقدار مورد انتظار E_{cv} را محاسبه کرد، محاسبه واریانس به این سادگی میسر نیست.

اگر N خطای اعتبارسنجی متقابل $\mathsf{e}_1, \ldots, \mathsf{e}_N$ با N خطا روی نقاط یک مجموعه کاملاً مجزای اعتبارسنجی با اندازه N معادل بودند، در آن صورت برای یک اندازه معقول N، E_{cv} به‌طور حتم یک تخمین قابل‌اتکا محسوب می‌شد. اگر تک‌تک e_n ها از یکدیگر مستقل بودند، چنین تناظری می‌توانست برقرار باشد، اما چنین فرضی خیلی خوش‌بینانه است. برای مثال دو خطای اعتبارسنجی e_n و e_m را در نظر بگیرید. خطای اعتبارسنجی e_n وابسته به g_n^- است که روی داده‌های شامل (x_m, y_m) آموزش می‌بیند، درنتیجه e_n به شکل غیرمستقیم به (x_m, y_m) وابسته است. از طرف دیگر خطای اعتبارسنجی e_m مستقیماً از طریق نقطه (x_m, y_m) محاسبه می‌شود، درنتیجه e_m نیز به (x_m, y_m) وابسته است. به‌این‌ترتیب میان e_n و e_m یک همبستگی از طریق نقطه‌داده (x_m, y_m) به وجود می‌آید. چنانچه یک فرضیه ثابت را با استفاده از N نقطه‌داده دست‌نخورده (مستقل) اعتبارسنجی می‌کردیم، چنین همبستگی وجود نداشت.

تخمین اعتبارسنجی متقابل تا چه اندازه از تخمینی که بر اساس یک مجموعه از N خطای اعتبارسنجی مستقل از هم حاصل می‌شود، بدتر است؟ به دست آوردن یک کران احتمالاتی از نوع VC و یا حتی محاسبه واریانس برای تخمین اعتبارسنجی متقابل کار آسانی نیست. یک راه برای کمّی‌سازی میزان اتکاپذیری E_{cv} این است که ببینیم چه تعداد نقطه‌داده‌های دست‌نخورده اعتبارسنجی لازم است تا یک میزان اتکاپذیری در حد E_{cv} را به وجود بیاورند. دو مقدار افراطی برای این اندازه مؤثر وجود دارد. در یک طرف N قرار دارد به این معنی که کلیه خطاهای اعتبارسنجی متقابل اساساً مستقل هستند. در طرف دیگر ۱ قرار دارد به این معنی که E_{cv} تنها در حد هر یک از خطاهای منفرد اعتبارسنجی متقابل e_n خوب است. به عبارت دیگر

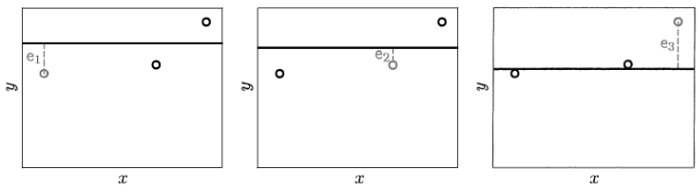
خطاهای اعتبارسنجی متقابل همه به یکدیگر وابسته هستند. درحالی‌که نمی‌توان چیزی را به صورت نظری اثبات کرد، در عمل E_{cv} نزدیک به سمت N است.



تعداد موثر مثال‌های دست‌نخورده که تخمیل قابل قیاسی از E_{out} را فراهم می‌کنند.

اعتبارسنجی متقابل برای انتخاب مدل: در شکل ۴-۱۱ تخمین E_m از خطای خارج-از-نمونه مدل $\mathcal{H}_m$ که از طریق یک مجموعه اعتبارسنجی به‌دست‌آمده بود نشان داده شد. ممکن است بتوان از تخمین اعتبارسنجی متقابل برای به دست آوردن E_m استفاده کرد: برای این کار لازم است که تخمین خطای خارج-از-نمونه هر یک از مدل‌های $\mathcal{H}_1, \ldots, \mathcal{H}_M$ را با استفاده از اعتبارسنجی متقابل به دست آوردیم؛ سپس مدلی که دارای کمترین خطای اعتبارسنجی متقابل است را انتخاب کنیم. اکنون مدل انتخاب‌شده را با استفاده از همه داده‌ها آموزش می‌دهیم تا فرضیه نهایی g حاصل شود، با این اعتقاد که $E_{out}(g^-)$ به‌خوبی از $E_{out}(g)$ پیروی می‌کند.

۴-۲: در شکل ۴-۱۳ از طریق یک آزمایش ساده با استفاده از سه نقطه تولیدشده از یک تابع ثابت نویزی، اعتبارسنجی متقابل را برای تخمین E_{out} یک مدل خطی به شکل $h(x) = ax + b$ نشان دادیم. در اینجا مدل دومی را در نظر می‌گیریم که یک مدل ثابت به شکل $h(x) = b$ است. در این مورد نیز می‌توان از اعتبارسنجی متقابل برای تخمین E_{out} استفاده کرد که این عمل در شکل ۴-۱۵ نمایش داده‌شده است.



شکل ۴-۱۵: اعتبارسنجی کنارگذاری-تک-نقطه برای یک برازش ثابت

اگر خطای درون-نمونه پس از برازش کلیه داده‌ها (سه نقطه) را در نظر بگیریم، مدل خطی برنده است زیرا قادر است از درجه آزادی بالاتر خود برای برازش بهتر داده‌ها استفاده کند. با استفاده از اعتبارسنجی متقابل، به‌طور مشابه خطای درون-نمونه برای مدل خطی صفر است زیرا هر دو نقطه را می‌توان با یک خط به هم وصل کرد. اما آنچه در اعتبارسنجی متقابل مهم

است، خطای این برازش روی تک نقطه باقیمانده (تنها نقطه مجموعه اعتبارسنجی) است که برای هر یک از برازش‌ها باید اندازه‌گیری شود. این خطاها در شکل ۴-۳ برای مدل خطی نمایش داده‌شده‌اند. خطاهای متناظر برای مدل ثابت نیز در شکل ۴-۱۵ نمایش داده‌شده‌اند. با مقایسه این دو شکل، حتی به صورت چشمی نیز می‌توان تشخیص داد که متوسط خطاهای اعتبارسنجی متقابل در مدل ثابت کمتر است ($E_{\text{cv}} = 0.063$ در مقابل $E_{\text{cv}} = 0.184$ برای مدل خطی). با اعتبارسنجی متقابل مدل ثابت برنده است. همین‌طور مدل ثابت دارای خطای خارج-از-نمونه کمتری نیز هست (توجه کنید که خود تابع هدف تابع ثابتی همراه با نویز است)؛ بنابراین در این مثال اعتبارسنجی متقابل مدل مناسب را انتخاب می‌کند.

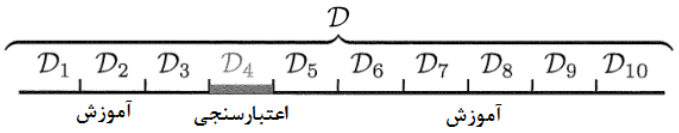
همان‌طور که در مثال ۳-۴ اشاره شد، یکی از کاربردهای مهم اعتبارسنجی تخمین مقدار بهینه پارامتر منظم‌سازی λ است. می‌توان از اعتبارسنجی متقابل نیز به این منظور استفاده کرد، با این تفاوت که به جای هر E_m از $E_{\text{cv}}(m)$ استفاده می‌شود. سپس مدلی که دارای E_{cv} کمتری است را انتخاب کرده و آن را روی کلیه داده‌های $\mathcal{D}$ آموزش می‌دهیم تا فرضیه نهایی g_{m^*} حاصل شود.

طبق شکل ۴-۱۴ مشاهده می‌شود که برای به دست آوردن E_{cv} برای یک مدل منفرد، به N دور یادگیری روی مجموعه‌های $\mathcal{D}_1, \ldots, \mathcal{D}_N$ هر یک با اندازه $N-1$ نیاز داریم. بنابراین استفاده از اعتبارسنجی متقابل برای M مدل، به MN دور یادگیری احتیاج خواهد داشت. این عملی زمان‌بر است. اگر قادر بودیم E_{cv} را به شکل تحلیلی به دست آوریم، جهش زیادی حاصل می‌شد. اما به‌کارگیری روش‌های تحلیلی برای اعتبارسنجی متقابل اغلب بسیار مشکل است. در این میان، مدل‌های خطی استثناء هستند و می‌توان یک فرمول تحلیلی دقیق برای محاسبه تخمین اعتبارسنجی متقابل به دست آورد.

حتی زمانی که برای یک مدل به‌آسانی نمی‌توان یک فرمول تحلیلی استخراج کرد، در عمل اعتبارسنجی متقابل قادر است تخمین خطای خارج-از-نمونه بسیار خوبی را فراهم کند به شکلی که بار محاسباتی اضافه اغلب ارزش تحمل دارد. همین‌طور این روش تقریباً در هر شرایطی قابل به‌کارگیری است و به اطلاعات زیادی در مورد جزئیات مدل احتیاج ندارد.

بااین‌حال برای مجموعه داده‌های بزرگ، زمان محاسبات می‌تواند یک پیامد محسوب شود. به همین دلیل روش کنارگذاری-تک-نقطه ممکن است همیشه روش منتخب نباشد. یک نسخه رایج مشتق شده از این روش اعتبارسنجی متقابل V-لایه نام دارد. در اعتبارسنجی متقابل

V-لایه، داده‌ها به V مجموعه مجزای $\mathcal{D}_1, \ldots, \mathcal{D}_V$ هرکدام به اندازه تقریبی N/V تفکیک می‌شوند. در این تقسیم‌بندی هر مجموعه $\mathcal{D}_v$ نقش مجموعه اعتبارسنجی را برای محاسبه خطای فرضیه g^- که از یادگیری روی مجموعه مکمل آن به‌دست‌آمده است بازی می‌کند. درنتیجه یک فرضیه همواره روی داده‌هایی اعتبارسنجی می‌شود که از آنها برای آموزش آن فرضیه خاص استفاده نشده است. خطای اعتبارسنجی V-لایه، میانگین V خطای اعتبارسنجی است که از مجموعه‌های اعتبارسنجی حاصل می‌شوند. درواقع اعتبارسنجی متقابل کنارگذاری-تک-نقطه را می‌توان به‌عنوان اعتبارسنجی متقابل N-لایه در نظر گرفت. فایده انتخاب $V \ll N$ کاهش بار محاسباتی است. اما عیب این روش در مقایسه با کنارگذاری-تک-نقطه‌این است که فرضیه g^- روی داده‌های کمتری آموزش می‌بیند و بنابراین اختلاف میان $E_{out}(g^-)$ (تخمین مجموعه اعتبارسنجی) و $E_{out}(g)$ افزایش می‌یابد. در عمل یک انتخاب رایج، اعتبارسنجی متقابل ۱۰-لایه است که یکی از لایه‌ها در شکل زیر نشان داده‌شده است.



اعتبارسنجی متقابل به روش ۱۰-لایه

۴-۳-۴ نظریه در مقابل عمل

مشابه چالش‌هایی که در بحث منظم‌سازی داشتیم، هر دو روش اعتبارسنجی و اعتبارسنجی متقابل چالش‌هایی را برای نظریه‌های ریاضی یادگیری ایجاد می‌کنند. نظریه تعمیم و به‌طور مشخص تحلیل VC زیربنای امکان‌پذیری یادگیری را تشکیل می‌دهد. این تحلیل نقش یک راهنما را ایفاء می‌کند که به‌وسیله آن می‌توان امکان تعمیم را با احتمال بالایی نتیجه‌گیری کرد. اما چنین نتیجه‌گیری در مورد تحلیل‌های اعتبارسنجی، اعتبارسنجی متقابل و منظم‌سازی به این سادگی به دست نمی‌آید و یا حتی در برخی مواقع امکان‌پذیر نیست. چیزی که امکان‌پذیر است و در عمل بسیار مؤثر است استفاده از نظریه به‌عنوان راهنما است. در مورد منظم‌سازی همان‌طور که احساس می‌شد محدود کردن فرایند انتخاب فرضیه موجب تعمیم بهتری می‌شود، حتی اگر از نظر فنی مجموعه فرضیات تغییری نکند. در مورد اعتبارسنجی انتخاب تعداد

محدودی پارامتر باعث آلوده شدن کامل تخمین E_{out} نمی‌شود، حتی اگر نظریه VC تضمین ضعیفی برای چنین تخمینی ارائه کند. و در آخر، در مورد اعتبارسنجی متقابل میانگین‌گیری از خطاهای اعتبارسنجی متعدد در عمل مفید واقع می‌شود، حتی اگر این تخمین‌ها مستقل از هم نباشند.

هرچند این تکنیک‌ها بر پایه چارچوب نظری محکمی بنا شده‌اند، اما همچنان باید به چشم "اکتشاف[۱]" دیده شوند زیرا به شکل کلی از یک توجیه ریاضی کامل برخوردار نیستند. یادگیری ماشین یک کار تجربی با پایه‌های نظری است. ما آنچه را که قابل‌اثبات است اثبات می‌کنیم و زمانی که اثبات جامعی در اختیار نداریم، از نظریه به‌عنوان راهنما استفاده می‌کنیم. در یک کاربرد عملی، اکتشاف ممکن است نسبت به یک رویکرد سختگیرانه که دارای فرضیات غیرواقعی است برنده باشد. در یک موقعیت مشخص، تنها راه متقاعد شدن در مورد اینکه چه چیزی کار می‌کند و چه چیزی کار نمی‌کند این است که آن تکنیک را آزمایش کنیم و ببینیم چه اتفاقی می‌افتد.

پیام‌های اصلی این فصل را می‌توان به شکل زیر خلاصه کرد:

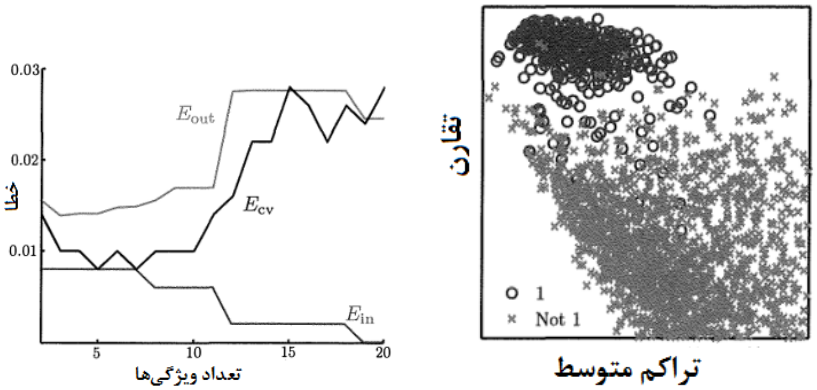
۱. نویز چه تصادفی و چه قطعی تأثیر معکوسی بر یادگیری دارد و منجر به بیش‌برازش می‌شود.

۲. منظم‌سازی با محدود کردن مدل و کاهش تأثیر نویز به تعدیل بیش‌برازش کمک می‌کند، درحالی‌که همچنان انعطاف لازم برای برازش داده‌ها را حفظ می‌کند.

۳. اعتبارسنجی و اعتبارسنجی متقابل تکنیک‌های مفیدی برای تخمین خطای خارج-از-نمونه هستند. یکی از کاربردهای مهم اعتبارسنجی انتخاب مدل و به‌طور ویژه تخمین مقدار مناسب پارامترهای منظم‌سازی است.

مثال ۳-۴: در اینجا قصد داریم کاربرد اعتبارسنجی را برای طبقه‌بندی ارقام دست‌نویس نشان دهیم. زیر مسئله موردنظر تشخیص رقم ۱ در تصویر است که بر اساس اندازه دو ویژگی تقارن و تراکم متوسط برای هر رقم قرار دارد. داده‌ها در شکل ۴-۱۶ الف نشان داده‌شده‌اند. به این منظور ۵۰۰ نقطه‌داده را به شکل تصادفی برای مجموعه آموزشی انتخاب کردیم و

[۱] heuristics



(ب) (الف) مسئله طبقه‌بندی ارقام

شکل ۴-۱۶: الف) داده‌های مربوط به ارقام که ۵۰۰ مورد از آنها به‌عنوان مجموعه آموزشی انتخاب شده‌اند. ب) داده‌ها توسط نگاشت چندجمله‌ای درجه ۵ به یک بردار ویژگی ۲۰ بعدی نگاشته شده‌اند. منحنی کارایی برحسب تعداد ویژگی‌هایی که برای طبقه‌بندی استفاده شده‌اند، نشان داده‌شده است.

بقیه را برای مجموعه اعتبارسنجی کنار گذاشتیم. همچنین از یک نگاشت غیرخطی به فضای ویژگی چندجمله‌ای درجه ۵ به شکل زیر استفاده کردیم:

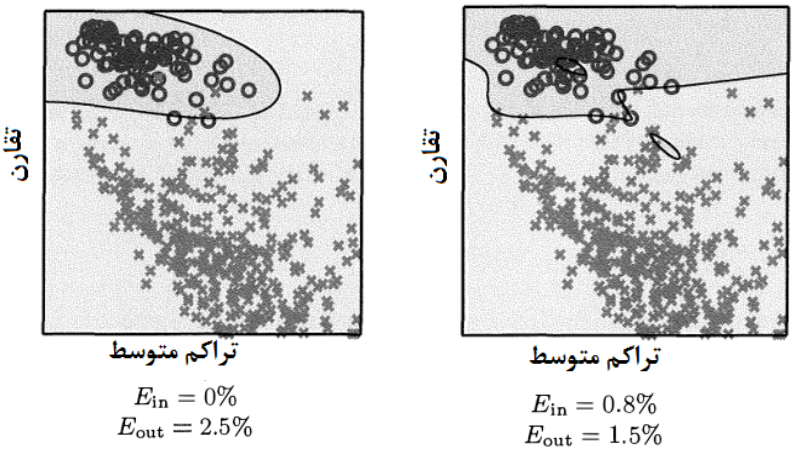
$$(1, x_1, x_2) \mapsto (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, \ldots, x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5)$$

شکل ۴-۱۶ (ب) خطای درون-نمونه را برحسب تعداد ویژگی‌های بکار گرفته‌شده در نگاشت نشان می‌دهد که از ۱ بُعد تا ۲۰ بُعد افزایش پیدا می‌کند. هر چه از ابعاد بیشتری استفاده شود، پیچیدگی مدل افزایش یافته و همان‌طور که انتظار می‌رود خطای درون-نمونه کاهش می‌یابد. خطای خارج-از-نمونه ابتدا کاهش پیدا می‌کند تا اینکه به یک موازنه تقریب-تعمیم برسیم و دوباره شروع به افزایش می‌کند. تخمین خطای فراهم شده توسط اعتبارسنجی متقابل به روش کنارگذاری-تک-نقطه، خطای خارج-از-نمونه را به‌خوبی تعقیب می‌کند. اگر قرار بود مدلی را بر اساس خطای درون-نمونه انتخاب کنیم، باید از هر ۲۰ بُعد استفاده می‌کردیم و در آن صورت با خطای خارج-از-نمونه بالایی مواجه می‌شدیم. خطای اعتبارسنجی متقابل در عددی بین ۵ تا ۷ بُعد ویژگی کمینه می‌شود. ما ۶ بُعد را به‌عنوان مدل منتخب در نظر می‌گیریم. جدول زیر مقادیر کارایی به‌دست‌آمده را خلاصه می‌کند:

E_{out}	E_{in}	
2.5%	0%	عدم اعتبارسنجی
1.5%	0.8%	اعتبارسنجی متقابل

تأثیر استفاده از اعتبارسنجی متقابل در بهبود کارایی به اندازه 1% است که بهبود نسبی عظیمی محسوب می‌شود و به معنی 40% کاهش در نرخ خطا است.

در اینجا جالب است نگاهی به مرزهای طبقه‌بندی که با و بدون استفاده از اعتبارسنجی ایجاد شده‌اند بیندازیم. طبقه‌بندهای حاصل همراه با ۵۰۰ نقطه‌داده مجموعه آموزشی در شکل بعد نشان داده‌شده‌اند.



(الف) طبقه‌بند ۲۰ بعدی (بدوع اعتبارسنجی)

(ب) طبقه‌بند ۶ بعدی (اعتبارسنجی کنارگذاری-تک-نقطه)

همان‌طور که به روشنی مشاهده می‌شود، کارایی خارج-از-نمونه پایین‌تر طبقه‌بندی که بدون اعتبارسنجی انتخاب شده است به علت بیش‌برازش تعداد کمی نقطه نویزی است که در داده‌های آموزشی وجود دارند. درحالی‌که داده‌های آموزشی در دو دسته تقریباً مجزا قرار دارند، شکل مرزهای نهایی تا حد زیادی خمیده شده‌اند که نشانه بیش‌برازش است. این تصویر ما را به یاد مثال اولی که در این فصل ارائه کردیم می‌اندازد و نشان می‌دهد که بیش‌برازش یک مسئله واقعی است و باید جدی گرفته شود.

فصل پنجم

سه اصل یادگیری

یادگیری ماشین برخی از اصول عمومی که در نوع خود مفاهیم جالبی هستند را برجسته می‌کند. تحلیل ریاضی و مثال‌های عملی که در فصل‌های قبل ارائه شدند چارچوب مناسبی را برای توضیح این اصول فراهم می‌کنند.

در این فصل ما سه اصل کلی را مورد بحث قرار می‌دهیم. اصل اول مربوط به انتخاب مدل است و به "تیغ اوکام[۱]" معروف است. دو اصل دیگر به داده‌ها مربوط می‌شوند. بایاس نمونه‌برداری اصل مهمی در مورد جمع‌آوری داده‌ها است و تجسس در دادها [۲] اصل مهم دیگری را در مورد مدیریت داده‌ها بیان می‌کند. درک این اصول به ما کمک می‌کند تا از به دام افتادن در چاله‌های معمول یادگیری ماشین پرهیز کنیم و همین‌طور بتوانیم کارایی تعمیم را به درستی تفسیر کنیم.

۵-۱ تیغ اوکام

هرچند این یک نقل‌قول دقیق از انیشتین نیست اما اغلب این جمله به وی نسبت داده می‌شود که «یک توصیف از داده‌ها باید تا حد ممکن ساده باشد اما نه ساده‌تر». اصل مشابهی به نام تیغ اوکام وجود دارد که به قرن چهارده میلادی برمی‌گردد و منتسب به ویلیام اوکام است که در آن "تیغ" به معنی کوتاه کردن یک توصیف به حداقل چیزی است که با داده‌ها همخوانی دارد.

در حوزه یادگیری، جریمه پیچیدگی مدل که در قسمت ۲-۲ معرفی شد یک مثال عینی از تیغ

[۱] Occam's razor

[۲] data snooping

اوکام است. اگر خطای درون-نمونه برابر با صفر باشد درنتیجه توصیف ارائه شده (فرضیه نهایی) با داده‌ها سازگار است. در این حالت، مناسب‌ترین توصیف با کمترین خطای تخمین خارج-از-نمونه (طبق کران VC) زمانی به دست می‌آید که پیچیدگی توصیف (که با $d_{\text{vc}}(\mathcal{H})$ اندازه‌گیری می‌شود) تا حد ممکن کوچک باشد. اصل زیربنایی را می‌توان به این شکل عنوان کرد:

«ساده‌ترین مدلی که با داده‌ها سازگار است، مطلوب‌ترین مدل نیز هست».

طبق این اصل ما باید ساده‌ترین مدلی که مناسب به نظر می‌رسد را انتخاب کنیم. با اینکه این اصل که "ساده‌تر بهتر است" ممکن است تا حدی قابل‌درک باشد، اما نه دقیق است و نه بدیهی. زمانی که قصد داریم این اصل را به یادگیری ماشین اعمال کنیم، دو سؤال اساسی پیش می‌آید:

۱. معنی مدل ساده چیست؟

۲. چرا مدل ساده‌تر بهتر است؟

اجازه دهید از سؤال اول شروع کنیم. دو رویکرد متفاوت برای تعریف پیچیدگی وجود دارد. اولی بر اساس خانواده‌ای از اشیاء قرار دارد و دیگری بر اساس یک شیء منفرد. ما قبلاً هر دوی این رویکردها را در تحلیل‌های خود دیده‌ایم. بُعد VC که در فصل ۲ معرفی شد مقیاسی از پیچیدگی است که بر اساس کل مجموعه فرضیات $\mathcal{H}$ (خانواده‌ای از اشیاء) قرار دارد. ترم منظم‌سازی در خطای افزوده که در فصل ۴ معرفی شد نیز مقیاسی از پیچیدگی است، اما در این مورد پیچیدگی مربوط به یک شیء منفرد است (فرضیه h).

این دو رویکرد به پیچیدگی تنها در حوزه یادگیری ماشین مطرح نمی‌شوند، بلکه هر جا بحث پیچیدگی پیش می‌آید، این رویکردها نیز دیده می‌شوند. برای مثال در نظریه اطلاعات، "آنتروپی"[۱] مقیاسی از پیچیدگی است که بر اساس خانواده‌ای از اشیاء قرار دارد، درحالی‌که "حداقل طول توصیف[۲]" مقیاس مرتبطی است که در مورد یک شیء منفرد تعریف می‌شود. دلیل تکرار این موضوع این است که درواقع این دو رویکرد به پیچیدگی با یکدیگر مرتبط هستند.

زمانی که می‌گوییم خانواده‌ای از اشیاء پیچیده هستند، منظورمان این است که این خانواده "بزرگ" است. به این معنی که شامل اشیاء بسیار متنوعی است. بنابراین هر شیء منفرد در

[۱] entropy

[۲] minimum description length

این خانواده 'یکی از بسیار' است. در مقابل، یک خانواده ساده از اشیاء "کوچک" است؛ تعداد اشیاء چنین خانواده‌ای نسبتاً کم است و هر شیء منفرد 'یکی از چندین' است.

چرا صرفاً تعداد اشیاء باید نشانگر سطح پیچیدگی باشد؟ تعداد اشیاء یک خانواده و پیچیدگی یک شیء منفرد هر دو به این وابسته هستند که چه تعداد پارامتر لازم است تا بتوان آن شیء را مشخص کرد. زمانی که تعداد پارامترهای یک مدل یادگیری را افزایش می‌دهید، هم‌زمان درجه تنوع $\mathcal{H}$ و درجه پیچیدگی هر h منفرد افزایش می‌یابد. برای مثال یک چندجمله‌ای درجه ۳ را در مقابل یک چندجمله‌ای درجه ۱۷ در نظر بگیرید. تنوع بیشتری در چندجمله‌ای‌های درجه ۱۷ وجود دارد و هم‌زمان یک چندجمله‌ای درجه ۱۷ نسبت به یک چندجمله‌ای درجه ۳ پیچیده‌تر است.

یک تعریف رایج از پیچیدگی شیء بر اساس تعداد بیت‌هایی است که برای توصیف آن شیء مورد نیاز است. طبق این تعریف، یک شیء ساده شیء‌ای است که توصیف کوتاهی دارد. درنتیجه یک شیء ساده نه‌تنها ذاتاً ساده است (توصیف مختصری دارد)، بلکه یکی از چندین مورد است، زیرا تعداد اشیایی که توصیف کوتاهی دارند نسبت به تعداد اشیایی که توصیف طولانی دارند بسیار کمتر است.

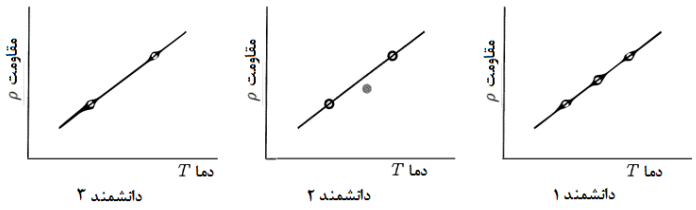
اکنون سؤال دوم را پاسخ می‌دهیم. زمانی که تیغ اوکام می‌گوید «ساده‌تر بهتر است» منظور این نیست که ساده‌تر لزوماً عالی‌تر و یا برازنده‌تر است. منظور این است که شانس ساده‌تر برای اینکه درست باشد بیشتر است. تیغ اوکام در مورد کارایی است نه زیبایی. چنانچه توصیف پیچیده‌ای از داده‌ها کارایی بهتری داشته باشد، ما آن را انتخاب می‌کنیم.

جمله " شانس ساده‌تر برای اینکه درست باشد بیشتر است" به این صورت توضیح داده می‌شود. ما سعی می‌کنیم داده‌های $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ را توسط فرضیه‌ای برازش کنیم (فرض کنید y_n ها دودویی هستند). تعداد فرضیات ساده نسبت به تعداد فرضیات پیچیده کمتر است. با فرضیات پیچیده، تعداد زیادی فرضیه یافت خواهند داشت که $\mathbf{x}_1, \ldots, \mathbf{x}_N$ را خُرد می‌کنند. بنابراین درهرصورت ما مطمئن خواهیم بود که قادریم مجموعه داده را برازش کنیم. این مطلب نشان می‌دهد که فارغ از مقادیر برچسب‌هایی $\mathbf{y}_1, \ldots, \mathbf{y}_N$، فرضیه‌ای برای برازش داده‌ها یافت خواهد شد حتی اگر این مقادیر کاملاً تصادفی باشند. در این صورت برازش داده‌ها دیگر معنی زیادی نخواهد داشت. اما چنانچه مدل ساده باشد و باوجود تعداد فرضیات کمتر، ما همچنان بتوانیم فرضیه‌ای را بیابیم که به شکل ایدئال دوبخشی

$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ را برازش کند، در آن صورت این اتفاق جای تعجب دارد و معنی‌دار خواهد بود.

تیغ اوکام به شکل رسمی تحت شرایط مختلفی اثبات شده است. گزاره بالا اساس این اثبات‌ها را در بر می‌گیرد. «چنانچه احتمال وقوع چیزی کم باشد، زمانی که اتفاق می‌افتد حادثه مهم‌تری محسوب می‌شود». اجازه دهید مثالی بزنیم.

فرض کنید یک فرضیه فیزیکی در مورد مقاومت یک فلز در دماهای مختلف ارائه کرده‌ایم. در این فرضیه، صرف‌نظر از مقادیر برخی ثابت‌ها که باید مشخص شوند، ادعا شده است که مقاومت P به شکل خطی به دمای T وابسته است. برای بررسی صحت این فرضیه و به دست آوردن مقادیر ثابت‌های ناشناخته، سه دانشمند سه آزمایش انجام داده و نتایج خود را مطابق شکل زیر ارائه کرده‌اند:



شکل ۵-۱

مشخص است که دانشمند اول توانسته است شواهد قانع‌کننده‌ای در تائید فرضیه به دست آورد. اگر اندازه‌گیری‌ها دقیق باشند، دانشمند دوم شواهدی را علیه فرضیه یافته است و ما باید دوباره به میز طراحی برگردیم. اما در مورد دانشمند سوم چه می‌توان گفت؟ درحالی‌که مدرک ارائه‌شده فرضیه را رد نکرده‌است، آیا می‌توان آن را نشانه درستی فرضیه دانست؟

پاسخ خیر است. زیرا می‌توان این سؤال را برعکس پرسید. فرض کنید که فرضیه غلط است، این داده‌ها چگونه می‌توانند ثابت کنند فرضیه غلط است؟ به هیچ طریق. چون هر دو نقطه‌ای را می‌توان با یک خط به هم وصل کرد. بنابراین نمی‌توان گفت احتمال دارد این مدل داده‌ها را برازش کند، بلکه قطعاً برازش خواهد کرد. زمانی که این اتفاق می‌افتد، برازش کاملاً بی‌اهمیت می‌شود.

این مثال یک مفهوم مرتبط با تیغ اوکام را نشان می‌دهد و آن اصل "ابطال ناپذیری"[۱] است.

[۱] non-falsifiability

این اصل بیان می‌کند چنانچه قرار است نتیجه بگیریم یک سری داده شواهدی را برای یک فرضیه فراهم می‌کنند، باید شانسی برای رد کردن آن فرضیه توسط این داده‌ها وجود داشته باشد. یک روش برای اینکه تضمین کنیم یک مجموعه داده شانسی برای رد کردن یک مدل دارد این است که بعد VC مجموعه فرضیات آن مدل کمتر از N، تعداد نقطه‌داده‌های مجموعه باشد.

در اینجا مثال دیگری از همان مفهوم را ارائه می‌کنیم. یک شرکت مالی قصد دارد چند دلال خوب استخدام کند، منظور از دلال خوب فردی است که بتواند نوسانات بازار را به‌خوبی پیش‌بینی کنند. فرض کنید از هر دلال خواسته می‌شود که نوسان بازار (بالا یا پایین) را برای پنج روز آینده پیش‌بینی کند و کسانی که پیش‌بینی بهتری داشته باشند، استخدام می‌شوند. ممکن است تصور شود که با این روش می‌توان بهترین دلال‌ها را به استخدام شرکت درآورد. اجازه دهید این مسئله را با دید یک مسئله یادگیری بررسی کنیم. فرض کنید هر دلال یک فرضیه پیش‌بینی کننده است. حال فرض کنید دایره استخدام خیلی بزرگ (پیچیده) باشد. ما با 2^5 دلال مصاحبه می‌کنیم. اتفاقاً این افراد مجموعه متنوعی را تشکیل می‌دهند و پیش‌بینی‌شان برای ۵ روز آینده کاملاً با یکدیگر متفاوت است. بنابراین لزوماً پیش‌بینی یکی از این افراد تحقق می‌یابد و استخدام می‌شود. استخدام دلال‌ها به این روش ممکن است ایده چندان خوبی نباشد، زیرا حتی اگر این دلال‌ها سکه‌ای را بالا می‌انداختند و طبق آن پیش‌بینی می‌کردند، نهایتاً یکی از آنها استخدام می‌شد. یک پیش‌بینی کننده خوب همیشه در چنین گروهی وجود دارد و درنتیجه پیدا کردن چنین فردی خیلی اهمیت ندارد. اما تصور کنید اگر ما تنها با دو دلال مصاحبه می‌کردیم و جواب یکی از آنها کاملاً درست در می‌آمد، در آن صورت انتخاب چنین فردی معنی داشت.

"یادگیری ماشین" تیغ اوکام را حتی به سطح فراتری از "ساده‌ترین حالت ممکن اما نه ساده‌تر" می‌برد. درواقع امکان دارد ما مدل ساده‌تری ازآنچه ممکن است را انتخاب کنیم. به‌طور مشخص ممکن است ما یک برازش غیر ایدئال از داده‌ها با استفاده از یک مدل ساده را به یک برازش ایدئال با استفاده از یک مدل پیچیده‌تر ترجیح دهیم. دلیل این انتخاب این است که ممکن است هزینه‌ای که باید برای یک برازش ایدئال برحسب جریمه پیچیدگی مدل (در کران تعمیم) بپردازیم در مقایسه با سودی که از این برازش به دست می‌آوریم خیلی بیشتر باشد. این ایده در شکل ۳-۳ به تصویر کشیده شده است و درواقع جلوه‌ای از بیش‌برازش است. این مطلب منطق پشت توصیه ما در فصل ۳ بود: «با یک مدل خطی شروع کنید» (یکی از ساده‌ترین مدل‌ها در حوزه یادگیری ماشین).

۵-۲ بایاس نمونه‌برداری

یک نمونه بارز از بایاس نمونه‌برداری در سال ۱۹۴۸ در انتخابات ریاست جمهوری آمریکا میان ترومن و دیوِی اتفاق افتاد. شب پایان انتخابات یک روزنامه معروف یک نظرسنجی تلفنی انجام داد و از افراد سؤال کرد به چه کسی رأی داده‌اند. نظرسنجی نشان می‌داد که دیوی برنده شده است. روزنامه آن‌قدر به خطای کوچک در نظرسنجی اطمینان داشت که تیتر زد «دیوی ترومن را شکست داد». زمانی که رأی‌شماری انجام شد، دیوی باخت درحالی‌که ترومن لبخندزنان تیتر روزنامه را نشان می‌داد.

این مورد نمونه‌ای از ناهنجاری آماری نبود که در آن روزنامه به‌شدت بدشانس بوده باشد (ضریب اطمینان را به خاطر بیاورید). در حقیقت این شکلی از نمونه‌برداری بود که فارغ از اندازه نمونه از ابتدا محکوم به شکست بود. حتی اگر نظرسنجی ده‌ها بار تکرار می‌شد نتایج تغییری نمی‌کردند.

حقیقت این است که در دهه چهل میلادی، تلفن وسیله گران‌قیمتی بود و کسانی که تلفن داشتند جزء گروه مرفه جامعه محسوب می‌شدند و نسبت به متوسط رأی‌دهندگان دیوی را ترجیح می‌دادند. ازآنجاکه روزنامه نظرسنجی را به‌وسیله تلفن انجام داد ناآگاهانه از توزیع درون-نمونه‌ای متفاوت از توزیع خارج-از-نمونه استفاده کرد.

بایاس نمونه‌برداری به شکل زیر تعریف می‌شود.

«اگر داده‌ها به شکل اریبی (دارای بایاس) نمونه‌برداری شوند، یادگیری نتیجه اریبی به همراه خواهد داشت.»

برای اعمال این اصل، باید مطمئن باشیم که توزیع‌های مجموعه‌های آموزشی و آزمایشی یکسان هستند. در غیر این صورت نتایج ممکن است نامعتبر باشند و یا حداقل نیاز به تفسیر دقیق داشته باشند.

اگر به خاطر بیاورید تحلیل VC فرضیات کمی داشت اما یکی از فرضیات اصلی این بود که مجموعه داده $\mathcal{D}$ باید توسط همان توزیعی تولید شده باشد که قرار است فرضیه نهایی g روی آن آزمایش شود. در عمل ممکن است ما مجموعه داده‌هایی داشته باشیم که تحت این شرایط ایدئال تولید نشده باشند. روش‌هایی در آمار وجود دارد که این "عدم انطباق" میان داده‌های آموزشی و داده‌های آزمایشی را جبران می‌کنند. اما در مورد یک مجموعه داده که در آن بخش‌های مشخصی از فضای نمونه کنار گذاشته شده‌اند (مانند کنار گذاشتن خانه‌هایی که تلفن ندارند در مثال بالا) روشی وجود ندارد. زمانی که چنین اتفاقی می‌افتد، کار چندانی نمی‌توان کرد مگر اینکه تائید کنیم که نتایج قابل‌اعتماد نیستند. دقت کنید پیش‌نیاز استفاده از کران‌های آماری مانند کران هافدینگ و کران VC وجود انطباق میان توزیع‌های آموزشی و آزمایشی است.

نمونه‌های زیادی را می‌توان مثال زد که چگونه بایاس نمونه‌برداری اتفاق می‌افتد. گاهی اوقات این امر به شکل سهوی و به علت اشتباه نمونه‌بردار رخ می‌دهد همانند مثال انتخابات ریاست جمهوری در بالا. در برخی موارد نیز این امر به دلیل عدم وجود انواع مشخصی از داده‌ها رخ می‌دهد. برای نمونه در مثال کارت اعتباری که در فصل یک آورده شد، مجموعه آموزشی از پایگاه داده‌ای به دست آمد که شامل اطلاعات مشتریان قبلی و نتیجه عملکردشان در بانک بود. بنابراین افرادی که درخواست کارت اعتباری داده و رد شده بودند شامل این مجموعه نمی‌شوند، زیرا داده‌ای مبنی بر نحوه عملکرد آنها در صورت تائید وجود ندارد. ازآنجاکه درخواست‌های آینده از جمعیت مرکبی شامل کسانی که ممکن بود در گذشته رد شوند می‌آید، توزیع مجموعه آزمایشی از توزیع مجموعه آموزشی متفاوت است و این نمونه‌ای از بایاس نمونه‌برداری است. در این حالت، کار چندانی نمی‌توان کرد مگر اینکه اعلام کنیم بایاسی در پیش‌بینی‌کننده تولیدشده وجود دارد زیرا مجموعه آموزشی نماینده جامعی از فضای ورودی نیست.

موارد رایج‌تر دیگری نیز وجود دارد که در آنها بایاس نمونه‌برداری با دخالت مستقیم افراد ایجاد می‌شود. خیلی غیرمعمول نیست که شخصی مثال‌هایی که دوست ندارد را بیرون بیندازد. برای مثال یک شرکت تجاری که قصد دارد یک سیستم تجاری خودکار را توسعه دهد، ممکن است داده‌هایی را برای آموزش سیستم انتخاب کند که همگی مربوط به زمانی باشند که بازار رفتار خوبی دارد. با این توجیه ظاهراً منطقی که آنها قصد ندارند اجازه دهند نویز فرایند آموزش را پیچیده کند. اگر آنها با این هدف مثال‌های بد را کنار بگذارند، قطعاً به هدف خود خواهند رسید؛ اما سیستم ایجادشده تنها در دوره‌هایی قابل‌اعتماد است که بازار رفتار خوبی دارد. اینکه در سایر اوقات چه اتفاقی می‌افتد، هرکسی می‌تواند حدس بزند. به‌طورکلی کنارگذاشتن مثال‌های آموزشی تنها بر اساس مقدارشان، برای نمونه مثال‌های حاشیه‌ای و یا مثال‌هایی که برابر با انتظارات قبلی ما نیستند، یک خطای رایج در نمونه‌برداری است.

بایاس‌های دیگری نیز وجود دارند که در آمار بیشتر مورد بحث قرار می‌گیرند. وجه اشتراک اکثر این بایاس‌ها این است که نتایج آماری بدست آمده را نامعتبر می‌کنند، زیرا این فرض اساسی که توزیع نمونه با توزیع کلی یکسان است را نقض می‌کنند. در حوزه یادگیری به طور خاص بایاس نمونه‌برداری که در مجموعه آموزشی رخ می‌دهد باید مورد توجه قرار گیرد.

۵-۳ تجسس در داده

تجسس در داده یک دام معمول دیگر در یادگیری ماشین است. اصل خیلی ساده است:

> «اگر یک مجموعه داده هر گامی در فرایند یادگیری را تحت تأثیر قرار دهد، توانایی آن در تخمین نتیجه کاهش می‌یابد.»

طبق این اصل اگر به دنبال تخمین بی‌طرفی از کارایی یادگیری هستید، باید مجموعه آزمایشی را در یک صندوقچه امن گذاشته و هرگز از آن در یادگیری استفاده نکنید. ما قبلاً این موضوع را به هنگام بحث در مورد تفاوت میان مجموعه آموزشی و آزمایشی بارها خاطرنشان کردیم، اما موضوع فراتر از این مسئله است. حتی اگر مجموعه داده‌ای عملاً در فرآیند آموزش استفاده نشود، همچنان ممکن است به شکل غیرمستقیم فرایند یادگیری را تحت تأثیر قرار دهد. برای اینکه از این دام پرهیز کنید، بسیار مهم است که قبل از دیدن هر نوع داده‌ای مدل یادگیری را انتخاب کنید. برای مثال ممکن است با نگاه کردن به یک مجموعه داده، تصور کنید که داده‌ها

به شکل خطی تفکیک‌پذیرند و زمانی که قرار است مدل یادگیری را انتخاب کنید به سمت استفاده از یک مدل خطی بروید.

می‌توان یک مدل را بر اساس یک سری اطلاعات کلی در مورد مسئله یادگیری مانند تعداد نقطه‌داده‌ها و یا دانش قبلی در مورد فضای ورودی و تابع هدف انتخاب کرد، اما نباید آن را بر اساس مجموعه داده $\mathcal{D}$ انتخاب کرد. کوتاهی در رعایت این قانون، کران VC را نامعتبر می‌کند و هر نوع نتیجه‌گیری بر اساس آن را زیر سؤال خواهد برد.

برای مثال فرض کنید یک شرکت تجاری قصد دارد سامانه‌ای را برای پیش‌بینی تغییرات نرخ ارز توسعه دهد. این شرکت داده‌هایی از تغییرات دلار آمریکا در مقابل پوند انگلستان در طول هشت سال را در اختیار دارد. ممکن است قبل از شروع یادگیری، داده‌ها را به یک میانگین و واریانس مشخص نرمال‌سازی کند و سپس با تفکیک آنها به مجموعه‌های آموزشی و آزمایشی عملیات یادگیری را شروع کند. برای هر روز، سیستم تلاش می‌کند بر اساس تغییرات نرخ ارز در ۲۰ روز گذشته، جهت تغییرات در آن روز را پیش‌بینی کند.

در این فرایند ممکن است شرکت هیچ گونه نگاهی به مجموعه آزمایشی نداشته باشد و به نتایج موفقیت‌آمیزی نیز روی این مجموعه دست یابد. اما پس از استقرار سیستم و بکارگیری آن روی داده‌های تجاری واقعی شاهد کارایی پایین سیستم باشد. در حقیقت در این مسئله ما باز شاهد تجسس غیرمستقیم در داده‌ها هستیم. در این مثال نرمال‌سازی به‌خودی‌خود ایده بدی نیست اما این عمل روی کلیه داده‌ها صورت می‌گیرد که داده‌های آزمایشی استخراج شده را نیز شامل می‌شود. در نتیجه مجموعه آزمایشی بر روی میانگین و واریانس مورد استفاده در نرمال‌سازی اثر گذاشته و به شکل غیر مستقیم بر روی انتخاب‌های یادگیری تأثیر می‌گذارد. به‌این‌ترتیب داده‌های آزمایشی آلوده شده و تخمین کارایی به‌دست‌آمده از آنها نامعتبر می‌شود.

یکی از موارد خیلی رایج تجسس در داده استفاده مجدد از همان مجموعه داده است. اگر شما یادگیری را ابتدا با یک مدل شروع کنید، سپس مدل دیگری را امتحان کنید و به همین شکل مدل‌های فراوانی را روی همان مجموعه داده بکار ببرید، شما نهایتاً موفق خواهید شد. به‌قول‌معروف «اگر شما داده‌ها را به مدت زیادی شکنجه دهید، داده‌ها اعتراف خواهند کرد». اگر شما کلیه دوبخشی‌های ممکن را روی یک مجموعه داده امتحان کنید، درنهایت هر مجموعه‌ای را برازش خواهید کرد. خواه این دوبخشی‌ها را مستقیماً با استفاده از یک مدل واحد امتحان کنید و یا به شکل غیرمستقیم به شکل دنباله‌ای از مدل‌ها آنها را امتحان کنید، در هر دو حالت

نتیجه یکی است. بعد VC مؤثر برای دنباله‌ای از مدل‌ها تنها برابر با بعد VC آخرین مدلی که موفق به برازش داده‌ها شد نیست، بلکه برابر با بعد VC اجتماع کلیه مدل‌هایی است که در تلاش‌های متعدد استفاده شده‌اند.

گاهی اوقات ممکن است استفاده مجدد از یک مجموعه داده توسط افراد متفاوت صورت گیرد. فرض کنید یک مجموعه داده عمومی وجود دارد و شما قصد دارید روی آن کار کنید. قبل از اینکه داده‌ها را دانلود کنید، مطالعه‌ای در مورد نتایجی که سایر افراد روی آن مجموعه داده با استفاده از تکنیک‌های مختلف به دست آورده‌اند انجام می‌دهید. سپس امیدبخش‌ترین تکنیک را به‌عنوان پایه کار خود انتخاب می‌کنید و سعی می‌کنید آن را توسعه داده و ایده‌های خود را به آن اضافه کنید. هرچند در این وضعیت ممکن است هنوز داده‌ای ندیده باشید، اما تا همین زمان نیز متهم به تجسس در داده هستید. انتخاب شما برای تکنیک پایه تحت تأثیر مجموعه داده است، حتی اگر این تاثیر از طریق اعمال دیگران باشد. ممکن است مشاهده کنید که تخمین به‌دست‌آمده خیلی خوش‌بینانه است. درواقع قبلاً ثابت شده است که تکنیک مورداستفاده با این مجموعه داده خاص سازگار است.

برای کمّی‌سازی آسیب ایجاد شده توسط تجسس در داده می‌توان جریمه پیچیدگی مدل را با احتساب این عمل محاسبه کرد. در مورد یک مجموعه داده عمومی، بعد VC مؤثر معادل بعد VC مجموعه فرضیاتی بسیار بزرگ‌تر از آن چیزی است که الگوریتم شما استفاده می‌کند. این مجموعه شامل تمامی فرضیاتی است که تاکنون توسط اشخاص مختلف در جهت رسیدن به یک راه‌حل مورد ملاحظه قرار گرفته (و اکثراً رد شده‌اند) و شما از نتایج منتشرشده آنها به‌عنوان پایه روشتان استفاده کرده‌اید. به شکل بالقوه این یک مجموعه بسیار عظیم با بعد VC بسیار بالا است. بنابراین تضمین تعمیم فراهم‌شده توسط معادله ۲-۹ بسیار بدتر از حالتی است که تجسس از داده صورت نمی‌گیرد.

تمام مجموعه داده‌هایی که در معرض تجسس از داده قرار می‌گیرند به یک نسبت آلوده نمی‌شوند. کران‌های ۱-۶ برای انتخاب از مجموعه متناهی فرضیات و ۲-۷ برای انتخاب از مجموعه نامتناهی فرضیات مقیاسی را برای تعین سطح آلودگی فراهم می‌کنند. هر چه انتخابی که بر اساس یک مجموعه داده صورت می‌گیرد شامل موارد جزئی‌تری باشد، مجموعه داده بیشتر آلوده شده و برای اندازه‌گیری کارایی فرضیه نهایی کمتر قابل‌اعتماد است.

برای مقابله با تجسس در داده دو رویکرد اصلی وجود دارد:

۱. پرهیز از تجسس: مقررات سخت‌گیرانه‌ای در مدیریت داده‌ها مورد نیاز است. داده‌هایی که قرار است برای ارزیابی کارایی نهایی استفاده شوند باید در یک محل امن نگهداری شده و تنها پس از اینکه فرضیه نهایی به دست آمد، بیرون آورده شوند. اگر برخی آزمایش‌های میانی مورد نیاز است، باید مجموعه داده‌های جداگانه‌ای به این منظور استفاده شوند. زمانی که یک مجموعه داده استفاده شد، هر جا که هدف تخمین کارایی است، آن مجموعه باید به‌عنوان آلوده تلقی شود.

۲. مقابله با تجسس: اگر قرار است از یک مجموعه داده بیش از یک‌بار استفاده کنید، سطح آلودگی را همواره در نظر داشته باشید و میزان اتکاپذیری تخمین کارایی را با توجه به این سطح برآورد کنید. کران‌های ۱-۶ و ۲-۷ راهنمایی را برای محاسبه اتکاپذیری نسبی مجموعه داده‌های مختلف که در نقش‌های مختلف در فرایند یادگیری استفاده می‌شوند فراهم می‌کنند.

تفاوت تجسس در داده و بایاس نمونه‌برداری

بایاس نمونه‌برداری مربوط به نحوه گرداوری داده‌ها قبل از هرگونه یادگیری است. تجسس در داده مربوط به نحوه انتخاب مدل یادگیری می‌شود. این مفاهیم مشخصاً مفاهیم متفاوتی هستند. اما مواردی وجود دارد که بایاس نمونه‌برداری به شکل پی‌آمدی از تجسس در داده اتفاق می‌افتد. در اینجا مثالی ارائه می‌کنیم. فرض کنید قرار است بازدهی سهام مختلف را بر اساس داده‌های تاریخی (سوابق داده) پیش‌بینی کنیم. برای بررسی کارایی یک قانون پیش‌بینی، شما داده‌های مربوط به کلیه شرکت‌های تجاری فعال را گرداوری می‌کنید و آن قانون را روی داده‌های سهام این شرکت‌ها در پنجاه سال اخیر آزمایش می‌کنید. فرض کنید شما قصد دارید تأثیر راهبرد "خرید و حفظ" را بررسی کنید، به این شکل که سهمی پنجاه سال پیش خریداری شده و تاکنون حفظ شده است. اگر چنین فرضیه‌ای را آزمایش کنید، احتمالاً به یک کارایی عالی برحسب سود خواهید رسید. اما زیاد هیجان‌زده نشوید، شما با انتخاب شرکت‌هایی که هم‌اکنون در حال فعالیت هستند ناآگاهانه داده‌ها را به سمت دلخواهتان منحرف کردید. به این معنی که شرکت‌هایی که اکنون وجود ندارند، بخشی از ارزیابی شما نیستند. زمانی که قانون بدست‌آمده را به اجرا می‌گذارید، این قانون بر روی کلیه شرکت‌ها اعمال می‌شود خواه این شرکت‌ها تا ۵۰ سال آینده باقی بمانند، خواه از بین بروند، زیرا شما نمی‌توانید از اکنون تشخیص دهید کدام شرکت فعال فعلی تا پنجاه سال دیگر نیز همچنان فعال باقی خواهد ماند. این یک نمونه

نوعی از بایاس نمونه‌برداری است. مشکل اصلی این است که داده‌های آموزشی نماینده جامع داده‌های آزمایشی نیستند. هرچند، اگر ریشه این مشکل را دنبال کنیم، درخواهیم‌یافت که ما برای انتخاب شرکت‌ها به داده‌های آینده آنها نگاه کرده‌ایم. ازآنجاکه از اطلاعاتی در آموزش استفاده کرده‌ایم که در تجارت واقعی به آنها دسترسی نداریم، این عمل نوعی تجسس در داده محسوب می‌شود.