فصل هشتم: یادگیری مبتنی بر نمونه‌ها

برخلاف متدهای یادگیری‌ای که توصیفی صریح از تابع هدف با استفاده از نمونه‌های آموزشی موجود ایجاد می‌کنند، یادگیری مبتنی بر نمونه‌ها[۱] فقط نمونه‌ها را ذخیره می‌کند و تعمیم روی نمونه‌ها را به زمان دسته‌بندی نمونه‌ی جدید موکول می‌کند. هرگاه نمونه‌ی جدیدی ارائه شد، رابطه‌ی آن با نمونه‌هایی که قبلاً ذخیره شده‌اند بررسی شده تا مقدار تابع هدف را برای این نمونه‌ی جدید تشخیص دهیم. یادگیری مبتنی بر نمونه‌ها، شامل متدهای نزدیک‌ترین همسایه[۲] و برازش وزن‌دار محلی[۳] که نمونه‌ها را به عنوان نقاطی در فضای اقلیدسی در نظر می‌گیرند می‌شود. این مبحث همچنین شامل متدهای استدلال مبتنی بر شرایط[۴] که از نمایش پیچیده‌ی نمادین برای نمایش فرضیه‌ها استفاده می‌کنند می‌شود. گاهی به متدهای مبتنی بر نمونه‌ها متدهای یادگیری تنبل[۵] نیز می‌گویند زیرا که محاسبات را تا دسته‌بندی نمونه‌های جدید به تعویق می‌اندازند. مزیت کلیدی این نوع تأخیر یا تنبلی این متدها دسته‌بندی محلی و به طور جداگانه برای هر یک از نمونه‌های جدید بجای تخمین تابع هدف برای کل فضای نمونه‌ای است.

۸,۱ معرفی

متدهای یادگیری مبتنی بر نمونه‌ها مثل نزدیک‌ترین همسایه و برازش وزن‌دار محلی، روش‌های ساده‌ای برای تخمین توابع هدف حقیقی مقدار و گسسته مقدارند. یادگیری در این الگوریتم‌ها به ذخیره‌ی نمونه‌های آموزشی در دسترس محدود می‌شود. زمانی که الگوریتم با یک نمونه‌ی جدید برخورد می‌کند، مجموعه‌ای از نمونه‌های مشابه از حافظه بازخوانی شده و برای دسته‌بندی نمونه‌ی جدید مورد استفاده قرار می‌گیرند. یکی از اختلاف‌های کلیدی بین این روش‌ها و متدهایی که در دیگر فصل‌های این کتاب مورد بحث قرار گرفت امکان ایجاد تخمین منحصربه‌فرد هر

---

۱ Instance based learning
۲ Nearest neighbor
۳ Locally weighted regression
۴ Case based reasoning
۵ lazy

نمونه‌ی جدید است. در واقع بسیاری از تکنیک‌ها فقط یک تخمین محلی از تابع هدف که در همسایگی نمونه‌ی جدید است خروجی می‌دهند و هیچ‌گاه تخمین جدیدی که برای کار بر روی کل فضای نمونه‌ای کار کند ایجاد نمی‌کنند. این خاصیت برای توابع هدفی که پیچیده اما به صورت محلی ساده‌اند مزیت دارد.

متدهای مبتنی بر نمونه‌ها را نیز می‌توان برای نمایش‌های پیچیده‌تر نمادین نمونه‌ها به کار برد. در یادگیری مبتنی بر شرایط، نمونه‌ها به فرم نمادین نمایش داده و فرایند تعیین نمونه‌های "همسایه" نیز بر اساس همین نمایش بیان می‌شود. از استدلال مبتنی بر شرایط در کارهایی نظیر ذخیره و استفاده‌ی دوباره‌ی تحقیق در یک میز کمک، استدلال درباره‌ی شرایط مجاز بر اساس شرایط قبلی، و حل مسائل زمان‌بندی پیچیده با استفاده‌ی از قسمت‌های مرتبط مسائلی حل شده‌ی قبلی به کار برده شده است.

یکی از مشکلات روش‌های مبتنی بر نمونه‌ها هزینه‌ی بالای دسته‌بندی نمونه‌های جدید است. این اشکال ناشی از این حقیقت است که تقریباً تمامی محاسبات در زمان دسته‌بندی نمونه‌های جدید (به جای زمان مواجهه با نمونه‌های یادگیری) انجام می‌شود. بنابراین، تکنیک‌هایی که برای فهرست[۱] کردن بهینه‌ی نمونه‌ها تأثیر قابل‌توجهی در کم کردن محاسبات لازم در زمان دسته‌بندی نمونه‌های جدید دارد. اشکال دوم بسیاری از روش‌های مبتنی بر نمونه‌ها، مخصوصاً روش‌های نزدیک‌ترین همسایه، این است که تمامی ویژگی‌های نمونه در زمان بازیابی نمونه‌های آموزشی مشابه در نظر گرفته می‌شود. اگر تابع هدف فقط به تعدادی از ویژگی‌های بسیار نمونه‌ها وابسته باشد، نمونه‌هایی که واقعاً مشابه نمونه‌ی جدید هستند با این نمونه فاصله بسیار زیاد داشته باشند.

در قسمت بعد الگوریتم k-Nearest Neighbor را به همراه چندین نسخه‌ی مختلف این الگوریتم پرکاربرد معرفی خواهیم کرد. در زیر قسمت بعد از آن برازش وزن‌دار محلی را مورد بحث و بررسی خواهد داد و متد یادگیری‌ای را که تخمین‌های موضعی‌ای در مورد تابع هدف می‌زند ارائه می‌کند، این متد را می‌توان تعمیم الگوریتم‌های k-Nearest Neighbor دانست. سپس شبکه‌ی radial basis را که پلی جالب بین یادگیری بر پایه‌ی نمونه‌ها و الگوریتم‌های شبکه‌های عصبی را بررسی خواهیم کرد. در قسمت بعدی به استدلال مبتنی بر شرایط، روشی مبتنی بر نمونه‌ها که از نمایش نمادین و استنتاج مبتنی بر دانش قبلی کمک می‌گیرد، بحث خواهد شد. این بخش شامل مثالی از کاربرد استدلال مبتنی بر شرایط در یک مسئله‌ی طراحی مهندسی را نیز در بر می‌گیرد. بالاخره، تفاوت‌های پایه‌ای بین ظرفیت‌هایی متدهای یادگیری تنبل که در این فصل مطرح می‌شوند و متدهای کوشا که در دیگر فصل‌های این کتاب مطرح می‌شوند را بررسی خواهیم کرد.

۸,۲ یادگیری k-Nearest Neighbor

ساده‌ترین متد یادگیری مبتنی بر نمونه‌ها، الگوریتم k-Nearest Neighbor است. این الگوریتم فرض می‌کند که تمامی نمونه‌ها نقاطی در فضای n بعدی $\Re^n$ هستند. نزدیک‌ترین همسایه‌ها یک نمونه با استفاده از تعریف استاندارد فاصله اقلیدسی تعریف می‌شوند. به عبارت دقیق‌تر اینکه هر نمونه‌ی دلخواه x با بردار زیر نمایش داده شود،

$$< a_1(x), a_2(x), \dots, a_n(x) >$$

که در آن $a_r(x)$ نشان دهند ی r امین ویژگی نمونه‌ی x است. پس فاصله‌ی بین دو نمونه‌ی x_i و x_j که با $d(x_i, x_j)$ نشان داده می‌شود به صورت زیر تعریف می‌شود:

---

[۱] indexing

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

در یادگیری نزدیک‌ترین همسایه[۱] ، تابع هدف ممکن است گسسته مقدار یا حقیقی مقدار باشد. بیایید فرض کنیم که تابع هدف گسسته مقدار است، $f: \Re^n \to V$، که در آن V مجموعه‌ی محدود $\{v_1, \ldots, v_s\}$ است. الگوریتم k-Nearest Neighber برای تخمین تابع گسسته مقدار در جدول ۸,۱ آورده شده است. همان‌طور که در جدول نیز نشان داده شده است، مقدار خروجی این الگوریتم $\hat{f}(x_q)$ تخمینی از $f(x_q)$ است که متداول‌ترین مقدار تابع هدف در بین نزدیک‌ترین همسایه‌های x_q است. اگر مقدار k را ۱ بگیریم الگوریتم ***1-nearest neighbor*** مقدار $f(x_i)$ را به $\hat{f}(x_q)$ نسبت می‌دهد (x_i نزدیک‌ترین همسایه‌ی x_q است.) برای مقادیر بزرگ‌تر k الگوریتم متداول‌ترین مقدار تابع هدف را میان k نزدیک‌ترین همسایه را به نمونه نسبت می‌دهد.

الگوریتم یادگیری:

- برای هر نمونه‌ی <x,f(x)>، نمونه را به مجموعه‌ی training_examples اضافه کن.

الگوریتم دسته‌بندی:

- نمونه‌ی x_q را برای دسته‌بندی بگیر
- $x_1 \ldots x_k$، k نمونه‌ی نزدیک‌تر به x_q را از training_examples پیدا کن.

مقدار زیر را خروجی بده

$$\hat{f}(\mathrm{x_q}) \leftarrow \arg\max_{\mathrm{v}\in\mathrm{V}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

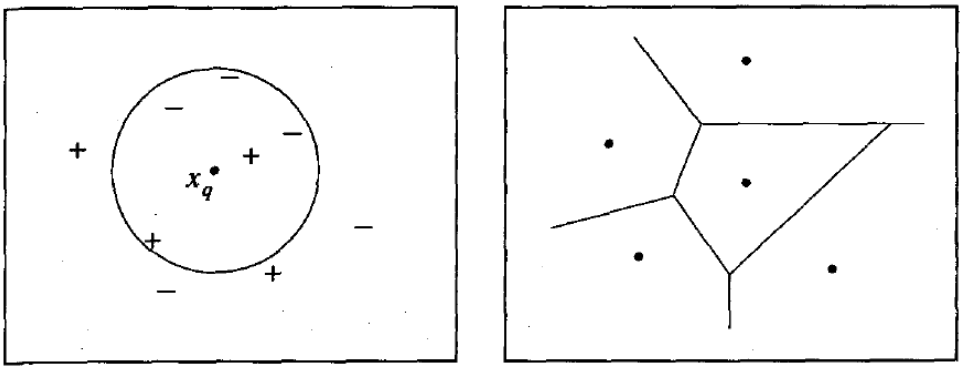
در این رابطه $\delta(a,b) = 1$ اگر a=b باشد، و در غیر این صورت $\delta(a,b) = 0$.

جدول ۸,۱ الگوریتم k-Nearest Neighbor برای تخمین تابع گسسته مقدار $\boldsymbol{f: \Re^n \to V}$.

شکل ۸,۱ عملیات الگوریتم k-Nearest Neighbor را در حالتی که نقاط در فضای دوبعدی‌اند و تابع هدف نیز منطقی است نشان می‌دهد. نمونه‌های مثبت و منفی به ترتیب با "+" و "–" در شکل نشان داده شده‌اند. نقطه‌ی x_q نیز در شکل نشان داده شده است. توجه دارید که در این شکل الگوریتم 1-Nearest Neighbor نمونه‌ی x_q را مثبت دسته‌بندی خواهد کرد در حالی که 5-Nearest Neighbor آن را منفی دسته‌بندی خواهد کرد.

---

[۱] Nearest Neighber



شکل ۸,۱ k-Nearest Neighbor

دسته‌ای از نمونه‌های مثبت و منفی به همراه یک نمونه‌ی جدید $\boldsymbol{x_q}$ در سمت چپ نشان داده شده است. الگوریتم 1-Nearest Neighbor نمونه‌ی جدید را مثبت دسته‌بندی می‌کند در حالی که الگوریتم 5-Nearest Neighbor همان نمونه را منفی دسته‌بندی می‌کند. در سمت راست شکل سطح تصمیم الگوریتم 1-Nearest Neighbor برای یک مجموعه‌ی متوسط از نمونه‌های آموزشی نشان داده شده است. چندضلعی‌های نشان داده شده‌ی دور هر نمونه، محدوده نقاطی را که نمونه به آن‌ها از نمونه‌های دیگر نزدیک‌تر است را نشان می‌دهد (در 1-Nearest Neighbor نمونه‌های داخل هر محدوده بر اساس نمونه‌ی آن محدوده دسته‌بندی خواهند شد).

طبیعت فضای فرضیه‌ای H که توسط الگوریتم k-Nearest Neighbor جستجو می‌شود چیست؟ توجه دارید که الگوریتم k-Nearest Neighbor هیچ‌گاه صریح فرضیه‌ی $\hat{f}$ را که تخمینی از f است معرفی نمی‌کند. این الگوریتم فقط نمونه‌های جدید را با توجه به نمونه‌های موجود دسته‌بندی می‌کند. با این وجود، هنوز می‌توان بررسی کرد که تابع ضمنی تخمین زده شده چیست یا چه دسته‌بندی با ثابت نگه‌داشتن نمونه‌های آموزشی و انتخاب نمونه‌های مختلف X به الگوریتم به دست می‌آید. شکل 8.1 دسته‌بندی الگوریتم 1-Nearest Neighbor را برای روی کل فضای نمونه‌ای نشان می‌دهد. فضای تصمیم نیز چندوجهی‌هایی هستند که هر کدام یک نمونه‌ی آموزشی را در بر می‌گیرند. برای هر نمونه‌ی آموزشی، چندوجهی محدوده‌ای را که نمونه‌هایش فقط تحت تأثیر یک نمونه‌ی آموزشی خاص‌اند را مشخص می‌کند. نمونه‌های خارج هر چندوجهی به نمونه‌ی دیگری نزدیک‌ترند. به این نوع نمودار گاهی اوقات نمودار ورونوی[۱] مجموعه‌ی نمونه‌های آموزشی نیز می‌گویند.

الگوریتم k-Nearest Neighbor به سادگی به توابع هدف پیوسته مقدار تعمیم می‌یابد. برای این تعمیم کافی است که به جای متداول‌ترین مقدار تابع هدف از میانگین k نمونه‌ی همسایه استفاده کنیم. به عبارت دقیق‌تر، برای تخمین تابع هدف حقیقی مقدار $f: \Re^n \rightarrow \Re$ کافی است که خط آخر الگوریتم بالا را با عبارت زیر جایگزین کنیم.

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{\mathrm{k}} \qquad (8.1)$$

[۱] Voronoi diagram

۸,۲,۱ الگوریتم Distance-Weighted Nearest Neighbor

الگوریتم k-Nearest Neighbor را می‌توان با اضافه کردن وزن به k نمونه‌ی همسایه بر اساس فاصله‌ی آن‌ها از نمونه‌ی x_q بهبود بخشید. برای مثال، در الگوریتم جدول ۸,۱ که توابع گسسته مقدار را تخمین می‌زند، می‌توانیم میزان تأثیر رأی هر یک از همسایه‌ها را متناسب با عکس فاصله‌شان از x_q تأثیر دهیم. این تغییر را می‌توان با تبدیل خط آخر الگوریتم به عبارت زیر اعمال کرد،

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \tag{8.2}$$

در این رابطه داریم،

$$w_i \equiv \frac{1}{d(x_q, x_i)^2} \tag{8.3}$$

در مواقعی که نمونه‌ی x_q با نمونه‌ی آموزشی x_i مساوی است، مقدار $d(x_q, x_i)^2$ صفر خواهد شد، در چنین شرایطی مقدار $f(x_i)$ را به $\hat{f}(x_q)$ نسبت می‌دهیم، اگر چندین نمونه‌ی آموزشی چنین شرایطی را داشتند متداول‌ترین مقدار هدف آن‌ها را انتخاب می‌کنیم.

برای توابع حقیقی مقدار نیز می‌توان به همین ترتیب با عوض کردن سطر آخر الگوریتم به رابطه‌ی زیر بهبود ذکر شده را اعمال کرد،

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i} \tag{8.4}$$

در این رابطه نیز w_i همان کمیت تعریف شده در رابطه‌ی ۸,۳ است. توجه داشته باشید که مقدار مخرج رابطه‌ی ۸,۴ مقدار کسر را نرمال می‌کند (برای مثال اگر برای تمامی x_i ها داشته باشیم $f(x_i) = c$، این مخرج به ما اطمینان می‌دهد که $\hat{f}(x_q) \leftarrow c$).

توجه دارید که تمامی نسخه‌های الگوریتم k-Nearest Neighbor که در بالا ذکر شد فقط k نقطه‌ی نزدیک‌تر به نمونه‌ی جدید را در نظر می‌گیرند. با اضافه کردن عامل وزن، واقعاً در نظر گرفتن تأثیر کل نمونه‌های آموزشی بر x_q ضرری نخواهد داشت، زیرا که نمونه‌های بسیار دور تأثیر بسیار کمی بر $\hat{f}(x_q)$ خواهند داشت. تنها اثر منفی این تغییر این است که دسته‌بندی کننده‌ی ما کندتر خواهد شد. متدی که از تمامی نمونه‌ها برای دسته‌بندی نمونه‌های جدید استفاده می‌کند را متد جهانی[۱] می‌نامند و متدهایی که فقط نزدیک‌ترین همسایه‌ها را در نظر می‌گیرند متدهای محلی[۲] می‌نامند. با تغییر قانون ۸,۴ به یک متد جهانی الگوریتم یادگیری به متد (Shepard 1968) تبدیل می‌شود.

۸,۲,۲ نکاتی در مورد الگوریتم k-Nearest Neighbor

الگوریتم Distance-weighted k-Nearest Neighbor متد استقرایی بسیار مؤثری است که در بسیاری از مسائل کاربردی مورد استفاده قرار می‌گیرد. این الگوریتم نسبت به داده‌های آموزشی خطادار حساس نیست و زمانی که مجموعه‌ی داده‌های آموزشی به اندازه‌ی کافی

---

[۱] global method

[۲] local method

بزرگ باشد بسیار مؤثر است. توجه داشته باشید که با گرفتن میانگین وزن‌دار k نمونه‌ی همسایه‌ی نزدیک x_q، الگوریتم می‌تواند اثـر داده‌هـای خطادار آموزشی را برطرف کند.

بایاس استقرایی k-Nearest Neighbor چیست؟ بایاس دسته‌بندی نمونه‌های جدید با توجه به شکل ۸,۱ به راحتـی مشـخص مـی‌شـود. بایاس استقرایی این الگوریتم این است که فرض می‌کند که دسته‌بندی نمونه‌ی x_q بسیار شبیه دسته‌بندی نمونه‌های دیگر نزدیک به آن است.

یک از مشکلات کاربردی استفاده‌ی الگوریتم k-Nearest Neighbor این است که فاصله‌ی بین نمونه‌ها با توجه بـه تمـامی ویژگـی‌هـای نمونه‌ها محاسبه می‌شود (در این مثال، تمامی محورهای فضای اقلیدسی). این رفتار الگوریتم با متدهای دیگر چون سیستم‌های یادگیری قوانین و یادگیری درختی که فقط تعدادی از ویژگی‌ها را در ساخت فرضیه به کار می‌برند متفـاوت اسـت. بـرای درک تـأثیر ایـن موضـوع، کـاربرد k-Nearest Neighbor را در مسئله‌ای که از ۲۰ ویژگی نمونه‌ها فقط ۲ ویژگی بر دسته‌بندی تابع هدف تأثیر دارند را در نظر بگیرید. در چنین شرایطی، نمونه‌هایی که این دو ویژگی‌شان نزدیک به هم است ممکن است در فضای ۲۰ بعدی بسیار دور از هم باشند. حاصل اینکه متد ساده‌ی به کار گرفته شده در k-Nearest Neighbor، که کل ۲۰ ویژگی را در نظر می‌گیرد، ما را گمراه خواهد کرد. فاصـله‌ی بـین همسـایه‌ها از تعداد زیادی از ویژگی‌های نامربوط محاسبه می‌شود. این مشکل، زمانی که تعداد ویژگی‌های نامربوط حاضر زیادنـد رخ مـی‌دهـد بعضـی اوقـات طلسم بعد[۱] نامیده می‌شود. Nearest-neighbor نسبت به این مشکل به شدت حساس است.

یکی از روش‌های جالب مقابله با این مشکل اعمال وزن در محاسبه‌ی فاصله‌ی بـین نمونه‌هاسـت. ایـن تغییـر مثـل کشـیدن و فشـرده کـردن محورهای فضای اقلیدسی است، محورهای ویژگی‌های نامربوط فشرده و محورهای ویژگی‌های مربوط کشیده می‌شوند. میزان کشش هر محـور از طریق یک روش ارزیابی[۲] مشخص می‌شود. برای مشخص شدن روش کار، فرض کنید که می‌خواهیم jامین محور را به انـدازه‌ی z_j بکشـیم (ضرب کنیم)، در این روش $z_1, \ldots, z_n$ طوری انتخاب می‌شوند تا خطای واقعی دسته‌بندی الگوریتم یادگیری مینیمم شود. دوم، توجـه داشـته باشید که این خطای واقعی را می‌توان با استفاده از ارزیابی به دست آورد. بنابراین، یکـی از الگوریتم‌هـای ممکـن انتخـاب یـک زیرمجموعـه‌ی تصادفی از داده‌های موجود برای نمونه‌های آموزشی و مشخص کردن $z_1, \ldots, z_n$ به صورتی است که خطا را بـرای بقیـه‌ی نمونـه‌ها مینـیمم کند. با چندین بار تکرار این فرایند تخمین این وزن‌ها دقیق‌تر می‌شوند. این فرایند کشیدن محورها برای بهینـه کـردن کـارایی k-Nearest Neighbor مکانیسمی برای کم کردن تأثیر ویژگی‌های نامربوط ایجاد می‌کند.

روش جایگزین مؤثرتر دیگر حذف ویژگی‌های نامربوط از فضای نمونه‌ای است. ایـن کـار مشـابه صـفر کـردن مقـدار z_i در متـد قبلـی اسـت. (Moore and Lee 1994) درباره‌ی کارایی متدهای cross-validation در انتخاب ویژگی‌های مربـوط از مجموعـه‌ی ویژگـی‌هـای موجود برای الگوریتم k-Nearest Neighbor را بررسی کرده‌اند. در کل، آن‌ها متدهای cross-validation و leave-one-out را که در آن مجموعه‌ی m تایی از نمونه‌های آموزشی متناوباً به مجموعه‌های m-1 تایی از نمونه‌های آموزشی و مجموعه‌ی تست یـک عضـوی در تمام حالات ممکن بررسی می‌شود. این روش leave-one-out به راحتی در k-Nearest Neighbor به کار برده می‌شود، زیرا کـه هر بار فقط مجموعه‌ی تست باز تعریف می‌شود هیچ تلاش آموزشی ی اضافی‌ای لازم نیست. توجه داشته باشید که روش‌های بالا را می‌تـوان بـا دید تغییر طول محورها با ضریب بررسی کرد. به طور مشابه می‌توان از ضرایبی برای تغییر طول محورها استفاده کـرد کـه در فضـای نمونـه‌ای

---

[۱] curse of dimensionality

[۲] cross-validation

ثابت نیست. با این وجود، زمانی که با این دید درجه‌ی آزادی الگوریتم را برای باز تعریف معیار فاصله زیاد می‌کنیم احتمال پدیده‌ی overfit نیز زیاد می‌شود. بنابراین، روش تغییر ناحیه‌ای محورها خیلی متداول نیست.

یکی از مشکلات کاربردی k-Nearest Neighbor فهرست بندی[۱] بهینه‌ی حافظه است. از آنجایی که این الگوریتم تمامی محاسبات را به زمان دریافت نمونه‌ی جدید موکول می‌کند، و ممکن است محاسبات قابل‌توجهی برای هر نمونه‌ی جدید لازم باشد، متدهای بسیاری برای فهرست بندی نمونه‌های آموزشی ایجاد شده است تا با هزینه کردن مقداری حافظه پیدا کردن نزدیک‌ترین همسایه‌ها راحت‌تر گردد. یکی از این متدهای فهرست بندی، متد kd-tree (Bentley 1975; Friedman 1977) است که در آن نمونه‌ها در برگ‌های درختی ذخیره می‌شوند و نمونه‌های مشابه نیز در همان گره یا گره‌های نزدیک ذخیره شده‌اند. گره‌های داخلی درخت، نمونه‌ی جدید x_q را با بررسی ویژگی‌هایش به سمت برگ مربوطه می‌فرستند.

۸,۲,۳ نکته‌ای در نمادگذاری

اکثر ادبیات به کار رفته در توضیح متدهای nearest-neighbor و weighted local regression عبارات تخصصی است که از مبحث تشخیص الگو آماری[۲] بر گرفته شده است. در خواندن چنین ادبیاتی بد نیست که با عبارات زیر آشنا باشید:

- برازش[۳] یعنی تخمین یک تابع هدف حقیقی مقدار.
- باقیمانده[۴] خطای $\hat{f}(x) - f(x)$ در تخمین تابع هدف است.
- تابع هسته[۵] تابعی از فاصله است که برای مشخص کردن وزن هر یک از نمونه‌های آموزشی به کار می‌رود. به عبارت دیگر، تابع هسته تابعی مثل K است که $w_i = K(d(x_i, x_q))$.

۸,۳ برازش وزن‌دار محلی

روش nearest-neighbor که در قسمت قبل توضیح داده شد را می‌توان تخمینی از تابع هدف f(x) برای یک نمونه‌ی $x = x_q$ دانست. الگوریتم برازش وزن‌دار محلی تعمیمی از این الگوریتم است. این الگوریتم تخمینی صریح از f در محدوده‌ی اطراف x_q می‌سازد. برازش وزن‌دار محلی از نمونه‌های آموزشی نزدیک یا کل نمونه‌ها به صورت وزن‌دار متناسب با فاصله[۶] برای ایجاد این تخمین محلی f استفاده می‌کند. برای مثال، ممکن است تابع هدف را در همسایگی اطراف x_q با استفاده از یک تابع خطی، یا یک تابع درجه دو، یا یک شبکه‌ی عصبی چندلایه، یا هر تابع دیگری تخمین بزنیم. عبارت "برازش وزن‌دار محلی" محلی است زیرا که دسته‌بندی فقط با توجه به نمونه‌های نزدیک نمونه‌ی جدید انجام می‌شود، وزن‌دار است زیرا که فاصله‌ی هر نمونه‌ی آموزشی بر تأثیر آن بر دسته‌بندی نمونه‌ی جدید اثر دارد، برازش است زیرا که این روش در یادگیری تخمین توابع حقیقی مقدار به کار می‌رود.

---

[۱] indexing
[۲] statistical pattern recognition
[۳] Regression
[۴] Residual
[۵] Kernel function
[۶] distance-weighted

با داشتن نمونه‌ی جدید x_q، روش کلی در برازش وزن‌دار محلی ساختن تخمینی مثل $\hat{f}$ است که در اطراف x_q نمونه‌های آموزشی را بپوشاند. سپس این تخمین برای محاسبه‌ی $\hat{f}(x_q)$ به کار می‌رود. مشخصات $\hat{f}$ از حافظه‌ی الگوریتم پاک خواهد شد زیرا که برای هر نمونه‌ی جدید باید تخمین محلی جدیدی ساخته شود.

۸,۳,۱ برازش وزن‌دار خطی محلی

حالتی را فرض کنید که برازش وزن‌دار محلی‌ای برای تخمین f در اطراف x_q از تابعی خطی به فرم کلی زیر استفاده می‌کند،

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \cdots + w_n a_n(x)$$

مثل قبل، در این رابطه نیز $a_i(x)$ نشان‌دهنده‌ی iامین ویژگی نمونه‌ی x است.

با توجه به آنچه که در فصل ۴ در مورد متدهایی مثل شیب نزول گفته شد، ضرایب مناسب $w_0, \ldots, w_n$ برای مینیمم کردن خطا بر روی نمونه‌های آموزشی استفاده می‌شود. در فصل ۴ علاقه‌ی ما به تخمین جهانی تابع هدف بود، بنابراین، خطای تعریف شده را برای تمامی نمونه‌های D تعریف کردیم،

$$E \equiv \frac{1}{2}\sum_{x \in D}\left(f(x) - \hat{f}(x)\right)^2 \tag{8.5}$$

که با آن به قانون شیب نزول زیر رسیدیم،

$$\Delta w_j = \eta \sum_{x \in D}\left(f(x) - \hat{f}(x)\right) a_j(x) \tag{8.6}$$

در این رابطه η ثابت یادگیری است. در اینجا قانون دوباره ارائه شده تا با نمادگذاری این فصل سازگاری داشته باشد (برای مثال، $t \rightarrow f(x), o \rightarrow \hat{f}(x), x_j \rightarrow a_j(x)$).

حال چگونه می‌توان این قانون را طوری تغییر داد تا به جای تخمینی جهانی، تخمینی محلی داشته باشد؟ راه‌حل ساده این است که خطای E را طوری تعریف کنیم تا تأکید بیشتری بر نمونه‌های محلی داشته باشد. سه روش مختلف اعمال این تغییر در زیر آورده شده. توجه داشته باشید برای تأکید بر اینکه این خطا تابعی از نمونه‌ی جدید x_q است می‌نویسیم $E(x_q)$.

- مینیمم کردن مربع خطای بین k نمونه‌ی نزدیک‌تر

$$E_1(x_q) \equiv \frac{1}{2}\sum_{x \in k\ nearest\ nbrs\ of\ x_q}\left(f(x) - \hat{f}(x)\right)^2$$

- مینیمم کردن مربع خطا برای کل نمونه‌ها، با این تفاوت که با افزایش فاصله بر اساس تابعی نزولی مثل K تأثیر نمونه‌ها کمتر می‌شود،

$$E_2(x_q) \equiv \frac{1}{2}\sum_{x \in D}\left(f(x) - \hat{f}(x)\right)^2 K(d(x_q, x))$$

- ترکیبی از دو روش اول

$$E_3 \equiv \frac{1}{2} \sum_{x \in k\ nearest\ nbrs\ of\ x_q} \left(f(x) - \hat{f}(x)\right)^2 K(d(x_q, x))$$

شاید روش دوم زیباترین تعریف خطا باشد، زیرا که در آن تمامی نمونه‌ها اثری بر دسته‌بندی x_q خواهند داشت. با این وجود، این روش محاسباتی لازم دارد که با افزایش تعداد نمونه‌های آموزشی به طور خطی افزایش می‌یابند. روش سوم که بین دو روش اول است، هزینه‌ی محاسباتی مستقل از تعداد کل نمونه‌های آموزشی دارد؛ این روش فقط k همسایه‌ی نزدیک‌تر را در نظر می‌گیرد.

اگر روش سوم را انتخاب کنیم قانون شیب نزول به صورت زیر به دست می‌آید (تمرین ۸,۱):

$$\Delta w_j = \eta \sum_{x \in k\ nearest\ nbrs\ of\ x_q} K\left(d(x_q, x)\right)\left(f(x) - \hat{f}(x)\right) a_j(x) \qquad (8.7)$$

توجه داشته باشید که تنها تفاوت بین این قانون جدید و قانون قبل در رابطه‌ی ۸,۶ این است که وزن هر نمونه آموزشی در مقدار $K\left(d(x_q, x)\right)$ ضرب شده است و خطا نیز فقط روی k نمونه‌ی نزدیک‌تر محاسبه می‌شود. در واقع اگر می‌خواهیم از توابع خطی برای تخمین تابع هدف بر روی دسته‌ی ثابتی از نمونه‌های آموزشی استفاده کنیم، متدهای مؤثرتری نیز وجود دارد تا مسئله را از راه مستقیم برای به دست آوردن ضرایب $w_0 \ldots w_n$ حل کنیم. (Atkeson 1997a) و (Bishop 1995) بررسی‌ای روی انواع این نوع متدها انجام داده‌اند.

۸,۳,۲ نکاتی در مورد برازش وزن‌دار محلی

در بالا تخمین تابع f را با استفاده از تابعی خطی در همسایگی نمونه‌ی جدید x_q بررسی کردیم. ادبیات برازش وزن‌دار محلی، شامل طیف وسیعی از متدهای جایگزین برای وزن دهی بر اساس فاصله‌ی نمونه‌های آموزشی است، و همچنین طیف وسیعی از متدهای تقریب محلی برای تابع هدف موجود است. در اکثر موارد، تابع هدف با یک تابع ثابت، خطی و یا درجه دو تخمین زده می‌شود. به دو دلیل از توابع پیچیده‌تر استفاده نمی‌شود: (۱) هزینه‌ی سازگار کردن توابع پیچیده‌تر برای نمونه‌های جدید به شدت زیاد است و (۲) این تخمین‌های ساده تابع هدف را در یک محدوده کوچک از فضای نمونه‌ای خیلی خوب مدل‌سازی می‌کنند.

۸,٤ توابع پایه‌ای شعاعی

یکی از روش‌های تخمین توابع که بسیار مشابه برازش وزن‌دار محلی و شبکه‌های عصبی است یادگیری با توابع پایه‌ای شعاعی[۱] است (Powell 1987; Broomhead and Lowe 1988; Moody and Darken 1989). در این روش تابع یاد گرفته شده تابعی به فرم زیر است.

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x)) \qquad (8.8)$$

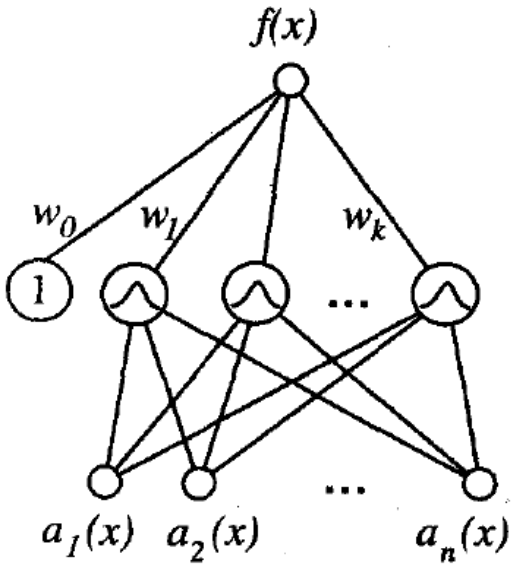
[۱] Radial basis functions

در این رابطه هر x_u نمونه‌ای از X است که در آن $K_u\big(d(x_u,x)\big)$ با افزایش $d(x_u,x)$ کاهش می‌یابد. در اینجا k ثابتی است که توسط کاربر تعیین می‌شود و تعداد توابع هسته‌ی مشمول را مشخص می‌کند. حتی اگر $\hat{f}(x)$ تخمینی جهانی‌ای از f باشد، تأثیر هر یک از این $K_u(d(x_u,x))$ به منطقه‌ی نزدیک به نقطه‌ی x_u محدود می‌شود. معمولاً تابع $K_u(d(x_u,x))$ را تابع گوسی (جدول ۵,۴) با میانگین x_u و واریانس σ_u^2 در نظر می گیرند.

$$K_u\big(d(x_u,x)\big)=e^{\frac{1}{2\sigma_u^2}d^2(x_u,x)}$$

در اینجا بحثمان را محدود به چنین توابع هسته‌ی گوسی می‌کنیم. همان‌طور که در (Harman 1990) نیز آمده است فرم تابعی رابطه‌ی ۸,۸ می‌تواند با افزایش k به میزان لازم و تغییر σ^2 در هر یک از این توابع به طور دلخواه، هر تابعی را تخمین بزند.

تابعی که در رابطه‌ی ۸,۸ آمده است را می‌توان مشابه یک شبکه‌ی دو لایه دانست که در آن لایه‌ی اول مقادیر مختلف $K_u\big(d(x_u,x)\big)$ را محاسبه می‌کند و لایه‌ی دوم ترکیبی خطی از این مقادیر خروجی را ایجاد می‌کند. مثالی از تابع پایه‌ای شعاع (RBF) در شکل ۸,۲ نشان داده شده است.



شکل ۸,۲ تابع شبکه‌ی پایه‌ای شعاعی.

هر واحد پنهان یک تابع فعالیت[۲] را که گوسی با میانگین x_u است محاسبه می‌کند. بنابراین، اگر مقدار x به x_u نزدیک نباشد مقدار تابع فعالیت تقریباً صفر خواهد بود. هر واحد خروجی ترکیبی خطی از توابع فعالیت لایه‌ی پنهان است. با وجود اینکه شکل نشان داده شده فقط یک خروجی دارد، اما شبکه‌هایی را می‌توان ساخت که چندین خروجی داشته باشند.

[۲] Activation function

با معلوم بودن مجموعه‌ی نمونه‌های آموزشی، شبکه‌ی RBF در فرایندی دو مرحله‌ای آموزش داده می‌شود. ابتدا تعداد واحدهای پنهان k معلوم و هر واحد پنهان u با تعیین مقدار x_u و σ_u^2 تعریف می‌گردد. سپس، وزن‌های w_u با استفاده از رابطه‌ی خطای جهانی ۸,۵ برای ماکزیمم کردن تناسب[۳] شبکه بر روی داده‌های آموزشی آموزش داده می‌شوند، چون توابع هسته ثابت نگه داشته شده‌اند، در این مرحله تعیین کردن وزن‌های خطی w_u می‌تواند بسیار مؤثر باشد.

روش‌های جایگزین بسیاری برای تعیین تعداد واحدهای لایه‌ی پنهان یا همان تعداد توابع هسته ارائه شده است. یکی از این روش‌ها در نظر گرفتن یک تابع هسته‌ی گوسی‌ای با میانگین x_i برای هر نمونه‌ی $< x_i, f(x_i) >$ است. می‌توان برای همه‌ی این توابع هسته میزان پهنای یکسانی را در نظر گرفت (واریانس‌ها را مساوی گرفت). یکی از مزیت‌های این روش تناسب کامل شبکه‌ی RBF با داده‌های آموزشی است. به عبارت دیگر، می‌توان برای هر دسته‌ی دلخواه آموزشی m نمونه‌ای با تعیین مناسب $w_0, \ldots, w_m$ توابع گوسی را طوری ترکیب کرد که برای تمامی نمونه‌های $< x_i, f(x_i) >$ داشته باشیم، $\hat{f}(x) = f(x_i)$.

روش دوم، انتخاب مجموعه‌ای از توابع هسته‌ای است که تعدادشان از تعداد نمونه‌های آموزشی کمتر است. این روش مخصوصاً زمانی که تعداد نمونه‌های آموزشی زیاد است بسیار مؤثرتر از روش اول است. مراکز توابع هسته می‌توانند توزیع یکنواختی بر روی فضای نمونه‌ای X داشته باشد. یا در مقابل، ممکن است بخواهیم مراکز توابع هسته را به طور غیریکنواخت در فضای نمونه‌ای پخش کنیم، معمولاً زمانی که خود نمونه‌های آموزشی توزیع غیریکنواخت در فضای نمونه‌ای دارند چنین روش مورد استفاده قرار می‌گیرد. در چنین شرایطی می‌توان مراکز توابع هسته را زیرمجموعه‌ای تصادفی از نمونه‌های آموزشی قرار داد، با این کار از توزیع احتمال حاکم بر توزیع نمونه‌ها نمونه‌برداری می‌کنیم. یا ممکن است خوشه‌های نمونه‌ها[۴] را تشخیص داده و تابع هسته‌ای با مرکز هر کدام را در نظر بگیریم. تعیین مکان توابع هسته به این صورت را می‌توان با استفاده از الگوریتم‌های خوشه یابی[۵] که با نمونه‌های آموزشی (و نه مقادیر هدفشان) را با ترکیبی از توابع گوسی تخمین می‌زنند انجام داد. الگوریتم EM، که در بخش ۶,۱۲,۱ بررسی شد، الگوریتمی برای پیدا کردن ترکیبی از k تابع گوسی برای تناسب بهینه با نمونه‌های مشاهده شده را ارائه می‌کند. در الگوریتم EM، میانگین‌ها طوری انتخاب می‌شوند که احتمال مشاهده‌ی نمونه‌های x_i به شرط داشتن میانگین‌های تخمینی حداکثر شود. توجه دارید که مقادیر تابع هدف $f(x_i)$ در محاسبات مراکز توابع هسته در متدهای خوشه یابی درگیر نمی‌شود و تنها تأثیر مقادیر تابع هدف $f(x_i)$ در این حالت مشخص کردن وزن‌های لایه‌ی خروجی است.

به طور خلاصه، شبکه‌های توابع پایه‌ای شعاعی، تخمینی جهانی با ترکیب خطی توابع هسته‌ی محلی برای تابع هدف ایجاد می‌کنند. مقدار توابع هسته فقط زمانی ناچیز نیست که نمونه‌ی x در ناحیه تعریف شده با مرکز و عرض باشد. بنابراین به شبکه‌های توابع پایه‌ای شعاعی می‌توان به دید ترکیب خطی هموار تعداد زیادی تخمین محلی از تابع هدف نگاه کرد. یکی از مزیت‌های کلیدی شبکه‌های RBF این است که این شبکه‌ها را می‌توان راحت‌تر از شبکه‌های feedforward با الگوریتم Backpropagation آموزش داد. این حقیقت از این رو است که لایه‌ی اول و لایه دوم شبکه‌های RBF به طور جداگانه آموزش داده می‌شوند.

---

[۳] fitness

[۴] prototypical clusters of instances

[۵] clustering algorithms

۸,۵ استدلال مبتنی بر شرایط

روش‌های مبتنی بر شرایط[۶] مثل k-Nearest Neighbor و برازش وزن‌دار محلی در سه ویژگی کلیدی مشترک‌اند. ابتدا اینکه این متدها تنبل‌اند[۷]، بدین معنا که تعمیم بر روی نمونه‌های آموزشی را به زمانی که یک نمونه‌ی جدید ارائه می‌شود موکول می‌کنند. دوم اینکه نمونه‌ی جدید را بر اساس نمونه‌های مشابه دسته‌بندی می‌کنند و با نمونه‌های متفاوت با نمونه‌ی جدید کاری ندارند. سوم اینکه به نمونه‌ها به شکل نقاطی در فضای n بعدی اقلیدسی نگاه می‌کنند. استدلال مبتنی بر شرایط (CBR) [۸] دو ویژگی اول را دارد، اما در ویژگی سوم مشابه دو متد قبلی نیست. در CBR، معمولاً نمونه‌ها با توضیحاتی غنی‌تر نمایش داده می‌شوند و متدهای استخراج نمونه‌های مشابه نیز استادانه‌تر هستند. CBR در مسائلی چون طراحی مفهومی قطعات مکانیکی بر اساس پایگاه داده‌ای از طراحی‌های قبلی (Sycara et al. 1992) ، استدلال درباره‌ی پرونده‌های قانونی بر اساس حکم‌های قبلی (Ashley 1990) ، و حل مسائل برنامه‌ریزی بر اساس استفاده‌ی دوباره و با ترکیب حل‌های قبلی مسائل مشابه (Veloso 1992) به کار گرفته شده است.

بیایید بحث را با مطرح کردن یک فرم کلی از مسائل مبتنی بر شرایط شروع کنیم. سیستم CADET (Sycara et al. 1992) از استدلال مبتنی بر شرایط برای همکاری در طراحی مفهومی قطعات مکانیکی ساده چون شیر آب استفاده می‌کند. این سیستم از پایگاه داده‌ای با حدود ۷۵ طراحی قطعه طراحی قبلی برای پیشنهاد دادن طراحی مفهومی متناسب با خاصیت‌های مسئله‌ی طراحی جدید استفاده می‌کند. هر نمونه‌ی ذخیره شده در حافظه (برای مثال، یک لوله‌ی آب) با دو ویژگی ساختار و قابلیت‌ها توصیف شده است. مسائل طراحی جدید معلوم کردن ساختار با دانستن قابلیت‌های لازم خواهد بود. این تعریف مسئله در شکل ۸,۳ نشان داده شده است. نیمه‌ی بالایی شکل توصیف یک نمونه‌ی ذخیره شده در حافظه به نام لوله‌ی اتصال T شکل[۹] را نشان می‌دهد. قابلیت‌های این قطعه با روابط بین جریان‌های آب و دمای ورودی و خروجی نمایش داده شده است. در نمایش قابلیت‌ها در سمت راست علامت + به این معناست که متغیر در سمت انتهای پیکان با افزایش متغیر سمت ابتدای پیکان افزایش خواهد یافت. برای مثال، جریان آب خروجی Q_3 با افزایش جریان آب ورودی Q_1 افزایش می‌یابد. به طور مشابه، علامت – به این معناست که متغیر انتهای پیکان با افزایش متغیر ابتدای پیکان کاهش می‌یابد. این نوع قابلیت‌ها رفتار لازم برای نوعی شیر آب را مشخص می‌کند. در اینجا Q_c نشان‌دهنده‌ی جریان آب سرد و Q_h جریان آب گرم به شیر و Q_m جریان ترکیبی خروجی شیر را نشان می‌دهد. به طور مشابه T_c و T_h و T_m دمای آب سرد، گرم و ترکیبی را نشان می‌دهند. متغیر C_t سیگنال کنترل دمای آب خروجی شیر و C_f سیگنال کنترل جریان خروجی شیر است. توجه دارید که کنترل‌های C_t و C_f بر جریان‌های Q_c و Q_h تأثیر می‌گذارند، و به طور غیرمستقیم بر جریان Q_m و دمای خروجی آب شیر T_m تأثیر دارند.

با معلوم بودن قابلیت انتظاری طراحی جدید، CADET در پایگاه داده‌ی خود به دنبال نمونه‌هایی با قابلیت‌های مشابه می‌گردد. اگر نمونه‌ای دقیقاً با قابلیت انتظاری هماهنگی داشت، پس می‌توان از همان طراحی برای حل مسئله استفاده کرد، پس طراحی نمونه به عنوان راه‌حل پیشنهادی سیستم ارائه می‌شود. اگر هیچ نمونه‌ای قابلیت دقیقاً با قابلیت‌های انتظاری وجود نداشت CADET حالت‌هایی را که گراف قابلیت‌هایشان با قسمتی از گراف قابلیت انتظاری تطابق داشته باشد پیدا خواهد کرد. برای مثال در شکل ۸,۳ قابلیت‌های اتصال T شکل با یکی

---

[۶] Cased-Based
[۷] lasy
[۸] Cased-Based reasoning
[۹] T-junction pipe

از زیر گراف‌های قابلیت‌های شیر مورد نظر تطابق دارد. در حالت کلی‌تر، CADET به دنبال زیر گراف‌هایی هم‌شکل[۱۰] بین در گراف قابلیت می‌گردد، بنابراین ممکن است قسمت‌هایی از یک قابلیت انتظاری ممکن است با قسمت‌های از یک طراحی خاص تطابق داشته باشد. علاوه بر این، سیستم می‌تواند به طرز استادانه‌ای قابلیت‌های اصلی را برای پیدا کردن گراف‌هایی با قابلیت‌های معادل تغییر دهد تا بتواند طراحی‌های بیشتری را که با گراف معادل تطابق دارند پیدا کند. این سیستم از دانش کلی تأثیرات فیزیکی برای ایجاد چنین گراف‌های قابلیت معادلی استفاده می‌کند. برای مثال، این سیستم از قانون بازنویسی‌ای استفاده می‌کند که عبارت

$$A \overset{+}{\rightarrow} B$$

را به عبارت

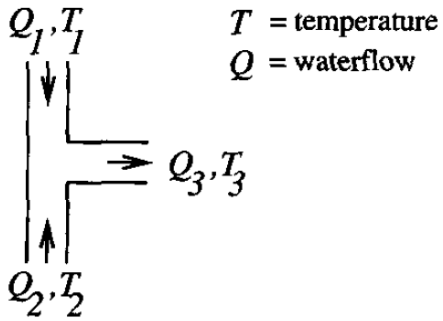
$$A \overset{+}{\rightarrow} x \overset{+}{\rightarrow} B$$

تبدیل می‌کند. این قانون بازنویسی را می‌توان به فرم اینکه "اگر A باید با افزایش B افزایش بیابد، کافی است که کمیتی مثل x را پیدا کنیم که B با افزایش x افزایش بیابد و x نیز با افزایش A افزایش بیابد. در اینجا x کمیتی جهانی است که مقدارش در تطابق گراف قابلیت‌های انتظاری با نمونه‌های پایگاه داده محدود است. در حقیقت گراف قابلیت‌های انتظاری شیر در شکل ۸,۳ همان قابلیت‌های اصلی است که با این قانون بازنویسی شده است.

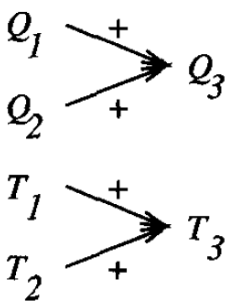
با بازیابی نمونه‌هایی که با زیر گراف‌هایی از قابلیت‌های انتظاری تطابق دارند، طراحی کلی را می‌توان کنار هم قرار داد. در کل، فرایند ایجاد راه‌حل نهایی از چندین نمونه‌ی بازیابی شده می‌تواند بسیار پیچیده باشد. این فرایند ممکن است علاوه بر ترکیب نمونه‌های بازیابی شده نیاز به طراحی قسمت‌هایی از سیستم از قوانین اولیه نیاز داشته باشد. همچنین ممکن است نیاز به بازگشت به انتخاب‌های قبلی طراحی زیر هدف‌ها، و متعاقباً رد کردن نمونه بازیابی شده داشته باشد. CADET قابلیت‌های بسیار محدودی در ترکیب و تجزیه‌ی نمونه‌های بازیابی شده دارد و بنابراین در این مرحله از فرایند به شدت به کاربر وابسته است. همان‌طور که در (Sycara et al. 1992) نیز توصیف شده، CADET یک سیستم کلی تحقیقاتی برای کاوش نقش بالقوه‌ی استدلال مبتنی بر شرایط در طراحی مفهومی است. این سیستم الگوریتم‌های لازم برای تغییر طراحی‌های اولیه و رسیدن به طراحی نهایی را شامل نمی‌شود.

A stored case: T–junction pipe

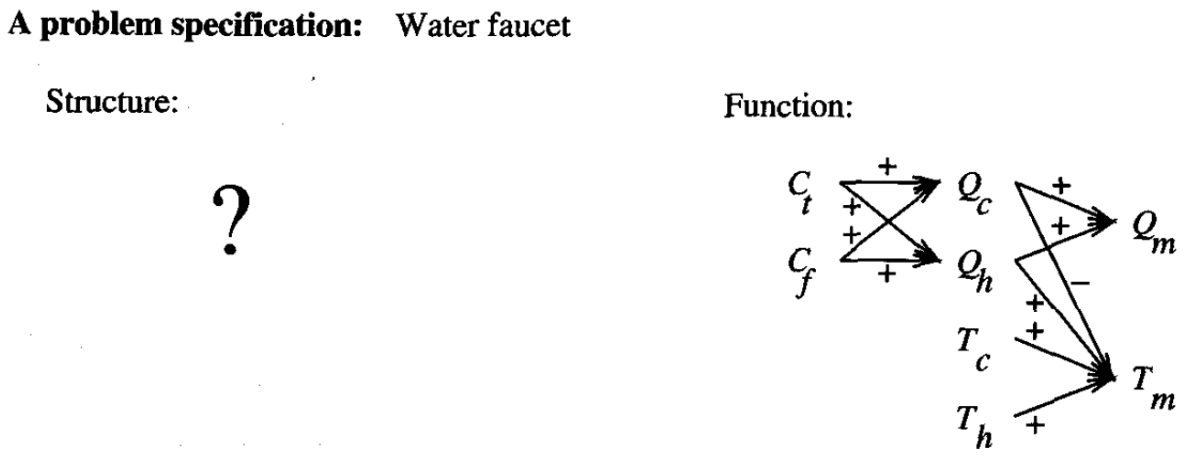
Structure:



Function:



[۱۰] isomorphism



شکل ۸,۳ یک نمونه‌ی ذخیره شده و یک مسئله‌ی جدید.
نیمه‌ی بالای یک طراحی جزئی را در پایگاه داده‌ی CADET نشان می‌دهد. کارایی‌ها با نمودار وابستگی کمیت‌ها در میان متغیرهای اتصال T شکل (که در متن توضیح داده شده) نمایش داده شده است. نیمه‌ی پایین شکل نمونه‌ای از مسئله‌ی معمول طراحی را نشان می‌دهد.

بررسی تناظر بین تعریف مسئله‌های CADET و تعریف مسئله‌ی کلی متدهای چون k-Nearest Neighbor می‌تواند بسیار مفید باشد. در CADET هر نمونه‌ی آموزشی ذخیره شده گراف قابلیت و ساختاری که این قابلیت را عملی می‌کند در برمی‌گیرد و نمونه‌های جدید گراف‌های قابلیت جدید هستند. بنابراین، می‌توان تعریف مسئله‌ی CADET را با تعریف X به عنوان فضای تمامی گراف‌های قابلیت به شکل نمادگذاری استانداردمان بیان کرد. تابع هدف f این گراف‌های قابلیت را به ساختارهای عملی آن‌ها تبدیل می‌کند. پس هر نمونه‌ی ذخیره شده به فرم $< x, f(x) >$ را می‌توان با گراف قابلیت x و ساختار f(x) که ساختار عملی x است نمایش داد. این سیستم باید با استفاده از نمونه‌های آموزشی یاد بگیرد تا ساختار خروجی $f(x_q)$ را برای قابلیت انتظاری x_q پیدا کند.

این طرح سیستم CADET تعداد زیادی از خواص عمومی سیستم‌های استدلال مبتنی بر شرایط را که آن‌ها را از الگوریتم‌های چون k-Nearest Neighbor تمییز می‌دهند نشان می‌دهد.

- نمونه‌ها یا حالات[۱۱] را ممکن است با توضیحات نمادین غنی، مثل گراف‌های قابلیت در CADET نمایش دهیم. این سیستم نیز به معیاری استاندارد مشابه فاصله‌ی اقلیدسی، مثل اندازه‌ی بزرگ‌ترین زیر گراف مشترک بین دو گراف قابلیت، برای تعیین تشابه بین نمونه‌ها نیاز دارد.
- ممکن است برای ایجاد راه‌حل جدید از ترکیب نمونه‌های بازیابی شده استفاده کنیم. این خاصیت مشابه روش k-Nearest Neighbor است، در k-Nearest Neighbor نیز از نمونه‌های مشابه برای ساخته دسته‌بندی نمونه‌ی جدید استفاده می‌شود. با این وجود، این فرایند در استدلال مبتنی بر شرایط ممکن است بسیار پیچیده باشد و نیاز به استدلال مبتنی بر دانش به جای متدهای آماری خواهد داشت.
- ممکن است رابطه‌ی نزدیکی بین بازیابی نمونه‌ها، استدلال مبتنی بر دانش و حل مسئله وجود داشته باشد. یکی از مثال‌های ساده‌ی چنین ارتباط نزدیکی در CADET دیده می‌شود، این سیستم از دانش کلی درباره‌ی تأثیرات برای بازنویسی گراف‌های قابلیت در تلاشش برای پیدا کردن نمونه‌های مشابه استفاده می‌کند. سیستم‌های دیگری نیز طراحی شده‌اند که کاملاً استدلال مبتنی بر شرایط را

[۱۱] cases

به سیستم‌های حل مسئله‌ی مبتنی بر جستجو[۱۲] تبدیل می‌کنند.Anapron (Golding and Rosenbloom 1991) و Prodigy/Analogy (Veloso 1992) چنین سیستم‌هایی هستند.

به طور خلاصه استدلال مبتنی بر شرایط متد یادگیری مبتنی بر نمونه‌هاست که نمونه‌ها در آن با توضیحات غنی نسبی بیان شده و بازیابی و ترکیب نمونه‌ها برای حل نمونه‌ی جدید وابسته به دانش قبلی و متدهای قوی جستجویی حل مسئله باشد. مشکلات فعلی استدلال مبتنی بر شرایط ایجاد متدهای بهینه‌ی فهرست بندی نمونه‌هاست. مشکل اصلی معیار تشخیص تشابهات (برای مثال، زیر گراف‌های مشابه در گراف‌های قابلیت) که تخمینی از رابطه‌ی نمونه جزئی با مسئله‌ی جزئی است. زمانی که سیستم CBR برای استفاده از نمونه‌های بازیابی شده تلاش می‌کند ممکن است مشکلات ناشی از این معیار تشابه را نپوشاند. برای مثال در CADET تکه طراحی‌های بازیابی شده ممکن است با یکدیگر در تضاد باشد، و در نتیجه دیگر ترکیب و ساخت طراحی سازگار غیرممکن خواهد بود. در کل زمانی که چنین اتفاقی می‌افتد، سیستم CBR می‌تواند بازگشته و به دنبال نمونه‌های دیگری در میان نمونه‌های موجود بگردد، یا مسئله را به متد حل مسئله‌ای دیگر واگذار کند. زمانی که چنین مشکلاتی تشخیص داده می‌شوند، نمونه‌های آموزشی برای بهبود معیار تشابه، یا به طور معادل، فهرست بندی ساختار در پایگاه داده فراهم می‌شود. در کل زمانی که نمونه‌ای بر اساس معیاری بازیابی می‌شود، اما بر اساس بررسی‌های بعدی نامربوط تشخیص داده می‌شود، باید معیار طوری بازبینی شود که این نمونه را در جستجوهای آینده رد کند.

۸,۶ نکاتی در مورد یادگیری‌های تنبل و کوشا[۱۳]

در این فصل سه متد تنبل را بررسی کردیم: k-nearest neighbors، برازش وزن‌دار محلی و استدلال مبتنی بر شرایط. این متدها برای اینکه چگونگی تعمیم روی نمونه‌های آموزشی را به زمانی که نمونه‌ی جدید ارائه می‌شود واگذار می‌کنند، تنبل خوانده می‌شوند. همچنین متدهای کوشایی را نیز بررسی کردیم: متدهایی که برای یادگیری شبکه‌های توابع پایه‌ای شعاعی استفاده می‌شود. این متدها را چون تعمیم را قبل از مواجهه با نمونه‌ی جدید انجام می‌دهند کوشا می‌نامیم، این تعمیم با ساختار شبکه و وزن‌های تعریف شده برای تخمین تابع هدف انجام می‌شود. به این ترتیب تمامی الگوریتم‌های معرفی شده در این کتاب (مثل، Backpropagation و C4.5) الگوریتم‌های یادگیری کوشا هستند.

آیا آنچه الگوریتم‌های تنبل می‌توانند یاد بگیرند با آنچه الگوریتم‌های کوشا می‌توانند یاد بگیرند تفاوت چشم‌گیری دارد؟ بیایید ابتدا دو نوع تفاوت را مشخص کنیم: تفاوت در زمان محاسبات و تفاوت در دسته‌بندی‌های تولید شده برای نمونه‌های جدید. تفاوت‌های واضحی در زمان محاسبه‌ی الگوریتم‌های یادگیری تنبل و کوشا وجود دارد. برای مثال، متدهای تنبل در طول آموزش محاسبات کمتری لازم دارند، اما در هنگام پیش‌بینی ویژگی هدف برای نمونه‌ی جدید محاسبات زیادی انجام می‌دهند.

سؤال اساسی‌تر این است که آیا تفاوت‌های اساسی‌ای در بایاس‌های استقرایی الگوریتم‌های تنبل و الگوریتم‌های کوشا وجود دارد. از این نظر تفاوت‌های زیر بین متدهای تنبل و کوشا وجود دارد.

- در متدهای تنبل گاهی تصمیم‌گیری برای چگونگی تعمیم بر روی داده‌های آموزشی D به نمونه‌ی آموزشی ارائه شده نیز وابسته می‌شود.

---

[۱۲] search-based

[۱۳] eager

- متدهای کوشا این وابستگی را نمی‌توانند داشته باشند، زمانی که یک متد کوشا با یک نمونه‌ی جدید مواجه می‌شود، تخمین جهانی را انجام داده است.

آیا این تمایز دقت تعمیم یادگیر را تحت تأثیر قرار می‌دهد؟ اگر دو یادگیر کوشا و تنبل از فضای فرضیه‌ای یکسانی مثل H استفاده‌کننده این تمایز تأثیرگذار می‌شود. برای تصور، فرض کنید فضای فرضیه‌ای تمام توابع خطی است. اگر از متد برازش وزن‌دار محلی که قبلاً مطرح شد برای این فضای فرضیه استفاده کنیم، برای هر نمونه‌ی جدید x_q برای تعمیم رو نمونه‌های آموزشی این متد فرضیه‌ای را انتخاب می‌کند که نزدیک x_q باشد. در نقطه‌ی مقابل، یک یادگیر کوشا که از همان فضای فرضیه‌ای توابع خطی استفاده می‌کند، تخمین خود را از تابع هدف قبل از مواجهه با نمونه‌های جدید مشخص می‌کند. بنابراین یادگیر کوشا فقط بر اساس یک تابع خطی نمونه‌های جدید و کل فضای نمونه‌ای را دسته‌بندی می‌کند. پس متدهای تنبل که از فضای فرضیه‌ای شاملی استفاده می‌کنند زیرا که با استفاده از مینیمم‌های موضعی توابع خطی برای تشکیل تخمینشان از تابع هدف استفاده می‌کنند. توجه داشته باشید که این شرایط برای یادگیرهای دیگر و فضای فرضیه‌های دیگر نیز صادق است. برای مثال نسخه‌ی تنبل Backpropagation می‌تواند برای هر نمونه‌ی جدید یک شبکه‌ی عصبی یاد بگیرد، اما نسخه‌ی کوشا (فصل ۴) فقط یک شبکه‌ی عصبی برای کل نمونه‌ها یاد خواهد گرفت.

نکته‌ی کلیدی در پاراگراف بالا این است که یادگیر تنبل می‌تواند با ترکیب تخمین‌های موضعی تابع هدف را یاد گیرد، در حالی که یادگیر کوشا فقط یک تخمین جهانی را با توجه به نمونه‌های آموزشی یاد می‌گیرد. این تفاوت بین یادگیری کوشا و تنبل به تفاوت بین تخمین موضعی و جهانی تابع هدف بر می‌گردد.

آیا می‌توان متدهای کوشایی ساخت که از تخمین‌های موضعی استفاده کند؟ شبکه‌های RBF تلاشی برای دست‌یابی به چنین متدهایی است. متدهای یادگیری RBF که ما بررسی کردیم تخمین جهانی از تابع هدف را در زمان یادگیری ایجاد می‌کنند، با این وجود، یک شبکه‌ی RBF این تابع جهانی را به صورت ترکیب خطی چندین تابع هسته‌ی موضعی بیان می‌کند. اما چون یادگیری RBF باید قبل از مواجهه با نمونه‌ی جدید این فرضیه را مشخص کند، این تخمین‌های موضعی مشابه توابع موضعی یادگیر تنبل که مخصوصاً برای نمونه‌ی جدید ساخته شده‌اند، منحصراً برای نمونه‌ی جدید ساخته نشده‌اند. در عوض، شبکه‌های RBF به صورت کوشا از توابع موضعی که در حوالی نمونه‌های آموزشی یا دسته نمونه‌های آموزشی ایجاد می‌شود، تشکیل شده‌اند، اما این تخمین موضعی‌ای حوالی نمونه‌ی مجهول جدید ساخته نمی‌شود (بدین مفهوم که هر تخمین منحصر به نمونه‌ی جدید نیست و برای تمامی نمونه‌های جدید ارائه می‌گردد).

به طور خلاصه، متدهای تنبل حق انتخاب بین فرضیه‌ها یا تخمین‌های موضعی تابع هدف برای هر نمونه‌ی جدید دارند. در حالی که متدهای کوشا محدودترند و باید با استفاده از یک فرضیه باید کل فضای نمونه‌ای را پوشش دهند. البته متدهای کوشا نیز می‌توانند از ترکیبی از تخمین‌های موضعی استفاده کنند (مثل شبکه‌های RBF). با این وجود، حتی این ترکیب تخمین‌های موضعی قابلیت کامل متدهای تنبل در تغییر بر اساس نمونه‌ی آموزشی مجهول را به یادگیر کوشا نمی‌دهد.

۸,۷ خلاصه و منابع برای مطالعه‌ی بیشتر

نکات اصلی این فصل شامل موارد زیر می‌شود:

- متدهای یادگیری مبتنی بر نمونه‌ها با دیگر روش‌های تخمین توابع از این جهت متفاوت‌اند که پردازش مربوطه به نمونه‌های آموزشی را به زمانی که لازم است نمونه‌ی جدیدی دسته‌بندی شود به تعویق می‌اندازند. در نتیجه، نیازی ندارند که فرضیه را به طور صریح در تمامی فضای نمونه‌ای مستقل از نمونه‌ی جدید بیان کنند. در مقابل برازش محلی از تابع هدف برای هر نمونه به ما می‌دهند.
- مزیت متدهای مبتنی بر نمونه شامل قدرت مدل‌سازی توابع هدف پیچیده با استفاده از تقریب‌هایی با پیچیدگی کمتر و اینکه داده‌های نمونه‌های آموزشی هیچ‌گاه از دست نخواهند رفت (نمونه‌ها به طور صریح ذخیره خواهند شد) است. مشکل عملیاتی اصلی پرهزینه بودن دسته‌بندی نمونه‌های جدید (مخصوصاً زمانی که نمونه‌ها با خواص سمبولیک پیچیده توصیف می‌شوند) و تأثیر منفی ویژگی‌های نامربوط است.
- الگوریتم k-nearest meighbor الگوریتمی مبتنی بر نمونه‌ها برای تقریب توابع هدف حقیقی مقدار گسسته یا پیوسته است با این فرض که نمونه‌ها متناسب با فضای اقلیدسی n بعدی‌اند. مقدار هدف یک نمونه‌ی جدید با مقادیر k نزدیک‌ترین نمونه‌ی آموزشی تقریب زده می‌شود.
- متدهای برازش وزن‌دار محلی یک تعمیم از الگوریتم k-nearest meighbor اند با این فرض که برای هر نمونه‌ی جدید تقریبی خطی ایجاد می‌گردد. برازش محلی تابع هدف ممکن است بر پایه‌ی فرم‌های مختلفی از توابع مثل ثابت، خطی، درجه دو و یا توابع هسته‌ی محدود در فضا باشد.
- شبکه‌های RBF نوعی از شبکه‌های عصبی هستند که از توابع هسته‌ی محدود در فضا ساخته می‌شوند. این شبکه‌ها را می‌توان مخلوطی از روش‌های مبتنی بر نمونه‌ها (تأثیر محدود در فضای هر تابع هسته) و روش شبکه‌های عصبی (تقریب کلی تابع هدف بر اساس نمونه‌های آموزشی و در زمان آموزش و نه در هنگام دسته‌بندی نمونه‌های جدید) دانست. استفاده از شبکه‌های عصبی RBF به طور موفق در کاربردهای از جمله تشخیص تصویر که در آن بررسی محلی‌ای کاملاً توجیه شده است به کار رفته است.
- استدلال مبتنی بر حالت نیز نوعی روش مبتنی بر نمونه است که در آن نمونه‌ها توسط توضیحات پیچیده‌ی منطقی به جای نقاط فضای اقلیدسی نمایش داده می‌شوند. با این توصیفات نمادین پیچیده‌ی نمونه‌ها، متدهای بسیار و غنی‌ای برای نگاشت نمونه‌های آموزشی به مقادیر توابع هدف پیشنهاد شده است. متدهای استدلال مبتنی بر حالت در بسیاری از کاربردها مثل مدل‌سازی استدلال قانونی و برای راهنمایی جستجو در تولیدهای پیچیده و مسائل حمل نقل به کار رفته است.

الگوریتم k-nearest neighbor از جمله الگوریتم‌های یادگیری ماشینی است که کاملاً مورد تحلیل و بررسی قرار گرفته، دلیل این بررسی‌ها سادگی و قدمت این الگوریتم است. Cover and Hart (1967) نتایج تئوری اولیه را مطرح می‌کنند، Duda and Hart (1973) دید کلی خوبی از این الگوریتم ارائه می‌کند. Bishop (1995) بحثی در مورد الگوریتم k-nearest neighbor مطرح کرده و رابطه‌ی آن را با تخمین چگالی توزیع احتمال را بررسی می‌کند. تحقیق جدید خوبی که درباره‌ی متدهای برازش خطی محلی در Atkeson et al. (1997) انجام شده است. کاربرد این متدها در کنترل ربات نیز توسط Atkeson et al. (1997b) انجام شده است.

بحث جامعی از توابع پایه‌ای شعاعی در Bishop (1995) انجام گرفته شده است. دیگر کاربردها نیز در Powell (1987) و Poggio and Girosi (1990) بررسی شده‌اند. برای الگوریتم EM بحث شده در این کتاب به قسمت ۶,۱۲ که در آن تخمین میانگین ترکیبی از چندین تابع گوسی آمده رجوع کنید.

Kolodner (1993) نیز معرفی‌ای بر استدلال مبتنی بر حالت انجام می‌دهد. دیگر تحقیقات کلی و مجموعه‌های تحقیقات جدید در Aamodt et al. (1994), Aha et al. (1991), Haton et al. (1995), Riesbeck and Schank (1989), Schank et al. (1994), Veloso and Aamodt (1995), Watson (1995), Wess et al. (1994) آمده است.

تمرینات

۸,۱ قانون شیب نزول را برای یک برازش وزن‌دار محلی که در رابطه‌ی ۸,۱ آمده استخراج کنید.

۸,۲ متد جایگزین زیر را برای فاصله در برازش وزن‌دار محلی در نظر بگیرید. مجموعه‌ای مجازی از نمونه‌های آموزشی به نام D' ایجاد کنید که: برای هر نمونه‌ی آموزشی $<x,f(x)>$ در مجموعه‌ی واقعی D تعداد $K(d(x_q,x))$ کپی در D' قرار دهید. حال تقریبی خطی برای مینیمم کردن معیار خطای زیر انجام دهید:

$$E_4 \equiv \frac{1}{2}\sum_{x\in D'}\left(f(x)-\hat{f}(x)\right)^2$$

ایده‌ی اصلی کپی کردن نمونه‌های آموزشی که بیشتر نزدیک نمونه هستند و کمتر آن‌هایی که دورتر هستند. شیب نزول را برای این معیار استخراج کنید. این قانون را بر حسب مجموع روی اعضای D بیان کنید نه اعضای D' و قانون را با قوانین روابط ۸,۶ و ۸,۷ مقایسه کنید.

۸,۳ نمونه‌ی تنبلی از الگوریتم کوشای $ID3$ پیشنهاد کنید (فصل ۳). مزیت‌ها و مضرت‌های الگوریتم پیشنهادی شما نسبت به الگوریتم کوشای اصلی چیست؟

فرهنگ لغات تخصصی فصل (فارسی به انگلیسی)

ارزیابی	cross-validation
وزن‌دار متناسب با فاصله	distance-weighted
باقیمانده	Residual
برازش	Regression
برازش وزن‌دار محلی	locally weighted regression
تابع پایه‌ای شعاعی	radial basis function
تابع هسته	Kernel function
تشخیص الگو آماری	statistical pattern recognition
تنبل	Lasy
حالات	Cases
طلسم بعد	curse of dimensionality
کوشا	Eager
الگوریتم نزدیک‌ترین همسایه	Nearest neighbor algorithm
الگوریتم‌های خوشه یابی	clustering algorithms
مبتنی بر شرایط	Cased-Based
متد جهانی	global method
متد محلی	local method
نزدیک‌ترین همسایه	Nearest Neighber

نمودار ورونوی مجموعه‌ی نمونه‌های آموزشی	Voronoi diagram
هم‌شکل	Isomorphism
یادگیری مبتنی بر نمونه‌ها	instance based learning