

فصل سوم: یادگیری درخت تصمیم گیری

یادگیری درخت¹ تصمیم گیری یکی از پرکاربردترین و کارآمدترین متدهای یادگیری استقرایی است. این متد در یادگیری توابع گسسته مقدار با داده های خطا دار به کار می رود. در این فصل به خانواده ای از الگوریتم های یادگیری درختی، مثل الگوریتم های ID3، ASSISTANT و C4.5 می پردازیم. این متدهای یادگیری درختی فضای فرضیه ای کاملی را جستجو می کنند و مشکل محدودیت فضای فرضیه ای را ندارند. بایاس های استقرایی این الگوریتم ها این است که همیشه درخت های کوچک تر را بر درخت های بزرگ تر ترجیح می دهند (اصل تیغ Occam).

۳.۱ مقدمه

یادگیری درختی متدی برای تخمین توابع هدف گسسته مقدار است، در یادگیری درختی تابع تخمین زده شده با یک درخت تصمیم گیری مشخص می شود. درخت های بدست آمده را نیز می توان به صورت دسته ای از دستور های if-then نیز نمایش داد تا بررسی آن برای انسان راحت تر گردد. این متدها از جمله متداول ترین متدها در یادگیری های استقرایی هستند و در حوزه ی وسیعی از کارهای یادگیری، از یادگیری تشخیص موارد پزشکی گرفته تا تشخیص میزان ریسک وام، مورد استفاده قرار گرفته اند.

۳.۲ نمایش درخت تصمیم گیری

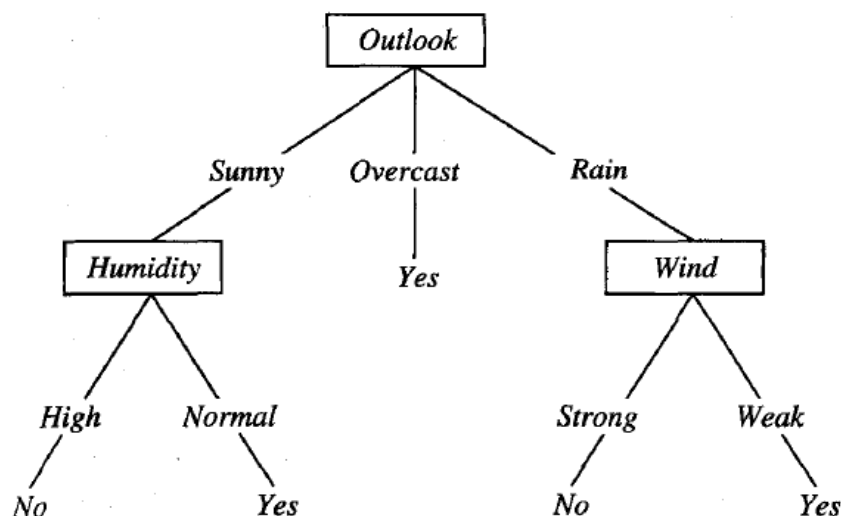
درخت تصمیم گیری با ترتیب کردن نمونه ها از ریشه به سمت برگ های درخت، نمونه ها را دسته بندی می کند. در این درخت هر گره ویژگی ای را در مورد نمونه و هر شاخه (که از آن گره خارج می شود) مقادیر مربوطه ی آن ویژگی را مشخص می کند. برای دسته بندی هر نمونه ابتدا از ریشه شروع می کنیم، به هر ویژگی که می رسیم از شاخه ای از درخت که ویژگی نمونه با آن مطابق است پایین می رویم. این فرایند برای زیر درخت ها نیز ادامه می یابد تا به دسته بندی نمونه برسیم.

¹ tree

شکل ۳.۱ یک مثال از درخت تصمیم گیری را نشان می‌دهد. این درخت تصمیم گیری نشان می‌دهد که مقدار هدف PlayTennis را نشان می‌دهد. برای مثال، نمونه‌ی

<Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong>

در چپ‌ترین گوشه‌ی پایین درخت قرار می‌گیرد، پس بنابراین این نمونه منفی دسته بندی خواهد شد (درخت پیش بینی می‌کند برای این مقادیر PlayTennis مقدار No را داشته باشد). درخت و نمونه‌های آمده در جدول ۲.۳ که برای توضیح الگوریتم یادگیری ID3 مورد استفاده قرار گرفته‌اند از (Quinlar 1986) گرفته شده‌اند.



شکل ۳.۱ درختی تصمیم گیری برای مفهوم PlayTennis.

نمونه‌ها با ترتیب شدن بین شاخه‌های درخت دسته بندی می‌شوند و در انتها مقدار برگ را بر می‌گردانند (در این مثال مقادیر Yes یا No). این درخت نمونه‌ی مذکور را برای مفهوم PlayTennis منفی دسته بندی خواهد کرد. در کل درخت‌های تصمیم گیری روابط فصلی‌ای از عطف شروط را برای دسته بندی نمونه‌ها به کار می‌برند. هر مسیر از ریشه‌ی درخت به سمت برگ‌ها عطفی از روابط در مورد ویژگی‌هاست و کل درخت نیز فصلی از این عطف‌هاست. برای مثال، شکل ۳.۱ متناظر با رابطه‌ی زیر است:

$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})$

$\vee (\text{Outlook} = \text{Overcast})$

$\vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})$

۳.۳ مسائل مناسب برای درخت تصمیم گیری

با وجود اینکه متد های یادگیری درختی زیادی با نیازها و قابلیت‌های متفاوت ارائه شده است، اما اغلب یادگیری درخت تصمیم گیری برای مسائلی با ویژگی‌های زیر مناسب است:

- نمونه‌ها با زوج مرتب دسته ویژگی‌ها و مقدار تابع هدف مشخص شوند. نمونه‌ها در این مسائل با دسته ای از ویژگی‌های ثابت (مثلاً Temperature و ... و مقادیرشان (مثل Hot) مشخص شوند. راحت‌ترین وضعیت برای یادگیری درخت تصمیم گیری حالتی است که هر ویژگی تعداد کمی از مقادیر را بتواند بگیرد (مثلاً فقط Hot، Mild و Cold). با این وجود، با الحاقی به الگوریتم‌های اصلی، که در بخش ۳.۷.۲ بحث خواهد شد، می‌توان ویژگی‌ها را از گسسته مقدار به حقیقی مقدار تغییر داد (مثلاً Temperature را با درجه مشخص کرد).
 - تابع هدف مقادیر، خروجی گسسته داشته باشد. درخت تصمیم گیری شکل ۳.۱ مقادیر منطقی را به هر یک از نمونه‌ها نسبت می‌دهد. متد های یادگیری درختی با افزایش تعداد مقادیر تابع هدف به راحتی به الگوریتم‌های یادگیری توابع گسسته مقدار تعمیم پیدا می‌کنند. با تعمیمی قابل توجه تر می‌توان توابعی با مقادیر حقیقی را با این متدها یاد گرفت، با این وجود استفاده از یادگیری درختی در یادگیری توابع حقیقی متداول نیست.
 - زمانی که هدف یادگیری توضیحات فصلی است. همان طور که پیش‌تر نیز گفته شد، یادگیری درختی ذاتاً روابط فصلی را یاد می‌گیرد.
 - داده های آموزشی می‌توانند خطا داشته باشند. متد های یادگیری درختی می‌توانند خود را با خطای موجود در داده های آموزشی وفق دهند، فرقی ندارد که مقدار تابع هدف نمونه بوده یا یکی از ویژگی‌ها اشتباه گزارش شده باشد.
 - نمونه های آموزشی می‌توانند ویژگی‌های مجهول داشته باشند. یادگیری درختی را حتی زمانی که نمونه های آموزشی ویژگی‌های مجهول دارند می‌توان به کار برد (مثلاً اگر ویژگی Humidity برای بعضی از روزها معلوم نباشد). این حالت در بخش ۳.۷.۴ بررسی خواهد شد.
- بسیاری از مسائل کاربردی دارای ویژگی‌هایی فوقند، به همین خاطر یادگیری درخت تصمیم گیری بسیار پر کاربرد شده است تا جایی که در مسایلی نظیر تشخیص موارد پزشکی، تشخیص دلیل خرابی تجهیزات و تشخیص ریسک وام بر اساس عقب افتادگی قسط‌ها به کار می‌روند. به چنین مسائلی، که هدف از یادگیری دسته بندی نمونه‌ها در یکی از دسته های موجود است، مسائل دسته بندی^۱ می‌گویند.
- در ادامه‌ی این فصل بدین ترتیب بحث را پی می‌گیریم: در بخش ۳.۴ الگوریتم اساسی ID3 را در یادگیری درختی و نحوه‌ی کار آن را توضیح خواهیم داد. در قسمت ۳.۵ جستجوی این الگوریتم در فضای فرضیه ای را بررسی و آن را با الگوریتم‌های فصل ۲ مقایسه خواهیم کرد. در بخش ۳.۶ بایاس‌های استقرایی این الگوریتم یادگیری درختی را بررسی خواهیم کرد و با بایاسی کلی‌تر به نام تیغ Occam^۲ آشنا خواهیم شد که ترجیح درخت‌های کوچک‌تر و ساده‌تر را در میان فضای فرضیه ای توجیه می‌کند. در بخش ۳.۷ پدیده‌ی overfit را بررسی خواهیم کرد و استراتژی‌های هرس^۳ را برای حل این مسئله بیان خواهیم کرد. در این قسمت در مورد مباحث پیشرفته تر دیگری نیز مثل چگونگی تعمیم یادگیری درختی برای یادگیری توابع حقیقی مقدار، یادگیری با ویژگی‌های مجهول و ویژگی‌های غیر هم هزینه نیز بحث شده است.

¹ classification problems

² Occam's razor

³ post-pruning

۳.۴ الگوریتم اساسی یادگیری درختی

اکثر الگوریتم‌هایی که برای یادگیری درختی ایجاد شده نسخه‌های مختلف یک الگوریتم اساسی هستند که از جستجوی حریصانه^۱ و بالا به پایین^۲ برای جستجوی فضای درخت‌های تصمیم‌گیری ممکن استفاده می‌کند. این روش الگوریتم ID3 نام دارد (Quinlan 1986) و تکامل یافته‌ی این الگوریتم نیز C4.5 نامیده می‌شود (Quinlan 1993). این دو الگوریتم موضوع بحث این بخش هستند. در این بخش الگوریتم پایه‌ای یادگیری درختی را معرفی می‌کنیم، این الگوریتم تقریباً همان ID3 است. در قسمت ۳.۷ نیز تعدادی از تعمیم‌های این الگوریتم، تعمیم‌های مربوط به الگوریتم C4.5 و چند الگوریتم دیگر، را توضیح خواهیم داد.

الگوریتم اساسی ما، یا همان ID3، درخت تصمیم‌گیری متناسب را با جستجوی بالا به پایین پیدا می‌کند، این جستجو با طرح این سؤال آغاز می‌شود "چه ویژگی‌ای باید در ریشه‌ی درخت بررسی شود؟" برای جواب دادن به این سؤال، تمامی ویژگی‌ها در تمامی نمونه‌ها توسط یک بررسی آماری بررسی می‌شود تا معلوم گردد تا کدام ویژگی به تنهایی تأثیر بیشتری بر دسته‌بندی نمونه‌ها دارد. سپس بهترین ویژگی انتخاب می‌شود و به عنوان گره ریشه‌ی درخت قرار می‌گیرد. برای هر مقدار این ویژگی انتخابی در ریشه‌ی درخت نمونه‌های آموزشی ترتیب می‌شوند و با توجه به این گره مسئله به مسئله‌های کوچک‌تر تبدیل می‌شود (هر نمونه‌ی آموزشی از طرف شاخه‌ای پایین می‌رود که مقدار آن با مقدار ویژگی نظیرش مطابق باشد). این فرایند برای زیر شاخه‌ها آنقدر اجرا می‌شود تا بالاخره هر نمونه درست دسته‌بندی شود، در هر تکرار همیشه ویژگی انتخابی برای گره ویژگی‌ای است که مهم‌ترین اثر را در دسته‌بندی دارد. آن طور که شرح داده شد، با یک فرایند حریصانه به جستجوی بهترین درخت ممکن می‌پردازیم، یعنی اینکه هیچ وقت الگوریتم به انتخاب‌هایی که قبلاً کرده بازنگری نمی‌کند. ساده شده‌ی این الگوریتم (برای یادگیری توابع منطقی مقدار، یا همان یادگیری مفهوم) در جدول ۳.۱ آمده است.

ID3 (Examples, Target_attribute, Attributes)

Examples مجموعه‌ی تمامی نمونه‌های آموزشی است. Target_attribute ویژگی‌ای است که مقادیرش توسط درخت پیش‌بینی می‌شود. Attributes لیستی از دیگر ویژگی‌هایی است که ممکن است توسط درخت بررسی شود. این الگوریتم درخت تصمیم‌گیری‌ای را که به درستی نمونه‌های داده شده را دسته‌بندی می‌کند بر می‌گرداند.

- گره‌ای برای ریشه‌ی درخت ایجاد کن
 - اگر تمامی نمونه‌های Examples، نمونه‌ی مثبتند ریشه را با + علامت‌گذاری کن و درخت را خروجی بده.
 - اگر تمامی نمونه‌های Examples، نمونه‌ی منفی‌اند ریشه را با - علامت‌گذاری کن و درخت را خروجی بده.
 - اگر مجموعه‌ی Attributes تهی است، ریشه را با متداول‌ترین دسته‌بندی نمونه‌ها علامت‌گذاری کن و درخت را خروجی بده.
 - در غیر این صورت :
 - A را ویژگی‌ای قرار بده که Examples را بهتر* دسته‌بندی می‌کنند.
 - ویژگی متناسب با گره ریشه را A قرار بده.
 - برای هر مقدار v_i از A،
- یک شاخه‌ی جدید در زیر ریشه متناسب با مقدار v_i اضافه کن.

¹ greedy

² top-down

³ label

- $Examples_{v_i}$ را زیر مجموعه ای از $Examples$ قرار بده که مقدار v_i را برای ویژگی A دارند
- اگر $Examples_{v_i}$ تهی بود،

- در زیر شاخه‌ی جدید گره برگی اضافه کن و آن را با متداول‌ترین مقدار $Target_attribute$ در $Examples$ علامت گذاری کن.
- در غیر این صورت، در زیر این شاخه‌ی جدید زیر درخت $ID3(Examples_{v_i}, Target_attribute, Attributes - \{A\})$ را اضافه کن.
- درخت ایجاد شده را برگردان.

※: بهترین ویژگی‌ای است که بالاترین مقدار بهره‌ی اطلاعات (که در رابطه‌ی ۳.۴ آمده) را داشته باشد.

جدول ۳.۱ خلاصه‌ی الگوریتم $ID3$ برای یادگیری توابع منطقی مقدار.

$ID3$ یک الگوریتم حریصانه است که درخت را از بالا به پایین رشد می‌دهد، در هر گره ویژگی‌ای انتخاب می‌شود که نمونه‌های آموزشی در آن ناحیه را بهتر دسته بندی کند. این فرایند آنقدر ادامه پیدا می‌کند تا درخت به طور کامل تمام نمونه‌های آموزشی را درست دسته بندی نماید، یا اینکه تمامی ویژگی‌ها استفاده شوند.

۳.۴.۱ کدام ویژگی بیشترین نقش را در دسته بندی دارد؟

مهم‌ترین انتخابی که در الگوریتم $ID3$ انجام می‌گیرد انتخاب ویژگی‌ای که در هر گره از درخت بررسی می‌شود است. ما ترجیح می‌دهیم که این ویژگی، ویژگی‌ای باشد که بیشترین تأثیر را در دسته بندی نمونه‌ها دارد. چه معیاری را می‌توان معیار خوبی برای برتری یک ویژگی دانست؟ در اینجا خاصیتی آماری به نام بهره‌ی اطلاعات^۱ را تعریف می‌کنیم که میزان تأثیر یک ویژگی را بر دسته بندی نمونه‌ها بر اساس دسته بندی تابع هدفشان اندازه گیری می‌کند. $ID3$ از بهره‌ی اطلاعات برای انتخاب ویژگی در هر مرحله از رشد درخت استفاده می‌کند.

۳.۴.۱.۱ آنروپی، معیار یکدستی نمونه‌ها

برای تعریف دقیق بهره‌ی اطلاعات از تعریف معیار دیگری به نام آنروپی^۲، که در تئوری اطلاعات^۳ کاربرد بسیار دارد، شروع می‌کنیم. این معیار یکدستی^۴ و عدم یکدستی یک دسته‌ی دلخواه از نمونه‌ها را مشخص می‌کند. با داشتن دسته‌ی S از نمونه‌های مثبت و منفی مفهوم هدف، آنروپی دسته‌ی S متناسب با این دسته بندی منطقی به صورت زیر تعریف می‌شود:

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \quad (3.1)$$

در این رابطه p_{\oplus} نسبت تعداد نمونه‌های مثبت به تعداد کل نمونه‌ها و p_{\ominus} نیز نسبت تعداد نمونه‌های منفی به تعداد کل نمونه‌هاست. همیشه در محاسبه‌ی آنروپی فرض می‌کنیم که $0 \log 0$ ، صفر است.

برای درک بهتر، فرض کنید که مجموعه‌ی S شامل ۱۴ نمونه از مفهومی منطقی باشد، از این ۱۴ نمونه ۹ نمونه مثبت و ۵ نمونه منفی هستند (برای خلاصه سازی به طور خلاصه می‌نویسیم $[9+, 5-]$). آنروپی مربوط به این مجموعه‌ی زیر خواهد بود:

$$Entropy([9+, 5-]) = -\left(\frac{9}{14}\right) \log_2 \frac{9}{14} - \left(\frac{5}{14}\right) \log_2 \frac{5}{14}$$

¹ information gain

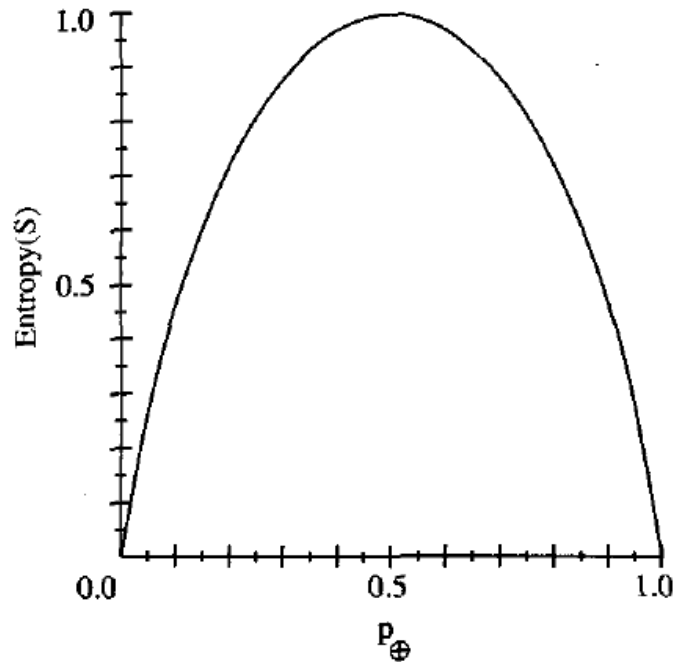
² entropy

³ information theory

⁴ Homogeneity

$$= 0.940 \quad (3.2)$$

توجه داشته باشید که زمانی آنتروپی صفر است که تمامی اعضای S از یک نوع دسته بندی باشند. برای مثال اگر تمامی نمونه‌ها مثبت باشند $(p_{\oplus} = 1)$ پس p_{\ominus} صفر خواهد بود و داریم که $Entropy(S) = -1 \log_2 1 - 0 \log_2 0 = -1 * 0 - 0 \log_2 0 = 0$. همچنین توجه دارید که زمانی آنتروپی ۱ است که تعداد نمونه‌های مثبت و منفی مساوی باشد. همیشه مقدار آنتروپی مقداری بین ۰ و ۱ است. شکل ۳.۲ شکل تابع آنتروپی را برای یک تابع منطقی مقدار بر حسب p_{\oplus} نشان می‌دهد.



شکل ۳.۲ میزان آنتروپی برای دسته بندی منطقی، برچسب مقدار نسبی p_{\oplus} یکی از تفسیرهای آنتروپی که در تئوری اطلاعات مطرح می‌شود حداقل تعداد بیت‌های لازم برای کد کردن یکی از اعضای دلخواه S است (برای مثال یک عضو تصادفی با احتمال یکنواخت). مثلاً اگر p_{\oplus} یک باشد، دریافت کننده‌ی اطلاعات می‌داند که نمونه‌ی انتخابی حتماً مثبت خواهد بود، پس نیازی به ارسال داده‌ای نیست، آنتروپی صفر است. از سوی دیگر، اگر $p_{\oplus} = 0.5$ باشد، برای ارسال هر نمونه دقیقاً ۱ بیت لازم خواهد بود تا برای دریافت کننده معلوم گردد که نمونه مثبت بوده یا منفی. و اگر $p_{\oplus} = 0.8$ باشد، مجموعه‌ای از اعضا را می‌توان با متوسط کمتر از ۱ بیت برای هر عضو کد کرد، در این کد، برای اعضای مثبت کد کوتاه‌تر و برای اعضای منفی کد بلندتر مورد استفاده قرار می‌گیرد.

تعریف بالا تعریف آنتروپی برای توابع هدف منطقی است، در حالت کلی‌تر اگر ویژگی هدف بتواند C مقدار متفاوت داشته باشد، آنتروپی S برای این دسته بندی C حالتی به صورت زیر تعریف می‌شود:

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2 p_i \quad (3.3)$$

در این رابطه p_i نسبتی از S است که مقدار i را دارد. توجه دارید که پایه‌ی لگاریتم همچنان ۲ باقی می‌ماند زیرا که آنتروپی متوسط تعداد بیت‌های لازم برای ارسال اطلاعات را حساب می‌کند. همچنین توجه داشته باشید که اگر ویژگی هدف C حالت ممکن داشته باشد، مقدار آنتروپی حداکثر $\log_2 C$ خواهد بود.

۳.۴.۱.۲ بهره‌ی اطلاعات، معیار کاهش انتظاری آنتروپی

با داشتن آنتروپی به عنوان معیاری برای میزان یکدستی مجموعه‌ای از نمونه‌های آموزشی، حال می‌توانیم معیاری برای تأثیر گذاری یک ویژگی در دسته‌بندی نمونه‌های آموزشی ارائه دهیم. همان‌طور که گفته شد این معیار بهره‌ی اطلاعات نامیده می‌شود. بهره‌ی اطلاعات میزان کاهش انتظاری آنتروپی از دسته‌بندی بر اساس ویژگی خاص است. به عبارت دقیق‌تر، بهره‌ی اطلاعات ویژگی A بر روی مجموعه‌ی S ، $Gain(S, A)$ را بر حسب مجموعه‌ی نمونه‌های موجود به شکل زیر تعریف می‌کنیم:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (3.4)$$

در این رابطه مقدار $Values(A)$ دسته تمام مقادیر ممکن برای ویژگی A است و S_v زیر تعداد نمونه‌هایی از S هستند که برای ویژگی A مقدار v را دارند ($S_v = \{s \in S | A(s) = v\}$). توجه داشته باشید که جمله‌ی اول در رابطه‌ی ۳.۴ فقط خود آنتروپی مجموعه‌ی S است و جمله‌ی دوم میانگین آنتروپی بعد از تقسیم S با ویژگی A است. میانگین یا امید آنتروپی‌ای که در این جمله آمده است همان مجموع آنتروپی برای تمامی S_v هاست که در نسبت نمونه‌ها $\frac{|S_v|}{|S|}$ ضرب شده است. پس بنابراین $Gain(S, A)$ امید کاهش آنتروپی با تقسیم بندی بر اساس ویژگی A است. به عبارت دیگر، $Gain(S, A)$ میزان اطلاعاتی است که در مورد مقدار تابع هدف با داشتن مقدار ویژگی A بدست می‌آوریم. مقدار $Gain(S, A)$ تعداد بیت‌هایی است که در کد کردن مقدار تابع هدف بر روی اعضای دلخواه S با داشتن مقدار ویژگی A آن‌ها صرفه جویی می‌شود.

برای مثال، فرض کنید که S مجموعه‌ی نمونه‌های آموزشی روزها باشد که توسط ویژگی $Wind$ با مقادیر $Strong$ و $Weak$ توصیف می‌شود. همان‌طور که قبلاً هم داشتیم این مجموعه‌ی S ، ۱۴ نمونه دارد، $[9+, 5-]$. از این ۱۴ نمونه برای مقدار “ $Wind = Weak$ ” ۶ نمونه‌ی مثبت و ۲ نمونه‌ی منفی داریم و بقیه‌ی نمونه‌ها برای مقدار “ $Wind = Strong$ ” است. بهره‌ی اطلاعات با توجه به این ۱۴ نمونه را می‌توان به شکل زیر محاسبه کرد:

$$Values(Wind) = Strong, Weak$$

$$S = [9+, 5-]$$

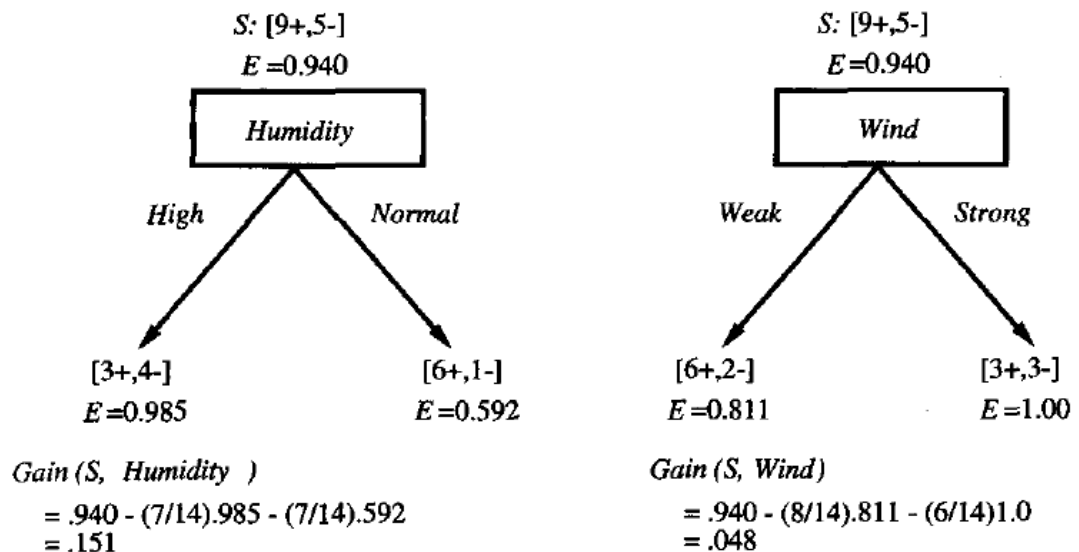
$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$\begin{aligned}
&= Entropy(S) - \left(\frac{8}{14}\right) Entropy(S_{Weak}) \\
&\quad - \left(\frac{6}{14}\right) Entropy(S_{Strong}) \\
&= 0.940 - \left(\frac{8}{14}\right) 0.811 - \left(\frac{6}{14}\right) 1.00 \\
&= 0.048
\end{aligned}$$

بهره‌ی اطلاعات دقیقاً معیاری است که در ID3 برای انتخاب بهترین ویژگی در هر مرحله از رشد درخت استفاده می‌شود. نمونه‌ای از استفاده از بهره‌ی اطلاعات برای تخمین میزان ارتباط ویژگی‌ها با تابع هدف در شکل ۳.۳ آمده است. در این شکل بهره‌ی اطلاعات برای دو ویژگی مختلف Humidity و Wind محاسبه شده تا معلوم گردد که کدام ویژگی، ویژگی بهتری برای دسته بندی نمونه‌های آموزشی آمده در جدول ۳.۲ است.



شکل ۳.۳ ویژگی Humidity بهره‌ی اطلاعاتی بیشتری برای دسته بندی نسبت به Wind دارد. در این شکل E نماد آنتروپی است و S مجموعه‌ی اولیه‌ی نمونه‌های آموزشی است. با داشتن مجموعه‌ی اولیه‌ی S [9+,5-]، با استفاده از ویژگی Humidity دو زیر مجموعه‌ی [3+,4-] (برای Humidity=High) و [6+,1-] (برای Humidity=Normal) بدست می‌آید. بهره‌ی اطلاعات این تقسیم بندی 0.151 است که از مقدار نظیر در تقسیم بندی بر اساس (0.048) باد بیشتر است.

۳.۴.۲ یک مثال

برای تصور بهتر از عملکرد ID3، کار یادگیری که توسط نمونه‌های آموزشی جدول ۳.۲ بیان شده است را در نظر بگیرید. در اینجا ویژگی هدف ویژگی PlayTennis است، که مقادیر Yes و No دارد. مرحله‌ی اول الگوریتم را در نظر بگیرید، در این مرحله بالاترین قسمت درخت تشکیل می‌شود. چه ویژگی‌ای باید در این قسمت بررسی شود؟ ID3 بهره‌ی اطلاعات را برای تمامی ویژگی‌ها (Outlook،

Temperature, Humidity و Wind) تعیین می‌کند، سپس ویژگی‌ای را که بالاترین بهره‌ی اطلاعات را دارد بر می‌گزیند. محاسبه‌ی لازم برای دو مورد از این ویژگی‌ها در شکل ۳.۳ آمده است. مقادیر بهره‌ی اطلاعات محاسبه شده برای تمامی ویژگی‌ها به شرح زیر است:

$$\text{Gain}(S, \text{Sky}) = 0.246$$

$$\text{Gain}(S, \text{Humidity}) = 0.151$$

$$\text{Gain}(S, \text{Wind}) = 0.048$$

$$\text{Gain}(S, \text{AirTemp}) = 0.029$$

در این روابط S همان مجموعه‌ی نمونه‌های جدول ۳.۲ است.

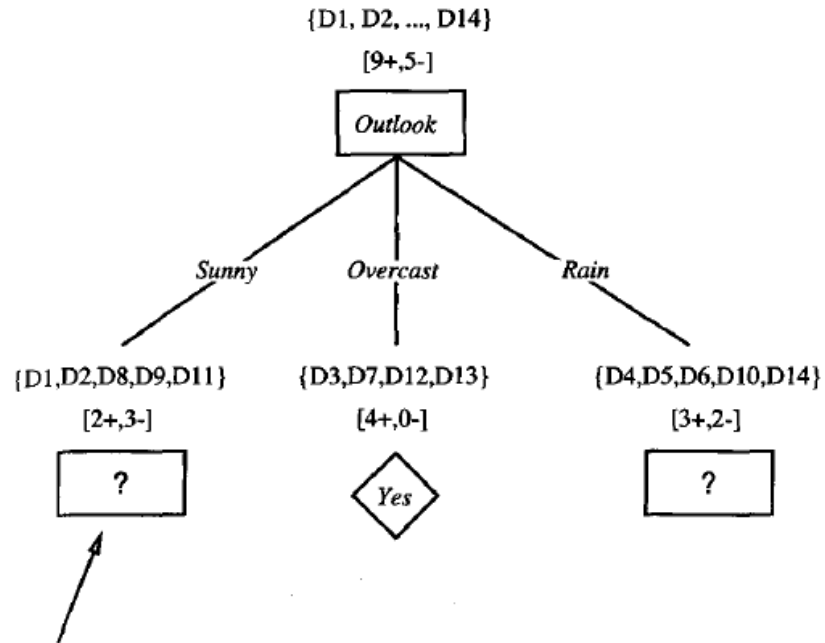
روز	Outlook	Temperature	Humidity	Wind	PlayTennis
روز ۱	Sunny	Warm	High	Weak	No
روز ۲	Sunny	Warm	High	High	No
روز ۳	Overcast	Warm	High	Weak	Yes
روز ۴	Rain	Mild	High	Weak	Yes
روز ۵	Rain	Cool	Normal	Weak	Yes
روز ۶	Rain	Cool	Normal	Strong	No
روز ۷	Overcast	Cool	Normal	Strong	Yes
روز ۸	Sunny	Mild	High	Weak	No
روز ۹	Sunny	Cool	Normal	Weak	Yes
روز ۱۰	Rain	Mild	Normal	Weak	Yes
روز ۱۱	Sunny	Mild	Normal	Strong	Yes
روز ۱۲	Overcast	Mild	High	Strong	Yes
روز ۱۳	Overcast	Hot	Normal	Weak	Yes
روز ۱۴	Rain	Mild	High	Strong	No

جدول ۳.۲ نمونه‌های آموزشی برای مفهوم هدف *PlayTennis*.

بنا بر بهره‌های اطلاعات محاسبه شده، ویژگی Outlook بیشترین تأثیر را بر PlayTennis بر روی نمونه‌های آموزشی دارد. بنابراین Outlook بهترین ویژگی برای بررسی در گره ریشه‌ی درخت است، و شاخه‌ها نیز مقادیر مختلف این ویژگی (Sunny, Cloudy و Rainy) خواهند بود. درخت حاصل در شکل ۳.۴ به همراه نمونه‌های مربوط به هر شاخه نشان داده شده است. توجه دارید که برای تمامی نمونه‌هایی که Outlook=Cloudy است، مقدار PlayTennis نیز بله است. بنابراین این گره از درخت با PlayTennis=Yes علامت گذاری می‌شود. در مقابل، برای دو وضع هوای Rainy و Sunny آنتروپی صفر نیست و درخت تصمیم‌گیری در زیر این شاخه‌ها رشد بیشتری خواهد کرد.

فرایند انتخاب یک ویژگی جدید و تقسیم نمونه‌ها دوباره برای گره‌های غیر پایانی^۱ انجام می‌شود با این تفاوت که در این مرحله فقط نمونه‌هایی که با گره تطابق دارند مورد استفاده قرار می‌گیرند و ویژگی‌هایی که قبلاً استفاده شده‌اند از مجموعه‌ی مربوطه حذف می‌گردند تا هر ویژگی در هر مسیر از ریشه تا برگ حداکثر یک بار ظاهر شود. این فرایند برای تمامی برگ‌های بدست آمده ادامه پیدا می‌کند تا یکی از دو شرط روبرو درست شود: (۱) همه‌ی ویژگی‌ها استفاده شوند، (۲) نمونه‌های تمامی برگ‌ها از مقدار یکسانی از تابع هدف را داشته باشند (آنتروپی‌شان صفر شود). شکل ۳.۴ محاسبه‌ی بهره‌ی اطلاعات برای مراحل بعدی رشد درخت را نشان می‌دهد. درخت کامل شده توسط ID3 برای تمامی ۱۴ نمونه‌ی جدول ۳.۲ در شکل ۳.۱ آمده است.

¹ nonterminal



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

شکل ۳.۴ درخت نیمه کاره ای که بعد از یک مرحله اجرای ID3 بدست می آید. نمونه های آموزشی هر گره دسته بندی و جدا شده اند. مقدار Cloudy چون تنها نمونه های مثبت دارد پس با Yes علامت گذاری شده است. دو برگ دیگر با انتخاب ویژگی هایی که بهره ای اطلاعات بیشتر (برای نمونه های همان شاخه) دارند باز هم توسعه خواهند یافت.

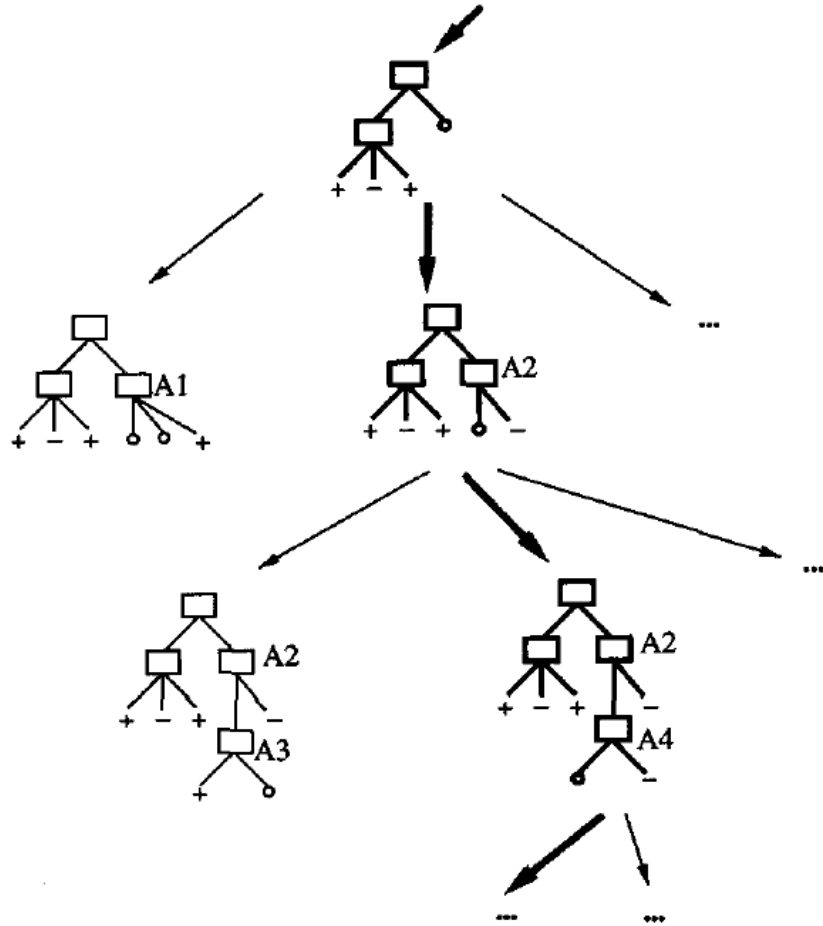
۳.۵ جستجو در فضای فرضیه ها در یادگیری درختی

مشابه دیگر متد های یادگیری استقرایی، ID3 را نیز می توان جستجویی در میان فضایی از فرضیه ها برای پیدا کردن متناسب ترین فرضیه با نمونه های آموزشی در نظر گرفت. فضای فرضیه ای که توسط ID3 جستجو می گردد مجموعه ای تمامی درخت های تصمیم گیری است. ID3 جستجوی ساده به پیچیده^۱ و hill-climbing را در این فضای فرضیه ای انجام می دهد. ابتدای این جستجو درخت بسیار ساده ای تهی است، سپس با ادامه ی فرایند کم کم درخت جزئی تر می گردد تا به درختی برسد که بتواند تمامی نمونه های آموزشی را درست دسته بندی کند. تابع بهره ای اطلاعات این جستجوی hill-climbing را کنترل می کند. این جستجو در شکل ۳.۵ نشان داده شده است.

با نگاه به جنبه ی جستجویی و با توجه به فضای جستجو و استراتژی جستجوی ID3، تعدادی از قابلیت ها و محدودیت های آن مشاهده می شود:

¹ simple-to-complex

- فضای فرضیه ای ID3 که همان تمامی درخت های تصمیم گیری است تمامی توابع گسسته مقدار متناهی را با توجه به ویژگی های موجود در بر می گیرد. زیرا که هر تابع متناهی گسسته مقدار را می توان با درخت تصمیم گیری ای نشان داد، پس ID3 مشکلی (که ممکن است تابع هدف در فضای فرضیه ای نباشد) که خیلی از متدها (مثل متدهایی که فقط ترکیب های عطفی را در نظر می گیرند) در مورد محدودیت فضای فرضیه ای دارند را ندارد.



شکل ۳.۵ جستجوی ID3 در فضای فرضیه ای.

- ID3 جستجوی ساده به پیچیده را در میان تمامی درخت های تصمیم گیری مختلف انجام می دهد. این جستجو توسط تابع بهره ای اطلاعات کنترل می شود.
- ID3 تنها یک فرضیه را در جستجو به دنبال فرضیه ی مطلوب در فضای درخت های تصمیم گیری دنبال می کند. این نوع جستجو با الگوریتم Candidate-Elimination که تمامی فرضیه های ممکن سازگار با نمونه های آموزشی را پیدا می کرد در تضاد است. با محدود شدن به یک فرضیه ی خاص، ID3 قابلیت معلوم کردن تمامی فرضیه های متناسب با نمونه های آموزشی را از دست می دهد. برای مثال، ID3 نمی تواند مشخص کند که چند درخت تصمیم گیری با نمونه های آموزشی مطابقت، یا اینکه نمی تواند بین این چندین درخت تصمیم گیری مختلف در زمان عدم هماهنگی رای گیری کند.

- نسخه‌ی اصلی ID3 هیچ بازنگری‌ای^۱ به عملیات‌های قبلی در جستجویش نمی‌کند. زمانی که یک ویژگی را برای مرحله‌ی خاص در درخت انتخاب کرد هیچ گاه برای تغییر آن به این مرحله باز نمی‌گردد. بنابراین این احتمال وجود دارد که به ریسک‌های روش‌های hill climbing بدون بازنگری دچار شویم: احتمال دارد که به جای یک مینیمم مطلق به یک مینیمم موضعی میل کنیم. در مورد ID3، این مشکل با انتخاب درختی که به صورت موضعی و در مسیر مورد بررسی بهینه‌ترین است ایجاد می‌شود. با این وجود، این جواب موضعی بهینه شاید نسبت به درخت‌هایی که در دیگر شاخه‌های دیگر جستجو وجود دارند صلاحیت کمتری داشته باشد. در ادامه به تغییری در الگوریتم اصلی خواهیم پرداخت و نوعی بازنگری را به الگوریتم اضافه می‌کند. (هرس درخت)
- ID3 در مرحله‌ی جستجو از تمامی نمونه‌های آموزشی برای انتخاب آماری‌اش در چگونگی تغییر درخت فعلی استفاده می‌کند. این کار با متد‌های دیگر که تک‌تک به سراغ نمونه‌های آموزشی می‌روند در تضاد است (الگوریتم‌هایی چون Find-S یا Candidate-Elimination). یکی از مزایای استفاده از خواص آماری تمامی نمونه‌های آموزشی (خواصی چون بهره‌ی اطلاعات) این است که جستجو نسبت به خطاها حساسیت کمتری خواهد داشت. ID3 به راحتی می‌تواند با داده‌های خطا دار آموزشی نیز کار کند، فقط کافی است که شرط خروج را از دسته بندی درست تمامی نمونه‌های آموزشی به دسته بندی درست اکثریت نمونه‌های آموزشی تغییر دهیم.

۳.۶ بایاس استقرایی در یادگیری درختی

خط مشی که ID3 برای تعمیم بر روی داده‌های آموزشی استفاده می‌کند چیست؟ به عبارت دیگر، بایاس استقرایی ID3 چیست؟ با توجه به آنچه که در فصل ۲ گفته شد، بایاس استقرایی دسته فرض‌هایی است که علاوه بر داده‌های آموزشی فرض می‌شود تا بتوان تعمیم دسته بندی اعمالی یادگیر را توجیه کرد.

با معلوم بودن مجموعه‌ی نمونه‌های آموزشی، تعداد بسیار زیادی درخت تصمیم‌گیری سازگار با این نمونه‌ها را می‌توان مشخص کرد. توصیف بایاس استقرایی ID3 جواب این سؤال است که چرا ID3 با وجود درخت‌های سازگار بسیار با نمونه‌های آموزشی، درختی به خصوص را انتخاب می‌کند؟ این درخت چه ویژگی‌هایی دارد؟ الگوریتم ID3 اولین درخت قابل قبول را که در جستجوی ساده به پیچیده و hill-climbing آن با آن مواجه می‌شود را از میان تمامی درخت‌های ممکن بر می‌گزیند. استراتژی جستجوی ID3 را می‌توان به صورت روبرو توصیف کرد: (a) درخت‌های کوتاه‌تر نسبت به درخت‌های بلندتر ارجحیت دارند، (b) درختی برگزیده می‌شود که بهره‌ی اطلاعاتش در نزدیکی ریشه بیشتر باشد. به دلیل تأثیرات پیچیده‌ی انتخاب ویژگی‌ها در ID3 مشخص کردن بایاس استقرایی به طور دقیق کمی دشوار است. با این وجود، می‌توان به صورت کلی گفت که این الگوریتم درخت‌های کوتاه‌تر را بر درخت‌های بلندتر ترجیح می‌دهد.

بایاس استقرایی تخمینی ID3: درخت‌های کوتاه‌تر نسبت به درخت‌های بلندتر ارجحیت دارند.

در واقع، می‌توان گفت که الگوریتمی مشابه ID3 وجود دارد که دقیقاً بایاس استقرایی فوق را دارد. الگوریتمی را در نظر بگیرید که جستجوی میان فرضیه‌ها را با درختی تهی آغاز می‌کند و با جستجوی کم عمقی^۲ (BFS) کم‌کم به طرف درخت‌های پیچیده‌تر می‌رود. یعنی ابتدا تمامی درخت‌هایی که عمق یک دارند را جستجو می‌کند سپس به سراغ عمق دو می‌رود و ... این الگوریتم در مواجهه با درختی که با تمامی

¹ backtrack

² breath first search

نمونه های آموزشی سازگار است، کوچک ترین درخت ممکن را در عمق فعلی خروجی می دهد (مثلاً درختی که کمترین تعداد گره را داشته باشد). این الگوریتم به اختصار BFS-ID3 خوانده می شود. BFS-ID3 کوتاه ترین درخت تصمیم گیری ای سازگار با داده های آموزشی را پیدا می کند، یعنی دقیقاً بایاس استقرایی "درخت های کوتاه تر نسبت به درخت های بلند تر ارجحیت دارند" را داراست. ID3 را می توان تخمینی از BFS-ID3 دانست، با این تفاوت که جستجو در ID3، جستجویی حریصانه برای یافتن کوتاه ترین درخت ممکن است و تمامی فضای فرضیه ها جستجو نمی شود.

چون ID3 از بهره ای اطلاعات و استراتژی hill-climbing استفاده می کند بایاس پیچیده تری نسبت به BFS-ID3 دارد. در کل، همیشه کوتاه ترین درخت ممکن پیدا نمی شود و الگوریتم تمایل دارد که درخت هایی را انتخاب کند که بهره ای اطلاعاتشان در نزدیکی ریشه بیشتر باشد.

تخمینی مناسب تر از بایاس استقرایی ID3: درخت های کوتاه تری نسبت به درخت های بلند تر ارجحیت دارند. درخت هایی که بهره ای اطلاعات بیشتری در نزدیکی ریشه دارند نیز ارجحیت دارند.

۳.۶.۱ بایاس های محدود کننده و بایاس های مطلوب

تفاوت های جالبی میان دو نوع مختلف بایاس که توسط دو الگوریتم ID3 و Candidate-Elimination به کار برده می شود وجود دارد. به تفاوت های این دو روش جستجوی فضای فرضیه ها توجه کنید:

- ID3 فضای فرضیه ای کاملی را جستجو می کند (فضایی که تمامی توابع گسسته مقدار متناهی را می تواند توصیف کند). از طرفی این جستجو تمامی فضای فرضیه ای را شامل نمی شود، جستجو از فرضیه های ساده تر شروع شده و به محض رسیدن به شرط خروج پایان می یابد (مثلاً زمانی که فرضیه ای تمامی نمونه های آموزشی را درست دسته بندی می کند). بایاس استقرایی این الگوریتم فقط ناشی از نحوه ی ترتیب جستجوی این فرضیه هاست و بایاس دیگری وجود ندارد.
- الگوریتم Candidate-Elimination فضای فرضیه ای غیر کاملی را جستجو می کند (تنها زیر مجموعه ای از تمامی مفهوم هایی را که می توان از نمونه ها یاد گرفت). اما این فضای فرضیه ای را کامل جستجو می کند و تمامی فرضیه هایی که با نمونه های آموزشی سازگار هستند را پیدا می کند. بایاس استقرایی این الگوریتم فقط ناشی از میزان شمول فضای فرضیه ای آن است و استراتژی جستجویش هیچ نقشی در بایاس ندارد.

به طور خلاصه، بایاس استقرایی ID3 از استراتژی جستجویش ناشی می شود در حالی که بایاس استقرایی Candidate-Elimination ناشی از فضایی جستجو آن است.

پس، بایاس استقرایی ID3 یک ترجیح فرضیه ها بر دیگر فرضیه هاست، بدون اینکه فضای فرضیه ای هیچ محدودیتی ایجاد کند. این نوع بایاس را معمولاً بایاس ترجیحی^۱ (یا بایاس جستجویی^۲) می نامند. در مقابل، بایاس Candidate-Elimination کاملاً به فضای فرضیه ای در نظر گرفته شده وابسته است. این نوع بایاس را نیز بایاس محدودیتی^۳ (یا بایاس زبانی^۴) می نامند.

¹ preference bias

² search bias

³ restriction bias

⁴ language bias

با دانستن اینکه بایاس‌های استقرایی برای تعمیم نمونه‌ها هستند، (با توجه به آنچه که در فصل ۲ گفته شد)، کدام نوع از بایاس پسندیده تر است؟ بایاس ترجیحی یا بایاس محدودیتی؟

معمولاً، بایاس‌های ترجیحی به بایاس‌های محدودیتی ترجیح داده می‌شوند، زیرا که به یادگیر اجازه می‌دهند تا در فضای فرضیه ای کاملی که مطمئناً تابع هدف مجهول را در بر می‌گیرد کار کند. در مقابل، بایاس‌های محدودیتی که توابع قابل یادگیری را به دسته‌ی خاصی محدود می‌کنند از ارجحیت کمتری برخوردارند، زیرا که پیش‌فرضی را در مورد تابع هدف مجهول می‌گذارند.

با وجود اینکه دو الگوریتم بحث شده ID3 بایاسی کاملاً ترجیحی و Candidate-Elimination بایاسی کاملاً محدودیتی دارد، اما الگوریتم‌هایی وجود دارند که بایاسشان ترکیبی از این بایاس‌هاست. برای مثال، برنامه ای که در فصل ۱ برای تخمین عددی در یادگیری بازی‌ها توضیح داده شد را در نظر بگیرید. در این برنامه، توابع تخمین یاد گرفته شده ترکیب‌های خطی دسته ای از ویژگی‌های صفحه بودند، و الگوریتم یادگیری پارامترهای این ترکیب خطی را تعیین می‌کرد با سازگاری بیشتری با نمونه‌های آموزشی داشته باشد. در این مثال، اینکه از توابع خطی برای نمایش تابع تخمین استفاده کنیم نوعی بایاس محدودیتی است (توابع غیر خطی را نمی‌توان با این نمایش نشان داد). از طرف دیگر، اینکه با روش خاصی (مثل الگوریتم LMS) پارامترها را تعیین کنیم بایاسی ترجیحی است، بایاسی که باعث می‌شود تمامی پارامترهای ممکن را بررسی نکنیم.

۳.۶.۲ چرا فرضیه‌های کوتاه تر ارجحیت دارند؟

آیا ترجیح ID3 برای درخت‌های کوتاه‌تر برای تعمیم نمونه‌های آموزشی مفید است؟ فیلسوفان سال‌ها درباره‌ی چنین سؤالی بحث کرده و می‌کنند. ویلیام او اوکام (William of Occam) یکی از اولین افرادی بود که درباره‌ی بحث‌هایی را مطرح کرد (سال 1320)، به همین دلیل، این نوع بایاس را بایاس تیغ Occam می‌نامند.

بایاس تیغ Occam: در میان فرضیه‌های سازگار، فرضیه‌هایی که ساده‌ترند ارجحیت دارند.

البته با نام‌گذاری یک بایاس نمی‌توان آن‌را توجیه کرد. حال چرا باید فرضیه‌های ساده تر ارجح باشند؟ توجه دارید که دانشمندان بعضی مواقع چنین بایاسی را مورد استفاده قرار می‌دهند. برای مثال، فیزیکدانان نظریه‌های ساده تر درباره‌ی حرکت سیارات را ترجیح می‌دهند. چرا؟ یکی از استدلال‌های ممکن این است که تعداد فرضیه‌های ساده تر نسبت به فرضیه‌های پیچیده تر بسیار کمتر است، پس به نظر می‌رسد احتمال اینکه فرضیه ای پیدا شود که به طور اتفاقی با نمونه‌های آموزشی سازگار باشد کم است. در مقابل، تعداد بسیار زیادی از فرضیه‌های پیچیده موجود است که با نمونه‌های آموزشی سازگارند اما در تعمیم نمونه‌های آموزشی عاجزند. برای مثال، فرضیه‌های درخت‌های تصمیم‌گیری را در نظر بگیرید. تعداد درخت‌هایی که ۵۰۰ گره دارند بسیار بیشتر از درخت‌هایی است که ۵ گره دارند. اگر ۲۰ نمونه‌ی آموزشی داشته باشیم، تعداد بسیار زیادی از درخت‌های ۵۰۰ گره ای با آن‌ها سازگارند، در حالی که جای تعجب ندارد که فقط یک درخت ۵ گره ای متناسب با آن ۲۰ نمونه‌ی آموزشی باشد. بنابراین احتمال اینکه سازگاری درختی با ۵ گره اتفاقی بوده باشد بسیار کمتر از احتمال اتفاقی بودن سازگاری درختی با ۵۰۰ گره است.

با بررسی‌های بیشتر، معلوم می‌گردد که اشکال کلی‌ای در استدلال بالا وجود دارد. با همین استدلال می‌توان گفت که باید درخت‌هایی که ۱۷ گره برگ و ۱۱ گره غیر برگ دارند که تمامی ویژگی‌های یازده‌گانه‌ی نمونه‌ها را به ترتیب بررسی می‌کنند احتمال اتفاقی بودن بسیار کمتری دارند، زیرا تعداد چنین درخت‌هایی بسیار کم است پس شانس اتفاقی بودن (بنا به استدلال بالا) بسیار کمتر خواهد بود. این اشکال اینجاست که

زیرمجموعه های کوچک بسیاری از فضای فرضیه ها وجود دارد که چنین تعداد کمی را دارند و پیدا کردن همه ی آن ها ساده نیست. پس چرا باید باور داشته باشیم زیرمجموعه ی درخت هایی که طول کوتاه تری دارند باید نسبت به آن دیگر زیر مجموعه های کوچک برتری داشته باشد؟

اشکال دومی که درباره ی این استدلال برای تیغ Occam^۱ پیش می آید این است که اندازه ی یک فرضیه با روش خاصی مشخص می شود که در یادگیر تعبیه شده. اگر دو یادگیر با روش های مختلف اندازه گیری اندازه ی فرضیه بر روی یک مسئله به کار گرفته شوند در آخر فرضیه های خروجی متفاوتی خواهند داشت، در حالی که هر دو عملیات خود را توسط تیغ Occam توجیه شده می دانند. برای مثال، تابعی که در شکل ۳.۱ نشان داده شده است را می توان با درختی با یک گره نیز نشان داد، درختی که یادگیر برای دسته بندی نمونه ها از ویژگی XYZ استفاده می کند، ویژگی منطقی XYZ زمانی درست است که نمونه، نمونه ی مثبتی باشد و در غیر این صورت غلط است. بنابراین دو یادگیر که هر دو از تیغ Occam استفاده می کنند اگر یکی ویژگی XYZ و دیگری ویژگی های Outlook, Temperature, Humidity و Wind را استفاده کنند درخت های متفاوتی خواهند داشت.

این بحث آخر نشان می دهد که تیغ Occam در دو یادگیر که از یک مجموعه نمونه های آموزشی یکسان استفاده می کنند و فقط نمایش داخلی نمونه هایشان متفاوت است دو فرضیه ی کاملاً متفاوت بدهد. با دانستن این حقیقت ممکن است به طور کلی تیغ Occam را رد کنیم. با این وجود، سؤال اینکه کدام نمایش درونی ممکن است با تکامل^۲ یا انتخاب طبیعی^۳ ایجاد شود را در نظر بگیرید. جمعیتی از یادگیر های مصنوعی ای را که از طریق فرایند های تکاملی زاد و ولد، جهش و انتخاب به وجود آمده اند را در نظر بگیرید. و ببایید فرض کنیم که این فرایند تکاملی می تواند سیستم های ادراکی این یادگیرها را از نسلی به نسلی تغییر دهد، مشابه تغییر ویژگی های داخلی ای که عوامل جهان اطراف را با آن ها درک می کنند. و برای بحث، فرض کنیم که این عوامل یادگیری از الگوریتم یادگیری یکسانی (مثلاً ID3) استفاده می کنند که با تکامل تغییر نخواهد یافت. منطقی است که فرض کنیم در طول زمان، تکامل نمایش های داخلی را ایجاد کند که موفقیت فرد در ارتباط با محیط را افزایش دهد. فرض کنیم که موفقیت یک عامل به شدت وابسته به قدرت تعمیمش دارد، بنابراین می توان انتظار داشت که نمایش های داخلی ای که خوب با الگوریتم یادگیری و بایاس استقرایی اش کار می کنند ایجاد شوند. اگر گونه هایی از یادگیرها از الگوریتم یادگیری از که از بایاس استقرایی تیغ Occam استفاده می کنند، وجود داشته باشد، انتظار خواهیم داشت که تکامل نمایش های داخلی ای را ایجاد کند که تیغ Occam برایشان استراتژی موفقتری است. نکته اصلی بحث در اینجا این است که تکامل نمایش های داخلی ای را ایجاد خواهد کرد که بایاس الگوریتم یادگیری نوعی خود توجیهی^۴ داشته باشد، زیرا که می تواند بسیار راحت تر از تغییر الگوریتم، نمایش را تغییر دهد.

فعلاً بحث مربوطه ی تیغ Occam را رها می کنیم. اما در فصل ۶، در قسمتی که قانون کمترین طول توضیح را بررسی خواهیم کرد، نسخه ای از تیغ Occam را که با چارچوب بیزی توجیه می شود را بررسی خواهیم کرد.

۳.۷ مشکلات یادگیری درختی

مشکلات کاربردی یادگیری درختی شامل، مشخص کردن حداکثر عمق درخت، چگونگی بررسی ویژگی های پیوسته، انتخاب معیار انتخاب ویژگی ها، یادگیری با داده هایی با بعضی ویژگی های مجهول، یادگیری با ویژگی های غیر هم هزینه و بهینه کردن محاسبات می شود. در ادامه

¹ Occam's razor

² evolution

³ Natural selection

⁴ self-fulfilling

هر یک از این موارد را بررسی خواهیم کرد و تغییراتی ID3 برای حل این مشکلات را نیز معرفی خواهیم کرد. C4.5 برای حل بعضی از این مشکلات ارائه شده است (Quinlar 1993).

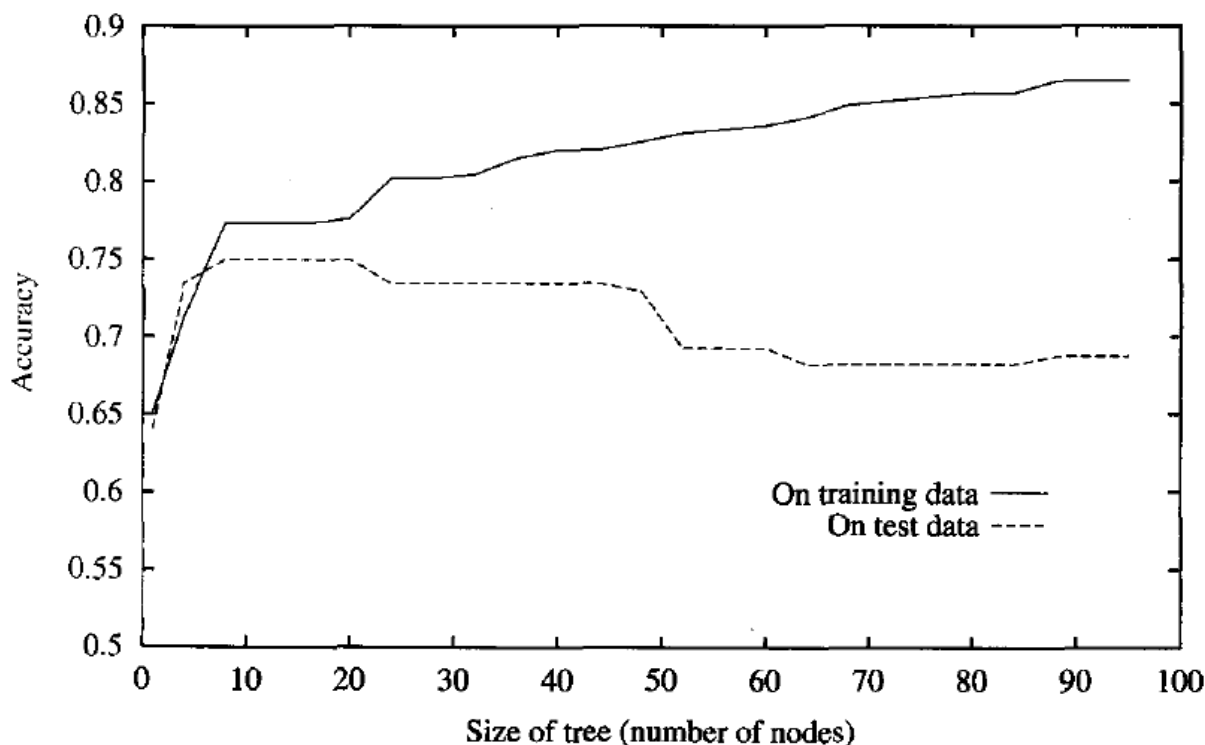
۳.۷.۱ حل مشکل overfit

الگوریتمی که در جدول ۳.۱ آمده است درخت را آنقدر رشد می‌دهد تا تمامی نمونه‌های آموزشی را درست دسته‌بندی کند. با وجود اینکه این استراتژی، استراتژی معقولی است اما همین استراتژی ممکن است مواقعی که داده‌ها خطا دارند یا تعدادشان به اندازه‌ی کافی نیست که تابع هدف را کامل تعریف کنند مشکل ساز باشد. به هر حال در چنین مواقعی، این الگوریتم درخت‌هایی را خروجی می‌دهد که مشکل overfit در نمونه‌های آموزشی دارند.

زمانی می‌گوییم که یک فرضیه مشکل overfit دارد که فرضیه‌ای دیگر موجود باشد که بر روی نمونه‌های آموزشی سازگاری کمتری داشته باشد اما در کل سازگاری بیشتری با کل نمونه‌ها (اعم از آموزشی و غیر آموزشی (جدید)) داشته باشد.

تعریف: اگر فضای فرضیه‌ها H باشد زمانی می‌گوییم که فرضیه مثل $h \in H$ مشکل overfit بر روی نمونه‌های آموزشی دارد که فرضیه‌ای مثل $h' \in H$ وجود داشته باشد به صورتی که خطای h بر روی نمونه‌های آموزشی نسبت به h' کمتر باشد اما خطای h' بر روی کل نمونه‌ها از خطای h کمتر باشد.

شکل ۳.۶ اثر پدیده‌ی overfit را در یک کاربرد Normal یادگیری درختی نشان می‌دهد. در این مثال، الگوریتم ID3 برای تشخیص بیماران دیابتی به کار رفته. محور افقی تعداد کل گره‌های درخت تصمیم‌گیری را در طول رشد درخت نشان می‌دهد. محور عمودی دقت تشخیص‌های درخت را نشان می‌دهد. منحنی توپر میزان دقت درخت را در نمونه‌های آموزشی و منحنی خط چین میزان دقت را بر روی دسته‌ی دیگری از نمونه‌ها نشان می‌دهد (دسته‌ای به جز نمونه‌های آموزشی). همان‌طور که پیش‌بینی می‌شد دقت درخت بر روی نمونه‌های آموزشی با افزایش اندازه‌ی درخت افزایش می‌یابد. با این وجود، دقت دسته‌ی دیگر کاهش می‌یابد. همان‌طور که دیده می‌شود، زمانی که اندازه‌ی درخت از حدود ۲۵ می‌گذرد، پیچیدگی بیشتر درخت باعث کاهش دقت در دسته‌ی دیگر می‌شود در حالی که دقت همچنان در نمونه‌های آموزشی بالا می‌رود.



شکل ۳۶ overfit در یادگیری درختی.

همین طور که ID3 گره های بیشتری برای رشد درخت به آن اضافه می کند، به طور مشابه دقت بر روی نمونه های آموزشی افزایش پیدا می کند. با این وجود، زمانی که دقت بر روی دسته ای از نمونه های غیر آموزشی بررسی می شود، ابتدا افزایش و سپس کاهش مشاهده می شود. برنامه و داده های استفاده شده برای این آزمایش در <http://www.cs.cmu.edu/~tom/mllbook.html> موجود می باشد.

چگونه ممکن است درخت h که نسبت h' عملکرد بهتری بر روی نمونه های آموزشی دارد، در کل نمونه ها عملکرد ضعیف تری داشته باشد؟ یکی از مواردی که چنین مشکلی ایجاد می شود مواقعی است که نمونه های آموزشی خطای تصادفی^۱ یا همان نویز داشته باشند. برای تصور، نمونه ی آموزشی مثبت زیر را در نظر بگیرید که اشتباهاً نمونه ی منفی در نظر گرفته شده:

<Outlook = Sunny, Temperature=Hot, Humidity=Normal, Wind=Strong, PlayTennis=No>

با دادن داده های آموزشی بدون خطا به ID3 درخت نشان داده شده در شکل ۳.۱ بدست می آید. با این وجود اگر این نمونه ی اشتباه را به نمونه های آموزشی اضافه کنیم، ID3 درخت پیچیده تری خروجی می دهد. در کل، نمونه ی جدید در برگ دوم شاخه ی سمت چپ شکل ۳.۱ قرار می گیرد، همراه دو نمونه مثبت قبلی روز ۹ و روز ۱۱. حال چون که این نمونه منفی است، ID3 درخت را در زیر این شاخه بیشتر رشد می دهد. البته، تا زمانی که نمونه ی اشتباه این گونه باشد (فقط مقدار تابع هدف اشتباه تعیین شده باشد)، ID3 ویژگی ای خواهد یافت تا نمونه غلط را از نمونه ی درست جدا کند. نتیجه این خواهد بود که ID3 درخت تصمیم گیری h ای را خروجی می دهد که پیچیده تر از درخت تصمیم گیری درست h' است (شکل ۳.۱). البته h بر روی نمونه های آموزشی دقت زیادی دارد در حالی که h' ساده تر آن میزان دقت را ندارد. با این وجود، با داشتن گره ای که جدیداً اضافه شده و مستقیماً تأثیر نمونه ی خطا دار بوده، انتظار داریم که h از h' دقت کلی بهتری داشته باشد.

¹ Random error

مثال بالا نشان داد که چگونه داده های خطا دار آموزشی می توانند باعث **overfit** شوند. در واقع، **overfit** حتی زمانی که نمونه های آموزشی خطا ندارند نیز ممکن است اتفاق بیفتد، مخصوصاً زمانی که تعداد نمونه ها با تعداد برگ های درخت برابر می شود. در چنین شرایطی، دور از انتظار نیست که نظم های اتفاقی در درخت پدیدار شوند، در این نظم ها به نظر می رسد که ویژگی های خاصی در دسته بندی نمونه ها تأثیر بسیار زیادی دارند در حالی که آن ویژگی ها هیچ ربطی به تابع هدف ندارند. هر گاه چنین نظم های اتفاقی ای ایجاد می شود، احتمال **overfit** نیز بالا می رود. مشکل **overfit** مشکل قابل توجهی در یادگیری درختی و بسیاری از متد های یادگیری دیگر است. برای مثال، در یک مطالعه ای آزمایشی ID3 که بر روی ۵ کار یادگیری و با داده های خطا دار و غیر قطعی^۱ انجام شد (Mingers 1989b)، در اکثر مسایل **overfit** دقت را بین ۱۰ تا ۲۵ درصد کاهش داد.

روش های بسیاری برای حل مسئله **overfit** در یادگیری درختی موجود است. این روش ها به دو دسته ی کلی تقسیم می شوند:

- روش هایی که جلوی رشد درخت را قبل از رسیدن به نقطه ای که تمامی نمونه ها را درست دسته بندی کند می گیرند،
 - روش هایی که اجازه می دهند تا درخت به اندازه ی دلخواه رشد کند سپس درخت را هرس^۲ می کنند.
- با وجود اینکه به نظر می رسد روش های دسته ی اول مستقیم ترند، اما روش های دسته ی دوم در کاربرد موفقیت بیشتری را از خود نشان داده اند. از آنجا که در روش اول معلوم نیست که چه زمان باید جلوی رشد درخت گرفته شود.

جدا از اینکه درخت با کدام روش درخت به اندازه ی اصلی می رسد، سؤال کلیدی این است که معیار درست اندازه نهایی ی درخت چیست؟ روش های زیر برای جواب به این سؤال پیشنهاد می شوند:

- استفاده از دسته ای از نمونه های اضافی (که با نمونه های آموزشی تداخل ندارند) برای تخمین کارایی گره ها و هرس آن ها.
- استفاده از تمام نمونه های موجود برای آموزش، استفاده از آزمونی آماری برای تخمین اینکه آیا رشد (یا هرس) یک گره از درخت تعمیمی را ایجاد می کند یا تنها باعث **overfit** می شود. برای مثال، (Quinlar 1986) از آزمون کای اسکوار (کی دو)^۳ برای جواب سؤال "آیا رشد یک گره به کارایی کلی درخت کمک می کند یا فقط باعث سازگاری با نمونه ی آموزشی می شود؟" استفاده می کند.

- استفاده از معیاری برای اندازه گیری پیچیدگی. در نظر گرفتن نوعی کد سازی برای درخت و متوقف کردن رشد درخت زمانی که این اندازه ی کد کمینه می شود. این روش مبتنی بر توجیه است که "کمترین طول توضیح"^۴ نامیده می شود و مفصلاً در فصل ۶ بررسی شده. برای اطلاعات بیشتر به (Quinlar and Rivest 1989) و (Mehta 1995) مراجعه کنید.

روش اول متداول ترین روش است و گاهی روش آموزش و مجموعه ی تایید^۵ نیز نامیده می شود. در اینجا به دو نسخه ی اصلی این روش می پردازیم. در این روش، داده های موجود به دو دسته تقسیم می شوند: دسته ی آموزشی، که از آن ها برای آموزش درخت استفاده می شود، و دسته ی تایید^۶، که از آن برای ارزیابی دقت فرضیه ها استفاده می شود، در کل، برای ارزیابی تأثیر هرس از این داده ها استفاده می شود. انگیزه ی

¹ nondeterministic

² post-prune

³ chi-square

⁴ Minimum Description Length principle

⁵ training and validation set

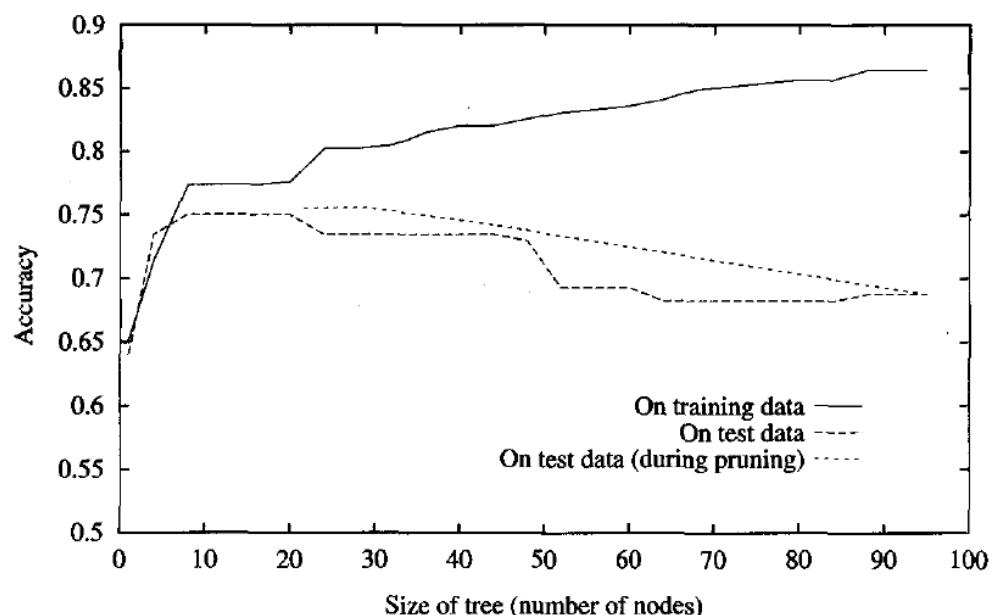
⁶ validation set

اولیه‌ی این روش این است که اگر چه ممکن است یادگیر با داده‌های خطا دار همراه شود، و به نظم‌های تصادفی در میان نمونه‌های آموزشی میل کند، اما احتمال اینکه دسته‌ی تایید نیز همان نظم‌های اتفاقی که نمونه‌های آموزشی دارند را داشته باشد بسیار کم است. بنابراین، می‌توان انتظار داشت که دسته‌ی تایید معیار خوب و مطمئنی در مقابل معیار غلت داده‌های آموزشی به ما بدهد. البته، مهم است که اندازه‌ی دسته‌ی تایید به اندازه‌ی کافی بزرگ باشد تا بتواند به تنهایی نمونه‌ی کاملی از نمونه‌ها را داشته باشد. یکی از ایده‌های معمول این است که یک سوم کل داده‌ها را برای دسته‌ی تایید نگه می‌دارند و از دو سوم دیگر برای آموزش درخت استفاده می‌کنند.

۳.۷.۱.۱ کاهش خطا با هرس کردن

دقیقاً چگونه می‌توان با استفاده از یک دسته‌ی تایید از **overfit** جلوگیری کرد؟ یکی از روش‌های ممکن **reduced-error pruning** نام دارد (Quinlar 1987)، بر اساس این روش تمامی گره‌های درخت مستعد هرس شدن هستند. البته هرس کردن یک گره تصمیم باعث حذف شدن زیر درخت متصل به گره مذکور و تبدیل آن به برگ تبدیل خواهد شد با علامت متداول‌ترین دسته بندی از داده‌های آموزشی می‌شود. گره‌هایی هرس خواهند شد که هرس شدنشان باعث تأثیر منفی در دسته بندی دسته‌ی تایید نشود. این عمل باعث حذف برگ‌هایی که بر اثر نظم‌های تصادفی داده‌های آموزشی ایجاد شده‌اند می‌شوند، زیرا که احتمال تکرار همان نظم‌های تصادفی در دسته‌ی تایید ناچیز است. در چندین مرحله گره‌ها هرس می‌شوند، در هر مرحله گره‌ای که هرسش باعث حداکثر افزایش دقت درخت در دسته بندی دسته‌ی تایید می‌شود هرس می‌شود. هرس کردن آنقدر ادامه خواهد یافت تا زمانی که هرس کردن درخت اثر منفی داشته باشد (دقت دسته بندی دسته‌ی تایید را کم کند).

اثر **reduce-error pruning** بر روی دقت درخت تصمیم گیری در شکل ۳.۷ نشان داده شده است. مشابه شکل ۳.۶ دقت درخت برای نمونه‌های آموزشی و غیر آموزشی در هنگام هرس نشان داده شده است. منحنی اضافه شده دقت را بر روی دسته‌ی تست درخت هرس شده نشان می‌دهد. زمانی که هرس کردن آغاز می‌شود درخت در حداکثر اندازه‌ی خودش است. با ادامه‌ی هرس کردن تعداد گره‌های درخت کاهش و دقت بر روی دسته‌ی تست افزایش می‌یابد. در اینجا داده‌های موجود به سه دسته تقسیم شده‌اند: نمونه‌های آموزشی، نمونه‌های دسته‌ی تایید، و دسته‌ی تست. از دسته‌ی آخر برای بررسی دقت درخت بر روی نمونه‌های جدید (قدرت تعمیم درخت) استفاده می‌شود. نمودار نشان داده شده دقت را بر روی نمونه‌های آموزشی و دسته‌ی تست نشان می‌دهد. دقت دسته‌ی ارزیابی که برای هرس کردن از آن استفاده می‌شود در شکل نشان داده نشده است.



شکل ۳.۷ اثر *reduced-error pruning* در درخت تصمیم گیری.

شکل همان منحنی‌های دقت نمونه‌های آموزشی و دسته‌ی تست را نشان می‌دهد (شکل ۳۶). علاوه بر این، اثر *reduced-error pruning* بر درخت خروجی *ID3* در شکل نشان داده شده است. توجه داشته باشید که دقت دسته‌ی تست با هرس شدن گره‌ها افزایش می‌یابد. در اینجا، دسته‌ی تایید که برای هرس استفاده شده از هر دو دسته‌ی آموزشی و تست مجزا بوده است.

زمانی که تعداد زیادی از داده‌ها در دسترس است، استفاده از دسته‌ای از آن‌ها برای کنترل هرس راه حل موثری است. مانع اصلی این روش این است محدودیت تعداد داده‌هاست. گاهی اوقات کم کردن قسمتی از داده‌ها برای استفاده در دسته‌ی تایید باعث کافی نبودن تعداد داده‌های موجود برای آموزش درخت می‌شود. در قسمت بعدی روش دیگری را برای هرس توضیح خواهیم داد که در کاربرد‌های عملی زمانی که تعداد داده‌ها کم است موفقیت آمیز بوده است. تکنیک‌های دیگری نیز از جمله بخش بندی داده‌ها در دسته‌های متعدد با ترکیب‌های مختلف در دفعات متعدد و میانگین گیری در میان درخت‌ها، ارائه شده است. بررسی‌های تجربی دیگر متدهای هرس در (Mingers 1989b) و (Malerba 1995) آمده است.

۳.۷.۱.۲ قانون پس هرس

در واقع، یکی از متدهای موفق پیدا کردن فرضیه‌ای با دقت بالا، تکنیکی به نام پس هرس^۱ است. نسخه‌ای از این متد هرس کردن در C4.5 (Quinlar 1993) استفاده شده است. قانون پس هرس مراحل زیر را شامل می‌شود:

۱. درخت متناسب با داده‌های آموزشی را پیدا کن، به درخت اجازه بده تا اندازه‌ی دلخواه رشد کند و *overfit* ایجاد شود.
۲. درخت را به دسته قوانین هم ارز تبدیل کن (برای هر مسیر از ریشه به برگ یک قانون).
۳. هر قانون را با حذف کردن شروطی که باعث افزایش دقت تخمینی‌اش می‌شود هرس کن.
۴. قوانین هرس شده را به ترتیب دقتشان مرتب کن، و در دسته بندی نمونه‌های جدید این سری را در نظر بگیر.

¹ post pruning

برای تصور، دوباره درخت تصمیم گیری شکل ۳.۱ را در نظر بگیرید. در قانون پس هرس، برای هر برگ در درخت یک قانون ایجاد می‌شود. تمامی گره‌هایی که بین ریشه و برگ قرار دارند جزو شروط قانون قرار می‌گیرند و دسته بندی برگ نیز، حکم قانون خواهد بود. برای مثال، برای چپ‌ترین مسیر درخت شکل ۳.۱ قانون زیر بدست می‌آید:

IF (Outlook = Sunny) \wedge (Humidity=High) THEN PlayTennis=No

مرحله‌ی بعدی حذف شروطی که حذفشان دقت تخمینی را کم‌تر نمی‌کند است. برای مثال، برای قانون بالا، قانون پس هرس حذف شروط (Outlook=Sunny) و (Humidity=High) را در نظر خواهد گرفت، و هر کدام از حذف‌ها که پیشرفت بهتری در دقت تخمینی قانون ایجاد کند را انجام می‌دهد و هرس شرط بعدی را به مرحله‌ی بعد موکول خواهد کرد. شرط اصلی هرس کردن این است که بعد از هرس دقت تخمینی کاهش نیابد.

همان طور که در بالا نیز گفته شد، یکی از راه‌های اندازه گیری دقت قوانین استفاده از دسته‌ی تایید است. متد دیگری که در C4.5 نیز آمده تخمین دقت قوانین بر اساس خود دسته‌ی آموزشی است، این تخمین با در نظر گرفتن تمایل نمونه‌های آموزشی به سمت قوانین موجود با بدبینی به نمونه‌های آموزشی انجام می‌شود. دقیق‌تر اینکه، C4.5 تخمین بدبین خود را با محاسبه‌ی دقت قوانین بر روی نمونه‌های آموزشی انجام می‌دهد سپس انحراف معیار^۱ این دقت تخمینی را با فرض توزیع دو جمله‌ای محاسبه می‌کند. برای اطمینان، حد پایین تخمین را به عنوان دقت قانون در نظر می‌گیرد (برای مثال برای فاصله‌ی اطمینان 95% ی دقت قانون با نگاه بدبینانه همان دقت بر روی نمونه‌های آموزشی منهای ۱.۹۶ برابر انحراف از معیار خواهد بود). در کل، برای مجموعه‌های این تخمین بدبینانه بسیار نزدیک به دقت مشاهده شده خواهد بود (یعنی مقدار انحراف معیار بسیار کوچک است)، در حالی که با کاهش اندازه دسته داده این مقدار از دقت مشاهده شده کمتر می‌گردد. با وجود اینکه این روش توجیهی آماری ندارد، اما در عمل کاربرد خود را اثبات کرده است. برای بازه‌های اطمینان و تخمین میانگین به فصل ۵ مراجعه کنید.

چرا درخت تصمیم گیری را قبل از هرس به قوانین تبدیل کنیم؟ این کار سه مزیت دارد:

- تبدیل به قوانین باعث می‌شود که تأثیرات مختلف یک گره در درخت مشخص و جدا گردد، زیرا که هر مسیر از ریشه تا برگ یک قانون را تشکیل می‌دهد و هرس گره‌های تصمیم اثرات مختلفی بر مسیرهای مختلف می‌گذارد. بعلاوه، اگر خود درخت را هرس کنیم دو انتخاب بیشتر نداریم، یکی اینکه گره را حذف کنیم و دیگری اینکه گره را دست نخورده باقی بگذاریم.
- تبدیل به قوانین تمایز بین ویژگی‌های که در نزدیک ریشه بررسی می‌شوند و ویژگی‌هایی که نزدیک برگ‌ها بررسی می‌شوند را از بین می‌برد. بنابراین با این کار مشکلات ساختاری مواجه نخواهیم شد، مشکلاتی نظیر چگونگی بازسازی دوباره درخت در صورت هرس شدن ریشه.
- تبدیل به قوانین درخت را برای خواندن راحت‌تر می‌کند. قوانین معمولاً راحت‌تر درک می‌شوند.

۳.۷.۲ کار با ویژگی‌های پیوسته

تعریف اولیه‌ی ما از ID3 منحصر به ویژگی‌های گسسته مقدار بود. هم خود ویژگی هدف و هم ویژگی‌هایی که در گره‌ها بررسی می‌شدند گسسته بودند. شرط گسسته بودن ویژگی‌هایی گره‌ها را می‌توان به راحتی با تغییرات کوچکی بر طرف کرد. ویژگی‌های پیوسته را می‌توان با

¹ Standard deviation

تعریف پویای هم ارز گسسته‌ی متغیرهای پیوسته با بازه بندی به گسسته تبدیل کرد. در کل، ویژگی پیوسته‌ی A را می‌توان با ویژگی منطقی A_c جایگزین کرد. این ویژگی زمانی که $A < C$ ، درست و در غیر این صورت غلط است. حال این سؤال پیش می‌آید که بهترین روش تعیین مقدار آستانه‌ی C چیست؟

برای مثال، فرض کنید که قصد داریم ویژگی پیوسته‌ی Temperature را در نمونه‌های آموزشی کار PlayTennis در جدول ۳.۲ اضافه کنیم. فرض کنید که برای گره خاصی از درخت نمونه‌هایی با Temperature و ویژگی هدف PlayTennis زیر را داریم.

Temperature	۴۰	۴۸	۶۰	۷۲	۸۰	۹۰
PlayTennis	No	No	Yes	Yes	Yes	No

چه مقدار آستانه‌ای را باید برای Temperature در نظر گرفت؟ مسلماً ما تمایل داریم مقدار آستانه‌ای را انتخاب کنیم که بیشترین بهره‌ی اطلاعات را داشته باشد. با ترتیب کردن نمونه‌ها بر اساس ویژگی پیوسته‌ی A و پیدا کردن مقادیر نزدیک به تغییر دسته بندی تابع هدف، می‌توان مقدار آستانه‌های پیشنهادی بدست آورد. می‌توان نشان داد که مقدار C در نقطه‌ای است که بهره‌ی اطلاعات این مرز ماکزیمم است (Fayyad 1991). با بررسی بهره‌ی اطلاعات برای این مقادیر پیشنهادی مقدار آستانه، می‌توان به مقدار C لازم پی برد. در مثال حاضر دو مقدار پیشنهادی برای مقدار آستانه‌ی Temperature وجود دارد (در دو نقطه‌ای که ویژگی هدف در حوالی‌شان تغییر می‌کند): $(48+60)/2$ و $(80+90)/2$. می‌توان برای هر کدام از ویژگی‌های نظیر این مقادیر پیشنهادی $(Temperature > 54)$ و $(Temperature > 85)$ بهره‌ی اطلاعات را محاسبه کرد و ویژگی‌ای که بهره‌ی بیشتر را دارد برگزید $(Temperature > 54)$. این ویژگی منطقی پویای^۱ ایجاد شده را می‌توان در کنار دیگر ویژگی‌ها در یادگیری درختی به کار برد. (Fayyad and Irani 1993) اضافاتی به این روش اضافه کردند، آن‌ها به جای مقدار آستانه از بازه‌هایی استفاده کردند. (Murthy و Ufgoff and Brodley 1991) (1994) نیز در مورد روش‌های تعریف ویژگی‌ها با ترکیبات خطی چندین ویژگی پیوسته و آستانه بندی آن‌ها بحث کرده‌اند.

۳.۷.۳ معیارهای دیگر برای انتخاب ویژگی‌ها

در تابع بهره‌ی اطلاعات بایاسی ذاتی وجود دارد که ویژگی‌هایی که تعداد بیشتری مقدار می‌پذیرند را به دیگر ویژگی‌ها ترجیح می‌دهد. برای مثال، ویژگی تاریخ را در نظر بگیرید که تعداد بسیار زیادی مقدار می‌تواند داشته باشد (مثلاً ۴ مارس ۱۹۷۹). اگر این ویژگی را به ویژگی‌های جدول ۳.۲ اضافه کنیم بهره‌ی اطلاعات این ویژگی از همه‌ی ویژگی‌ها بیشتر خواهد بود، زیرا که تاریخ هر روز به تنهایی می‌تواند ویژگی هدف را با استفاده از نمونه‌های آموزشی مشخص کند. بنابراین، ویژگی تاریخ به عنوان ویژگی ریشه انتخاب خواهد شد و الگوریتم به درختی با عمق یک خواهد رسید که تمامی نمونه‌های آموزشی را درست دسته بندی می‌کند. البته این درخت تصمیم‌گیری بر روی نمونه‌های جدید خیلی ضعیف عمل خواهد کرد، زیرا که با وجود اینکه تمامی نمونه‌های آموزشی را درست دسته بندی می‌کند اما قدرت پیش بینی بسیار ضعیفی دارد.

مشکل ویژگی تاریخ چیست؟ بیایید ساده نگاه کنیم، مقادیر این ویژگی بسیار زیاد است و تمامی نمونه‌ها را به دسته‌های بسیار کوچکی تقسیم می‌کند. به همین خاطر، بهره‌ی اطلاعات نسبی بسیار زیادی بر روی نمونه‌های آموزشی خواهد داشت در حالی که پیش بینی‌های بسیار ضعیفی دارد.

¹ dynamic

یکی از روش‌های حل این مشکل، انتخاب ویژگی‌ها با معیاری دیگر (به غیر بهره‌ی اطلاعات) است. یکی از جایگزین‌های موفق نسبت بهره^۱ است (Quinlar 1986). معیار نسبت بهره ویژگی‌هایی چون تاریخ را با جمله‌ای به نام تقسیم اطلاعات^۲ جریمه می‌کند. این جمله به حجم و یکنواختی پخش نمونه‌ها حساس است:

$$SplitInformation(S, A) \equiv - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \quad (3.5)$$

در این رابطه S_1 تا S_c زیر مجموعه‌ی نمونه‌هایی هستند که از تقسیم S با استفاده از ویژگی c مقداری A ایجاد می‌شوند. توجه داشته باشید که $SplitInformation$ در حقیقت همان آنتروپی S با توجه به مقادیر ویژگی A است. این تعریف با تعریف قبلی ما که فقط از آنتروپی برای تعیین ویژگی‌ای که بررسی می‌شود استفاده می‌کردیم کمی تفاوت دارد.

$GainRatio$ یا همان نسبت بهره بر اساس بهره‌ی اطلاعات و تقسیم اطلاعات به شکل زیر تعریف می‌شود:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)} \quad (3.6)$$

توجه داشته باشید که جمله‌ی $SplitInformation$ احتمال انتخاب ویژگی‌هایی که تعداد زیادی مقدار دارند را کاهش می‌دهد. برای مثال، مجموعه‌ای از n نمونه که کاملاً با ویژگی A تقسیم می‌شوند را در نظر بگیرید (مثل تاریخ در مثال قبلی). در این حالت مقدار $SplitInformation$ ، $\log_2 n$ خواهد بود. در مقابل ویژگی منطقی B که همان n نمونه را به دو دسته‌ی مساوی تقسیم می‌کند مقدار $SplitInformation$ ۱ خواهد داشت. اگر دو ویژگی A و B بهره‌ی اطلاعات مساوی داشته باشند مطمئناً B نسبت بهره‌ی بیشتری خواهد داشت.

یکی از مشکلات استفاده‌ی $GainRatio$ به جای $Gain$ این است که زمانی که برای S_i داشته باشیم $|S_i| \approx |S|$ مخرج بسیار کوچک یا حتی صفر خواهد شد. در هر صورت $GainRatio$ یا بسیار بزرگ می‌شود یا تعریف نشده می‌گردد در حالی که این ویژگی تقریباً برای همه‌ی نمونه‌های S یکی است. برای پرهیز از این مشکل می‌توان ابتدا معیار $Gain$ را محاسبه کرد و سپس برای ویژگی‌هایی که این معیار از میانگین بزرگ‌تر است $GainRatio$ را محاسبه کرد (Quinlar 1986).

می‌توان برای حل مشکل مذکور به جای $GainRatio$ از معیاری دیگری که بر اساس فاصله است و توسط (Lopez de Mantaras 1991) ارائه شده استفاده کرد. این معیار بر اساس تعریف فاصله‌ی متریک قسمت‌های داده عمل می‌کند. هر ویژگی بر اساس قسمت بندی داده‌ایی که انجام می‌دهد و قسمت بندی بهینه^۳ (قسمت بندی‌ای که تمامی نمونه‌ها را درست دسته بندی می‌کند) سنجیده می‌شود و ویژگی‌ای که تشابه بسیاری به قسمت بندی بهینه دارد انتخاب خواهد شد. (Lopez de Mantaras 1991) این معیار فاصله را تعریف و اثبات می‌کند که این معیار به سمت ویژگی‌هایی که مقادیر بسیاری دارند بایاس ندارد. وی تحقیقاتی را که نشان می‌دهد درخت‌های تولیدی بر اساس

¹ gain ratio

² split information

³ perfect partition

این معیار با درخت‌هایی که بر اساس Gain و GainRatio ساخته می‌شوند تفاوتی ندارند ارائه می‌کند. با این وجود این معیار مشکلات کاربردی معیار GainRatio را ندارد و در تحقیقات وی این معیار درخت‌های بسیار کوچک‌تری برای ویژگی‌هایی با مقادیر بسیار ایجاد می‌کند.

معیار های متنوع دیگری نیز برای این مسئله ارائه شده‌اند (برای مثال، Breiman et al. 1984; Mingers 1989a; Kearns and Mingers (1989a). (Mansour 1996; Dietterich 1996). تحلیلی تحقیقی از تأثیر نسبی چندین معیار مختلف بر روی مسائل متنوع انجام داده‌است. وی اختلاف قابل توجهی را در اندازه‌ی درخت‌های هرس نشده ناشی از معیار های مختلف را گزارش می‌کند. با این وجود در زمینه های تحقیقات وی به نظر می‌رسد معیار انتخاب ویژگی‌ها تأثیر کمتری بر دقت نهایی تعمیم نسبت به متد پس هرس دارد.

۳.۷.۴ کار با نمونه های آموزشی ای که ویژگی های مجهول دارند

در بعضی موارد، در داده های موجود تمامی ویژگی ها معلوم نیست. برای مثال، در تشخیص بیماری ای که بیماری بر اساس دسته ای از آزمایشات آزمایشگاهی تشخیص داده می‌شود ممکن است "جواب آزمایش خون" (blood-Test-Result) برای دسته‌ی معدودی از بیماران در دسترس باشد. در چنین شرایطی، متداول است که این ویژگی‌های مجهول با دیگر ویژگی‌های نمونه و بر اساس دیگر نمونه‌ها تخمین زده می‌شوند.

وضعیتی را در نظر بگیرید که $Gain(S,A)$ برای یک گره n در درخت تصمیم گیری محاسبه می‌شود تا تخمین زده شود که آیا برای این گره A بهترین ویژگی است یا خیر. فرض کنید که $\langle x, c(x) \rangle$ یکی از نمونه های آموزشی مجموعه‌ی S باشد که در آن ویژگی $A(x)$ مجهول باشد.

یکی از روش‌های برخورد با این ویژگی مجهول این است که متداول‌ترین مقدار ویژگی نمونه‌هایی که به گره n می‌رسند را به آن اختصاص دهیم. یا ممکن است متداول‌ترین مقدار ویژگی را بین نمونه‌هایی که به گره n می‌رسند و مقدار تابع هدف $C(x)$ را دارند به آن اختصاص دهیم. بعد از اختصاص مقدار به این ویژگی می‌توان از نمونه‌ها برای درخت تصمیم گیری موجود استفاده کرد. این استراتژی مفصلاً در (Mingers 1989a) توضیح داده شده است.

راه حل دومی نیز وجود دارد، می‌توان از فرایند پیچیده تری (نسبت به اختصاص متداول‌ترین مقدار) برای اختصاص احتمال به هر کدام از مقادیر ممکن استفاده کرد. این احتمال‌ها را می‌توان بر اساس تعداد دفعات تکرار مقادیر مختلف A در میان نمونه های گره n مشخص کرد. برای مثال، اگر ویژگی A ویژگی ای منطقی باشد و گره n نیز ۶ نمونه با مقدار $A=1$ و ۴ نمونه با مقدار $A=0$ داشته باشد، آنگاه احتمال اینکه $A(x)=1$ را ۰.۶ و احتمال اینکه $A(x)=0$ باشد را ۰.۴ در نظر می‌گیریم. با این تقسیم بندی ۰.۶ نمونه‌هایی که های ویژگی مجهول را دارند از شاخه‌ی $A=1$ و ۰.۴ آن‌ها از شاخه‌ی $A=0$ پایین خواهند رفت. این نسبت‌ها برای محاسبه‌ی بهره‌ی اطلاعات به نمونه‌ها اختصاص داده می‌شود و ممکن است این کار در زیر درخت‌های بعدی نیز (اگر ویژگی ای معلوم نباشد) دوباره انجام گردد. همچنین می‌توان چنین نسبت‌هایی را بعد از یادگیری برای دسته بندی نمونه های جدیدی که بعضی ویژگی‌ها را ندارند اعمال کرد. در چنین حالتی این دسته بندی نمونه‌ی جدید محتمل‌ترین دسته بندی خواهد بود، این محتمل‌ترین^۱ دسته بندی با جمع وزن دار دسته بندی‌های مختلف هر گره برگ درخت انجام خواهد گرفت. از این متد برای کار با نمونه های آموزشی ای که ویژگی‌های مجهول دارند در C4.5 مورد استفاده قرار گرفته است (Quinlan 1993).

¹ Most probable

۳.۷.۵ کار با ویژگی‌های غیر هم ارزش

در بعضی از کارهای یادگیری ممکن است نمونه‌ها ویژگی‌های غیر هم هزینه ای داشته باشند. برای مثال، در مثال تشخیص بیماری ممکن است بیماران را با ویژگی‌های درجه‌ی حرارت بدن، آزمایش بافت، نبض، نتیجه‌ی آزمایش خون، و ... توصیف کنیم. این ویژگی‌ها مسلماً قیمت یکسانی ندارند، هم از نظر هزینه پولی آزمایش و هم از نظر هزینه راحتی بیمار. در چنین کارهایی درخت‌هایی را ترجیح می‌دهیم که تا جایی که ممکن باشد آزمایش‌های کم هزینه تر را انجام دهد و آزمایش‌های پر هزینه را به تشخیص‌های آخر موکول کند.

ID3 می‌تواند با اضافه کردن جمله‌ی هزینه به معیار انتخاب ویژگی‌اش این هزینه‌ها را در نظر بگیرد. برای مثال، می‌توان رابطه‌ی $Gain$ را بر هزینه تقسیم کرد تا ویژگی‌هایی که هزینه‌ی کمتر دارند ارجحیت بیشتری داشته باشند. با وجود اینکه این روش‌ها تضمین نمی‌کنند که روش بهینه ای در هزینه پیدا کنند اما بایاسی در جستجو به سمت ویژگی‌های کم هزینه تر ایجاد می‌کنند.

(Tan and Schlimmer 1990) و (Tan 1993) روشی برای این کار ابداع کردند و در کار ادراک یک ربات به کاربردن، در این کار ربات یاد می‌گرفت که چگونه اشیاء مختلف را با حس کردن آن‌ها با بازوهایش دسته بندی کند. در این کار، ویژگی‌ها خروجی‌های حسگر متحرک ربات بودند. هزینه‌ی هر ویژگی با تعداد ثانیه‌هایی که طول می‌کشید تا حسگر در آن موقعیت قرار گیرد و خروجی بدهد اندازه گیری می‌شد. آن‌ها ثابت کردند که با استفاده از معیار زیر می‌توان بدون کاهش دقت دسته بندی می‌توان مؤثرترین استراتژی تشخیص اشیاء را یاد گرفت:

$$\frac{Gain^2(S, A)}{Cost(A)}$$

(Nunez 1988) راه حل مشابهی را ارائه می‌دهد و آن را برای یادگیری تشخیص‌های پزشکی به کار می‌برد. در این کاربرد ویژگی‌ها نشانه‌های بیماری و آزمایش‌های آزمایشگاهی با هزینه‌های مختلف هستند. در سیستمی که وی ارائه داد معیار دیگری برای انتخاب ویژگی‌ها ارائه شده بود:

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

در این رابطه $w \in [0,1]$ ثابتی است که اهمیت نسبی هزینه در مقابل بهره‌ی اطلاعات را معلوم می‌کند. (Nunez 1991) به روش تجربی این دو روش را در مسایل مختلف مقایسه کرد.

۳.۸ خلاصه و منابع برای مطالعه‌ی بیشتر

نکات اصلی این فصل:

- یادگیری درختی متدی کاربردی در یادگیری مفهوم و توابع گسسته مقدار است. خانواده‌ی الگوریتم‌های مشابه ID3 شامل الگوریتم‌هایی می‌شود که درخت را از ریشه به سمت پایین حریصانه رشد می‌دهند، در هر مرحله از رشد درخت برای هر شاخه تصمیم جدید بهترین ویژگی انتخاب می‌شود.

- ID3 فضای فرضیه ای کاملی (فضای فرضیه ای تمامی درخت‌های تصمیم ممکن برای توابع گسسته مقدار) را جستجو می‌کند. به همین دلیل این الگوریتم مشکل اساسی که هنگام جستجوی دسته فضای فرضیه های محدود به وجود می‌آید، این احتمال که ممکن است تابع هدف در فضای فرضیه ای مفروض نباشد، را ندارد.
- بایاس استقرایی ID3 درخت‌های کوچک‌تر را ارجح می‌داند؛ به این معنا که درخت را فقط تا زمانی که نمونه های آموزشی را دسته بندی کند رشد می‌دهد.
- مسئله‌ی overfit بر روی نمونه های آموزشی مسئله ای مهم در یادگیری درختی است. زیرا که نمونه های آموزشی فقط نمونه ای از تمامی نمونه های ممکن هستند، ممکن است ما به درخت شاخه‌هایی را بیفزاییم که کارایی درخت را بر روی نمونه های آموزشی افزایش داده اما کارایی برای نمونه های خارج این مجموعه کاهش یابد. به همین دلیل متد های هرس درخت تصمیم برای پرهیز از overfit در یادگیری درختی (و دیگر الگوریتم‌های یادگیری‌ای که از بایاس ترجیحی استفاده می‌کنند) از اهمیت خاصی برخوردارند.
- انواع بسیاری از تغییرات برای ID3 توسط محققان ایجاد شده است. این تغییرات شامل متد های پس هرس درخت‌ها، کار با ویژگی‌های حقیقی مقدار، کار با نمونه های آموزشی‌ای که ویژگی‌های مجهول دارند، بازنگری در درخت با افزایش تعداد نمونه های آموزشی، استفاده از معیار های دیگری به جای معیار gain برای انتخاب ویژگی‌ها و در نظر گرفتن هزینه اندازه گیری ویژگی‌های نمونه‌هاست.

در میان اولین کارهایی که بر روی درخت تصمیم انجام گرفته، (Hunt et al. CLS یا Hunt's Concept Learning System) (1966) و کار (Friedman and Breiman) روی سیستم CART (Friedman 1977; Breiman et al. 1984) جزو مهم‌ترین‌ها هستند. سیستم ID3 (Quinlan 1979, 1983) نیز پایه بحث این فصل را تشکیل می‌دهد. دیگر کارهای اولیه روی یادگیری درختی شامل ASSISTANT (Kononenko et al. 1984; Cestnik et al. 1987) می‌شود. پیاده سازی الگوریتم‌های استقرایی درختی هم اکنون به صورت تجاری روی بسیاری از سیستم عامل‌ها ارائه می‌شود.

برای مطالعه‌ی بیشتر روی استقرای یادگیری درختی، کتاب (Quinlan 1993) بسیاری از مسائل عملی را بررسی کرده و کد های قابل اجرایی برای C4.5 را در بر دارد. (Mingers 1989a) و (Buntine and Niblett 1992) دو بررسی بر روی اختلاف بین روش‌های مختلف انتخاب ویژگی را بررسی می‌کنند. بررسی‌هایی که یادگیری درختی و دیگر متدهای یادگیری را مقایسه می‌کنند را می‌توان در بسیاری از مقالات شامل (Dietterich et al. 1995; Fisher and McKusick 1989; Quinlan 1988a; Shavlik et al. 1991; Thrun et al. 1991; Weiss and Kapouleas 1989) پیدا کرد.

تمرینات

۳.۱ درخت‌های تصمیمی که توابع منطقی زیر را بیان می‌کند بیابید:

$$AV \rightarrow B \quad (a)$$

$$AV[B \wedge C] \quad (b)$$

$$A \text{ XOR } B \quad (c)$$

$$[A \wedge B] \vee [C \wedge D] \quad (d)$$

۳.۲ مجموعه‌ی نمونه‌های آموزشی زیر را در نظر بگیرید:

شماره‌ی نمونه‌ی آموزشی	دسته بندی	a_1	a_2
۱	+	T	T
۲	+	T	T
۳	-	T	F
۴	+	F	F
۵	-	F	T
۶	-	F	T

(a) آنتروپی این مجموعه از نمونه‌های آموزشی با توجه به دسته بندی تابع هدف چقدر است؟

(b) بهره‌ی اطلاعات ویژگی a_2 برای این نمونه‌های آموزشی چقدر است؟

۳.۳ عبارت زیر غلط یا درست است؟

اگر درخت تصمیم D2 یک خاص سازی از D1 باشد، آنگاه D1 کلی‌تر است از D2. فرض کنید که D1 و D2 درخت‌های تصمیم متغیر منطقی دلخواهی هستند و D2 یک خاص سازی از D1 است اگر ID3 بتواند D1 را به D2 تأیید دهد. اگر جمله بالا درست است آن را اثبات کرده در غیر این صورت مثال نقض بیاورید. (مفهوم کلی‌تر بودن در فصل ۲ تعریف شده است)

۳.۴ ID3 جستجویی برای یافتن تنها یک فرضیه‌ی سازگار انجام می‌دهد در حالی که Candidate-Elimination تمامی فرضیه‌های سازگار را پیدا می‌کند. رابطه‌ای بین این دو الگوریتم یادگیری در نظر بگیرید.

(a) درخت تصمیمی که ID3 با نمونه‌های آموزشی EnjoySport یاد می‌گیرد را پیدا کنید. مفهوم هدف در جدول ۲.۱ فصل ۲ آورده شده است.

(b) رابطه‌ی بین درخت تصمیم یادگیری شده و فضای ویژه‌ی نشان داده شده در شکل ۲.۳ در فصل ۲ که از همین نمونه‌های آموزش بدست آمده چیست؟

(c) نمونه‌ی آموزشی زیر را به نمونه‌های آموزشی اضافه کرده و درخت تصمیم جدیدی را یاد بگیرید. این بار بهره‌ی اطلاعات بدست آمده برای هر ویژگی در هر مرحله از رشد درخت را تعیین کنید.

EnjoySport	Forecast	Water	Wind	Humidity	Air-Temp	Sky
No	Same	Warm	Weak	Normal	Warm	Sunny

(d) فرض کنید که می‌خواهیم یادگیری مشابه ID3 طراحی کنیم که فضایی از فرضیه‌های درخت‌های تصمیم را جستجو کرده و مشابه (Candidate-Elimination) تمامی فرضیه‌های سازگار با این داده‌ها را پیدا کند. به طور خلاصه این که، می‌خواهیم Candidate-Elimination را برای جستجوی فضای فرضیه‌ای درخت‌های تصمیم بکار ببریم. مجموعه‌های S و G را که از نمونه‌های آموزشی جدول ۲.۱ بدست می‌آیند را تعیین کنید. توجه داشته باشید که S باید خاص‌ترین درخت‌های ساخته شده با استفاده از داده‌ها و G باید کلی‌ترین درخت‌های ساخته شده را در بر بگیرد. نشان دهید که این دو مجموعه با اعمال نمونه‌ای آموزشی دوم چگونه تغییر می‌کنند (می‌توانید درخت‌هایی را که یک مفهوم را ارائه می‌کنند و فقط ساختار غیر یکسان دارند حذف کنید). چه مشکلاتی را در اعمال الگوریتم Candidate-Elimination به فضای فرضیه‌ای درخت‌های تصمیم می‌بینید؟

فرهنگ لغات تخصصی فصل (فارسی به انگلیسی)

آنترپی	Entropy
بایاس	Bias
بایاس استقرایی	inductive bias
بهره‌ی اطلاعات	information gain
تقسیم اطلاعات	split information
تئوری اطلاعات	information theory
جستجوی کم عمق	breath first search
حریصانه	Greedy
خاص	Specific
خاص‌ترین کلی‌سازی‌ها	maximal generalization
درخت تصمیم‌گیری	decision tree
دسته‌بندی	Classify
دسته‌ی تایید	validation set
زیر درخت	Subtree
سازگار	Consistent
شامل	Expressive
علامت‌گذاری	Label
غیر قطعی	Nondeterministic
فرضیه	Hypothesis
فضای ویژه	space version
کلی	General
کلی‌ترین خاص‌سازی‌ها	minimal specialization
نمونه آموزشی	training example
نمونه غیر آموزشی	unobserved instance
مسائل دسته‌بندی	classification problems
مفهوم هدف	target concept
نسبت بهره	gain ratio
نقیض	Negation

post-prune	هرس
decision tree learning	یادگیری درختی