فصل سوم: یادگیری درخت تصمیم‌گیری

یادگیری درخت[1] تصمیم‌گیری یکی از پرکاربردترین و کارآمدترین متدهای یادگیری استقرایی است. این متد در یادگیری توابع گسسته مقدار با داده‌های خطادار به کار می‌رود. در این فصل به خانواده‌ای از الگوریتم‌های یادگیری درختی، مثل الگوریتم‌های ID3، ASSISTANT و C4.5 می‌پردازیم. این متدهای یادگیری درختی فضای فرضیه‌ای کاملی را جستجو می‌کنند و مشکل محدودیت فضای فرضیه‌ای را ندارند. بایاس‌های استقرایی این الگوریتم‌ها این است که همیشه درخت‌های کوچک‌تر را بر درخت‌های بزرگ‌تر ترجیح می‌دهند (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 تصمیم‌گیری مشخص می‌شود. درخت‌های به دست آمده را نیز می‌توان به صورت دسته‌ای از دستورهای if-then نیز نمایش داد تا بررسی آن برای انسان راحت‌تر گردد. این متدها از جمله متداول‌ترین متدها در یادگیری‌های استقرایی هستند و در حوزه‌ی وسیعی از کارهای یادگیری، از یادگیری تشخیص موارد پزشکی گرفته تا تشخیص میزان ریسک وام، مورد استفاده قرار گرفته‌اند.

۳,۲ نمایش درخت تصمیم‌گیری

درخت تصمیم‌گیری با ترتیب کردن نمونه‌ها از ریشه به سمت برگ‌های درخت، نمونه‌ها را دسته‌بندی می‌کند. در این درخت هر گره ویژگی‌ای را در مورد نمونه و هر شاخه (که از آن گره خارج می‌شود) مقادیر مربوطه‌ی آن ویژگی را مشخص می‌کند. برای دسته‌بندی هر نمونه ابتدا از ریشه شروع می‌کنیم، به هر ویژگی که می‌رسیم از شاخه‌ای از درخت که ویژگی نمونه با آن مطابق است پایین می‌رویم. این فرایند برای زیر درخت‌ها نیز ادامه می‌یابد تا به دسته‌بندی نمونه برسیم.

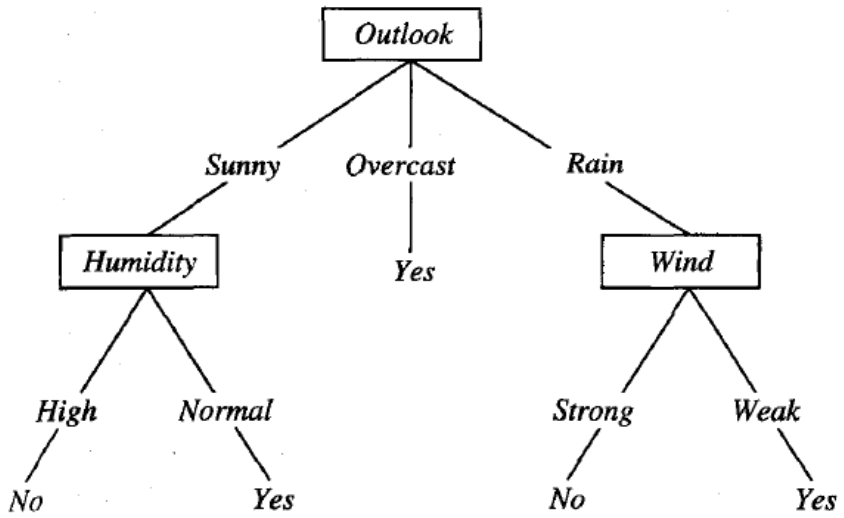
---

[1] tree

شکل ۳,۱ یک مثال از درخت تصمیم‌گیری را نشان می‌دهد. این درخت تصمیم‌گیری 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 دسته‌بندی خواهد شد (درخت پیش‌بینی 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 تصمیم‌گیری برای مفهوم PlayTennis.
نمونه‌ها با ترتیب شدن بین شاخه‌های درخت دسته‌بندی 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 دسته‌بندی خواهد کرد.

در کل درخت‌های تصمیم‌گیری روابط فصلی‌ای از عطف شروط را برای دسته‌بندی 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:

(Outlook = Sunny ∧ Humidity = Normal)

∨ (Outlook = Overcast)

∨ (Outlook = Rain ∧ Wind = Weak)

۳,۳ مسائل مناسب برای درخت تصمیم‌گیری

با وجود اینکه متدهای یادگیری درختی زیادی با نیازها و قابلیت‌های متفاوت ارائه شده است، اما اغلب یادگیری درخت تصمیم‌گیری برای مسائلی با ویژگی‌های زیر مناسب است:

- نمونه‌ها با زوج مرتب دسته ویژگی‌ها و مقدار تابع هدف مشخص شوند. نمونه‌ها در این مسائل با دسته‌ای از ویژگی‌های ثابت (مثلاً Temperature و ...) و مقادیرشان (مثل Hot) مشخص شوند. راحت‌ترین وضعیت برای یادگیری درخت تصمیم‌گیری 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 تصمیم‌گیری شکل ۳,۱ مقادیر منطقی را به هر یک از نمونه‌ها نسبت می‌دهد. متدهای 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.
- داده‌های آموزشی می‌توانند خطا داشته باشند. متدهای یادگیری درختی می‌توانند خود را با خطای موجود در داده‌های آموزشی وفق دهند، فرقی ندارد که مقدار تابع هدف نمونه بوده یا یکی از ویژگی‌ها اشتباه گزارش شده باشد.
- نمونه‌های آموزشی می‌توانند ویژگی‌های مجهول داشته باشند. یادگیری درختی را حتی زمانی که نمونه‌های 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 فوق‌اند، به همین خاطر یادگیری درخت تصمیم‌گیری بسیار پرکاربرد شده است تا جایی که در مسائلی 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 دسته‌بندی نمونه‌ها در یکی از دسته‌های موجود است، مسائل دسته‌بندی[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 فرضیه‌ای 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 فرضیه‌ای 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 پیشرفته‌تر 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.

---

[۱] classification problems

[۲] Occam’s razor

[۳] post-pruning

۳,٤ الگوریتم اساسی یادگیری درختی

اکثر الگوریتم‌هایی که برای یادگیری درختی ایجاد شده نسخه‌های مختلف یک الگوریتم اساسی هستند که از جستجویی حریصانه[۱] و بالا به پایین[۲] برای جستجوی فضای درخت‌های تصمیم‌گیری ممکن استفاده می‌کند. این روش الگوریتم ID3 نام دارد (Quinlan 1986) و تکامل‌یافته‌ی 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 جستجویی 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 زیرشاخه‌ها 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 مثبت‌اند ریشه را با + علامت‌گذاری[۳] 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 بهترین دسته‌بندی می‌کنند.*
  - *ویژگی متناسب با گره ریشه را A قرار بده.*
  - *برای هر مقدار v_i از A*

---

[۱] greedy
[۲] top-down
[۳] label

- *یک شاخه‌ی جدید در زیر ریشه متناسب با مقدار v_i اضافه کن.*
- *$Examples_{v_i}$ را زیرمجموعه‌ای از Examples قرار بده که مقدار v_i را برای ویژگی A دارند*
- *اگر $Examples_{v_i}$ تهی بود،*
  - *در زیرشاخه‌ی جدید گره برگی اضافه کن و آن را با متداول‌ترین مقدار Target_attribute در Examples علامت‌گذاری 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 دسته‌بندی کند. این فرایند آنقدر ادامه پیدا می‌کند تا درخت به طور کامل تمام نمونه‌های آموزشی را درست دسته‌بندی نماید، یا اینکه تمامی ویژگی‌ها استفاده شوند.

۳,٤,۱ کدام ویژگی بیشترین نقش را در دسته‌بندی 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 دسته‌بندی 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 دسته‌بندی نمونه‌ها بر اساس دسته‌بندی تابع هدفشان اندازه‌گیری می‌کند. ID3 از بهره‌ی اطلاعات برای انتخاب ویژگی در هر مرحله از رشد درخت استفاده می‌کند.

۳,٤,۱,۱ آنتروپی، معیار یکدستی نمونه‌ها

برای تعریف دقیق بهره‌ی اطلاعات از تعریف معیار دیگری به نام آنتروپی[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، آنتروپی دسته‌ی S متناسب با این دسته‌بندی منطقی به صورت زیر تعریف می‌شود:

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

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 آنتروپی فرض می‌کنیم که 0log0، 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 خلاصه‌سازی به طور خلاصه می‌نویسیم [-5,+9]). آنتروپی مربوط به این مجموعه‌ی زیر خواهد بود:

---

۱ information gain

۲ entropy

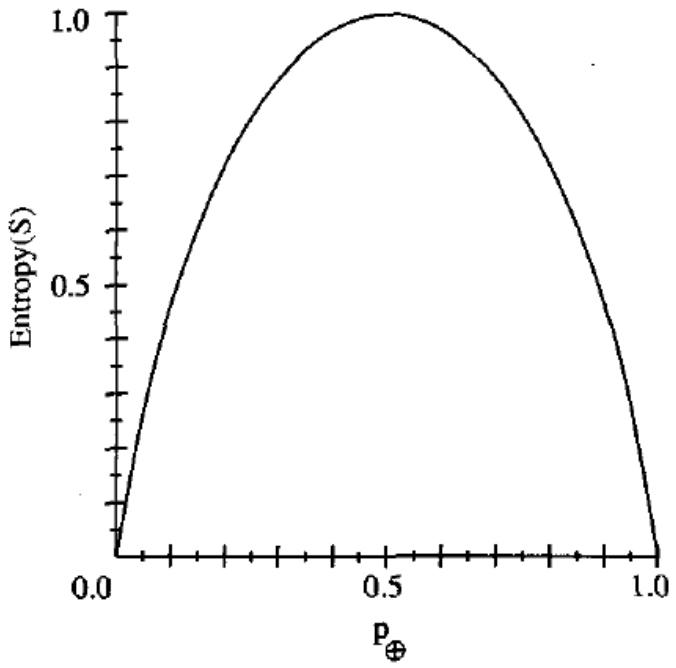
۳ information theory

۴ Homogeneity

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

$$= 0.940 \qquad (3.2)$$

توجه داشته باشید که زمانی آنتروپی صفر است که تمامی اعضای S از یک نوع دسته‌بندی 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 دسته‌بندی 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، دریافت‌کننده‌ی اطلاعات می‌داند که نمونه‌ی انتخابی حتماً مثبت خواهد بود، پس نیازی به ارسال داده‌ای نیست، آنتروپی صفر است. از سوی دیگر، اگر $p_\oplus$ ۰,۵ باشد، برای ارسال هر نمونه دقیقاً ۱ بیت لازم خواهد بود تا برای دریافت‌کننده معلوم گردد که نمونه مثبت بوده یا منفی. و اگر $p_\oplus$ ۰,۸ 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 بلندتری 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 دسته‌بندی c حالتی به صورت زیر تعریف می‌شود:

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

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.

۳,٤,۱,۲ بهره‌ی اطلاعات، معیار کاهش انتظاری آنتروپی

با داشتن آنتروپی به عنوان معیاری برای میزان یکدستی مجموعه‌ای از نمونه‌های آموزشی، حال می‌توانیم معیاری برای تأثیرگذاری یک ویژگی در دسته‌بندی نمونه‌های آموزشی ارائه دهیم. همان طور 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:

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

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 تقسیم‌بندی 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 به دست 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 صرفه‌جویی می‌شود.

برای مثال، فرض کنید که S مجموعه‌ی نمونه‌های آموزشی روزها باشد که توسط ویژگی Wind با مقادیر Strong و Weak توصیف می‌شود. همان طور 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

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

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

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

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

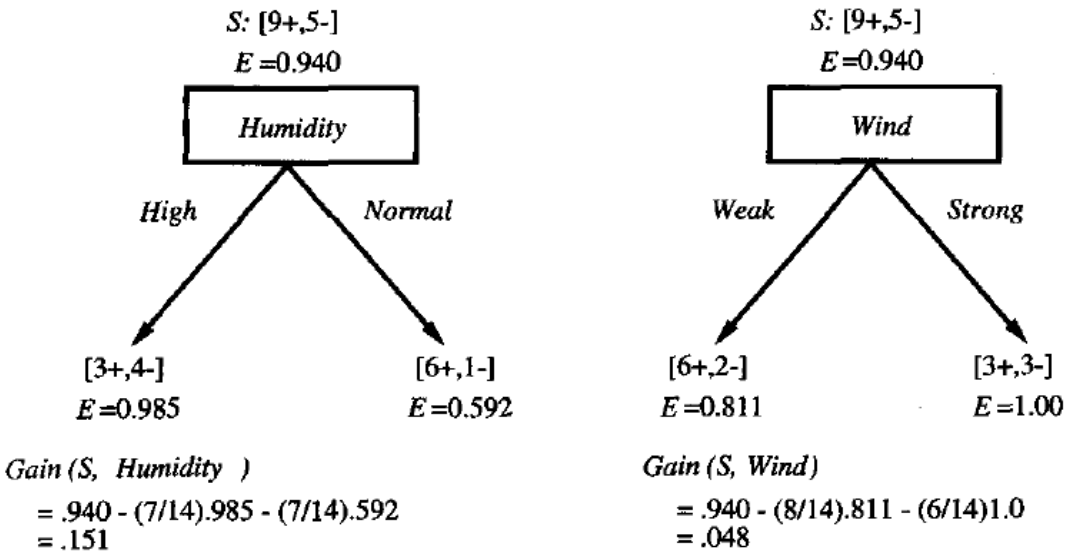
$$= Entropy(S) - \left(\frac{8}{14}\right) Entropy(S_{\text{Weak}})$$

$$- \left(\frac{6}{14}\right) Entropy(S_{\text{Strong}})$$

$$= 0.940 - \left(\frac{8}{14}\right) 0.811 - \left(\frac{6}{14}\right) 1.00$$

$$= 0.048$$

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 دسته‌بندی نمونه‌های آموزشی آمده در جدول ۳,۲ است.



شکل ۳,۳ ویژگی Humidity بهره‌ی اطلاعاتی بیشتری برای دسته‌بندی 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 زیرمجموعه‌ی [3+,4-] (برای Humidity=High) و [6+,1-] (برای Humidity=Normal) به دست می‌آید. بهره‌ی اطلاعات این تقسیم‌بندی 0.151است که از مقدار نظیر در تقسیم‌بندی بر اساس (0.048) باد بیشتر است.

۳,٤,۲ 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 مـی‌گزینـد. محاسـبه‌ی 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:

Gain(S, Sky) = 0.246

Gain(S, Humidity) = 0.151

Gain(S, Wind) = 0.048

Gain(S, 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

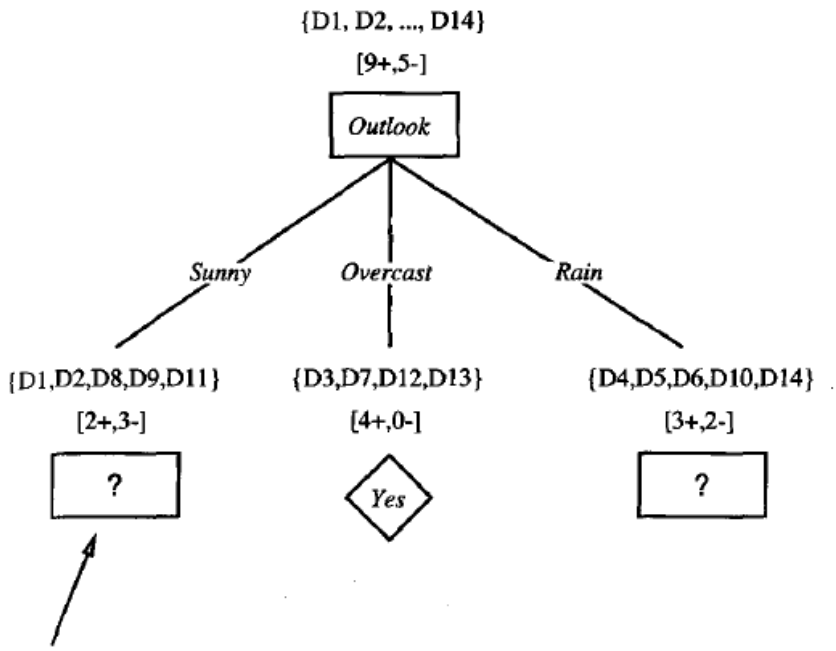
جدول ۳,۲ نمونه‌های آموزشی برای مفهوم هدف PlayTennis.

بنا بر بهره‌های اطلاعات محاسبه شده، ویژگی Outlook بیشترین تـأثیر را بـر PlayTennis بـر روی نمونـه‌هـای آموزشـی دارد. بنـابراین Outlook بهترین ویژگی برای بررسی در گره ریشه‌ی درخت اسـت، و شـاخه‌هـا نیـز مقـادیر مختلـف ایـن ویژگـی (Sunny، Cloudy و Rainy) خواهند بود. درخت حاصل در شکل ۳,۴ به همراه نمونه‌های مربوط به هر شاخه نشان داده شده است. توجه داریـد کـه بـرای تمـامی نمونه‌هایی که Outlook=Cloudy است، مقـدار PlayTennis نیـز بلـه اسـت. بنـابراین ایـن گـره از درخـت بـا PlayTennis=Yes علامت‌گذاری می‌شود. در مقابل، برای دو وضع هوای Rainy و Sunny آنتروپی صفر نیست و درخت تصمیم‌گیری در زیر این شاخه‌هـا رشـد 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 هـر ویژگـی در هر مسیر از ریشه تا برگ حداکثر یک بار ظاهر شود. این فرایند برای تمامی برگ‌های به دست آمده ادامه پیدا می‌کند تا یکی از دو شرط روبـرو درست شود: (۱) همه‌ی ویژگی‌ها استفاده شوند، (۲) نمونه‌های تمامی برگ‌ها از مقدار یکسانی از تابع هدف را داشته باشند (آنتروپـی‌شـان صـفر شود). شکل ۳,۴ محاسبه‌ی بهره‌ی اطلاعات برای مراحل بعدی رشد درخت را نشان می‌دهد. درخت کامل شـده توسـط ID3 بـرای تمـامی ۱۴ نمونه‌ی جدول ۳,۲ در شکل ۳,۱ آمده است.

---

[۱] nonterminal



S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

شکل ۳,۴ درخت نیمه‌کاره‌ای که بعد از یک مرحله اجرای ID3 به دست می‌آید.
نمونه‌های آموزشی هر گره دسته‌بندی و جدا شده‌اند. مقدار Cloudy چون تنها نمونه‌های مثبت دارد پس با Yes علامت‌گذاری شده است. دو برگ دیگر با انتخاب ویژگی‌هایی که بهره‌ی اطلاعات بیشتر (برای نمونه‌های همان شاخه) دارند باز هم توسعه خواهند یافت.

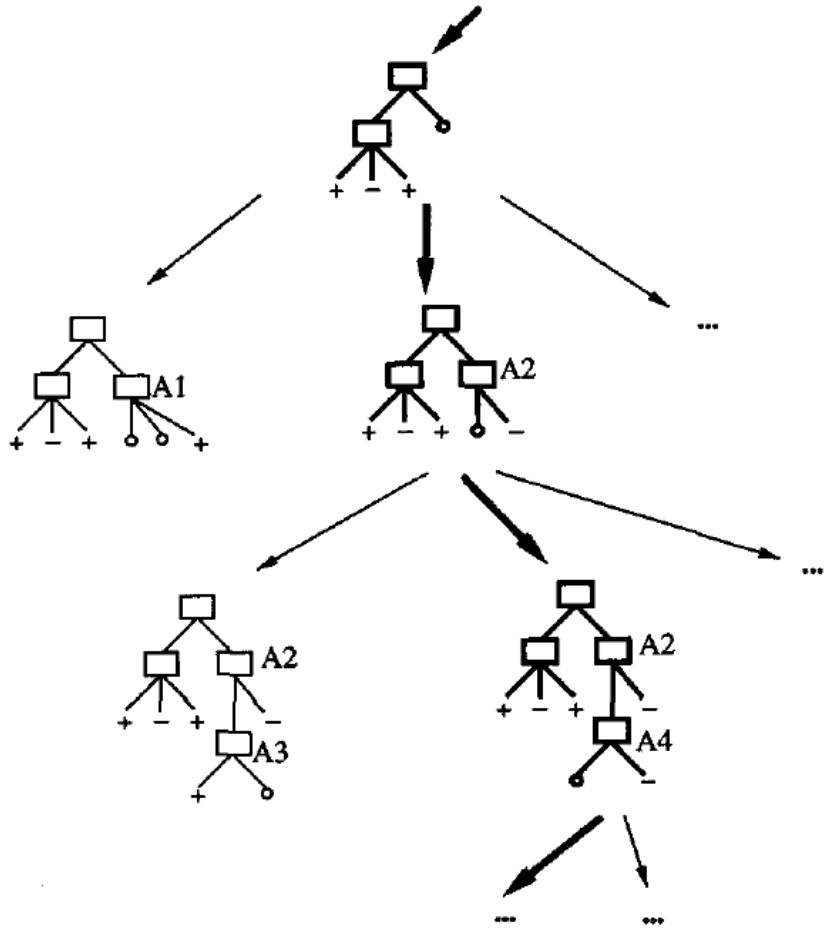
۳,۵ جستجو در فضای فرضیه‌ها در یادگیری درختی

مشابه دیگر متدهای یادگیری استقرایی، ID3 را نیز می‌توان جستجویی در میان فضایی از فرضیه‌ها برای پیدا کردن متناسب‌ترین فرضیه با نمونه‌های آموزشی در نظر گرفت. فضای فرضیه‌ای که توسط ID3 جستجو می‌گردد مجموعه‌ی تمامی درخت‌های تصمیم‌گیری است. ID3 جستجوی ساده به پیچیده[۲] و hill-climbing را در این فضای فرضیه‌ای انجام می‌دهد. ابتدای این جستجو درخت بسیار ساده‌ی تهی است، سپس با ادامه‌ی فرایند کم‌کم درخت جزئی‌تر می‌گردد تا به درختی برسد که بتواند تمامی نمونه‌های آموزشی را درست دسته‌بندی کند. تابع بهره‌ی اطلاعات این جستجوی hill-climbing را کنترل می‌کند. این جستجو در شکل ۳,۵ نشان داده شده است.

با نگاه به جنبه‌ی جستجویی و با توجه به فضای جستجو و استراتژی جستجوی ID3، تعدادی از قابلیت‌ها و محدودیت‌های آن مشاهده می‌شود:

[۲] simple-to-complex

- فضای فرضیه‌ای ID3 که همان تمامی درخت‌های تصمیم‌گیری است تمامی توابع گسسته مقدار متناهی را با توجه بـه ویژگـی‌هـای موجود در بر می‌گیرد. زیرا که هر تابع متناهی گسسته مقدار را می‌توان با درخت تصمیم‌گیری‌ای نشان داد، پـس ID3 مشـکلی (کـه ممکن است تابع هدف در فضای فرضیه‌ای نباشد) که خیلی از متدها (مثل متدهایی که فقط ترکیب‌های عطفی را در نظر می‌گیرنـد) در مورد محدودیت فضای فرضیه‌ای دارند را ندارد.



شکل ۳،۵ جستجوی ID3 در فضای فرضیه‌ای.
ID3 جستجویی ساده به پیچیده را در میان تمامی درخت‌های تصمیم‌گیری مختلف انجام می‌دهد. این جستجو توسط تابع بهره‌ی اطلاعات کنترل می‌شود.

- ID3 تنها یک فرضیه را در جستجو به دنبال فرضیه‌ی مطلوب در فضای درخت‌های تصمیم‌گیری دنبال می‌کند. این نوع جسـتجو بـا الگوریتم Candidate-Elimination که تمامی فرضیه‌های ممکن سازگار با نمونه‌های آموزشی را پیدا می‌کرد در تضاد است. با محدود شدن به یک فرضیه‌ی خاص، ID3 قابلیت معلوم کردن تمامی فرضیه‌های متناسب با نمونه‌های آموزشی را از دست می‌دهد. برای مثال، ID3 نمی‌تواند مشخص کند که چند درخت تصمیم‌گیری با نمونه‌های آموزشی مطابق‌اند، یا اینکـه نمـی‌توانـد بـین ایـن چندین درخت تصمیم‌گیری مختلف در زمان عدم هماهنگی رأی‌گیری کند.

- نسخه‌ی اصلی ID3 هیچ بازنگری‌ای[۳] به عملیات 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 خطادار آموزشی نیز کار کند، فقط کافی است که شرط خروج را از دسته‌بندی درست تمامی نمونه‌های آموزشی به دسته‌بندی درست اکثریت نمونه‌های آموزشی تغییر دهیم.

۳,٦ 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 دسته‌بندی 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 کم‌عمق[۴] (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 نمونه‌های آموزشی سازگار است، کوچک‌ترین درخت ممکن را در عمق فعلی خروجی می‌دهد (مثلاً درختی که کمترین تعداد گره را داشته باشد).

---

[۳] backtrack

[۴] breath first search

این الگوریتم به اختصار BFS-ID3 خوانده می‌شود. BFS-ID3 کوتاه‌ترین درخت تصمیم‌گیری‌ای سازگار با داده‌های آموزشی را پیدا می‌کند، یعنی دقیقاً بایاس استقرایی "درخت‌های کوتاه‌تر نسبت به درخت‌های بلندتر ارجحیت دارند" را داراست. ID3 را می‌توان تخمینی از BFS-ID3 دانست، با این تفاوت که جستجو در ID3، جستجویی حریصانه برای یافتن کوتاه‌ترین درخت ممکن است و تمامی فضای فرضیه‌ها جستجو نمی‌شود.

چون ID3 از بهره‌ی اطلاعات و استراتژی hill-climbing استفاده می‌کند بایاس پیچیده‌تری نسبت به BFS-ID3 دارد. در کل، همیشه کوتاه‌ترین درخت ممکن پیدا نمی‌شود و الگوریتم تمایل دارد که درخت‌هایی را انتخاب کند که بهره‌ی اطلاعاتشان در نزدیکی ریشه بیشتر باشد.

تخمینی مناسب‌تر از بایاس استقرایی ID3: درخت‌های کوتاه‌تری نسبت به درخت‌های بلندتر ارجحیت دارند. درخت‌هایی که بهره‌ی اطلاعات بیشتری در نزدیکی ریشه دارند نیز ارجحیت دارند.

۳,٦,۱ بایاس‌های محدود کننده و بایاس‌های مطلوب

تفاوت‌های جالبی میان دو نوع مختلف بایاس که توسط دو الگوریتم ID3 و Candidate-Elimination به کار برده می‌شود وجود دارد. به تفاوت‌های این دو روش جستجوی فضای فرضیه‌ها توجه کنید:

- ID3 فضای فرضیه‌ای کاملی را جستجو می‌کند (فضایی که تمامی توابع گسسته مقدار متناهی را می‌تواند توصیف کند). از طرفی این جستجو تمامی فضای فرضیه‌ای را شامل نمی‌شود، جستجو از فرضیه‌های ساده‌تر شروع شده و به محض رسیدن به شرط خروج پایان می‌یابد (مثلاً زمانی که فرضیه‌ای تمامی نمونه‌های آموزشی را درست دسته‌بندی می‌کند). بایاس استقرایی این الگوریتم فقط ناشی از نحوه‌ی ترتیب جستجوی این فرضیه‌هاست و بایاس دیگری وجود ندارد.
- الگوریتم Candidate-Elimination فضای فرضیه‌ای غیر کاملی را جستجو می‌کند (تنها زیرمجموعه‌ای از تمامی مفهوم‌هایی را که می‌توان از نمونه‌ها یاد گرفت). اما این فضای فرضیه‌ای را کامل جستجو می‌کند و تمامی فرضیه‌هایی که با نمونه‌های آموزشی سازگار هستند را پیدا می‌کند. بایاس استقرایی این الگوریتم فقط ناشی از میزان شمول فضای فرضیه‌ای آن است و استراتژی جستجویش هیچ نقشی در بایاس ندارد.

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

پس، بایاس استقرایی ID3 یک ترجیح فرضیه‌ها بر دیگر فرضیه‌هاست، بدون اینکه فضای فرضیه‌ای هیچ محدودیتی ایجاد کند. این نوع بایاس را معمولاً بایاس ترجیحی[۵] (یا بایاس جستجویی[۶]) می‌نامند. در مقابل، بایاس Candidate-Elimination کاملاً به فضای فرضیه‌ای در نظر گرفته شده وابسته است. این نوع بایاس را نیز بایاس محدودیتی[۷] (یا بایاس زبانی[۸]) می‌نامند.

---

[۵] preference bias
[۶] search bias
[۷] restriction bias
[۸] 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 پسندیده‌تر است؟ بایاس ترجیحی یا بایاس محدودیتی؟

معمولاً، بایاس‌های ترجیحی به بایاس‌های محدودیتی ترجیح داده می‌شوند، زیرا که به یادگیر اجازه می‌دهند تا در فضای فرضیه‌ای 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، برنامه‌ای 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 دسته‌ای 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 غیرخطی 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.

۳,٦,۲ چرا فرضیه‌های کوتاه‌تر 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 آن را توجیه کرد. حال چرا باید فرضیه‌های ساده‌تر ارجح باشند؟ توجه دارید که دانشمندان بعضی مواقع چنین بایاسی را مورد استفاده قرار می‌دهند. برای مثال، فیزیکدانان نظریه‌های ساده‌تر درباره‌ی حرکت سیارات را ترجیح می‌دهند. چرا؟ یکی از استدلال‌های ممکن این است که تعداد فرضیه‌های ساده‌تر نسبت به فرضیه‌های پیچیده‌تر بسیار کمتر است، پس به نظر می‌رسد احتمال اینکه فرضیه‌ای 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 گره‌دارند بسیار بیشتر از درخت‌هایی است که ۵ گره‌دارند. اگر ۲۰ نمونه‌ی آموزشی داشته باشیم، تعداد بسیار زیادی از درخت‌های ۵۰۰ گره‌ای با آن‌ها سازگارند، در حالی که جای تعجب ندارد که فقط یک درخت ۵ گره‌ای 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

زیرمجموعه‌های کوچک بسیاری از فضای فرضیه‌ها وجود دارد که چنین تعداد کمی را دارند و پیدا کردن همه‌ی آن‌ها ساده نیست. پس چرا باید باور داشته باشیم زیرمجموعه‌ی درخت‌هایی که طول کوتاه‌تری دارند باید نسبت به آن دیگر زیرمجموعه‌های کوچک برتری داشته باشد؟

اشکال دومی که درباره‌ی این استدلال برای تیغ Occam[۹] پیش می‌آید این است که اندازه‌ی یک فرضیه با روش خاصی مشخص می‌شود که در یادگیر تعبیه شده. اگر دو یادگیر با روش‌های مختلف اندازه‌گیری اندازه‌ی فرضیه بر روی یک مسئله به کار گرفته شوند در آخر فرضیه‌های خروجی متفاوتی خواهند داشت، در حالی که هر دو عملیات خود را توسط تیغ Occam توجیه شده می‌دانند. برای مثال، تابعی که در شکل ۳,۱ نشان داده شده است را می‌توان با درختی با یک گره نیز نشان داد، درختی که یادگیر برای دسته‌بندی نمونه‌ها از ویژگی XYZ استفاده می‌کند، ویژگی منطقی XYZ زمانی درست است که نمونه، نمونه‌ی مثبتی باشد و در غیر این صورت غلت است. بنابراین دو یادگیر که هر دو از تیغ Occam استفاده می‌کنند اگر یکی ویژگی XYZ و دیگری ویژگی‌های Outlook، Temperature، Humidity و Wind را استفاده کنند درخت‌های خروجی متفاوتی خواهند داشت.

این بحث آخر نشان می‌دهد که تیغ Occam در دو یادگیر که از یک مجموعه نمونه‌های آموزشی یکسان استفاده می‌کنند و فقط نمایش داخلی نمونه‌هایشان متفاوت است دو فرضیه‌ی 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[۱۰] یا انتخاب طبیعی[۱۱] ایجاد شود را در نظر بگیرید. جمعیتی از یادگیرهای مصنوعی‌ای را که از طریق فرایندهای تکاملی زاد و ولد، جهش و انتخاب به وجود آمده‌اند را در نظر بگیرید. و بیایید فرض کنیم که این فرایند تکاملی می‌تواند سیستم‌های ادراکی این یادگیرها را از نسلی به نسلی تغییر دهد، مشابه تغییر ویژگی‌های داخلی‌ای که عوامل جهان اطراف را با آن‌ها درک می‌کنند. و برای بحث، فرض کنیم که این عوامل یادگیری از الگوریتم یادگیری یکسانی (مثلاً ID3) استفاده می‌کنند که با تکامل تغییر نخواهد یافت. منطقی است که فرض کنیم در طول زمان، تکامل نمایش‌های 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 می‌توان انتظار داشت که نمایش‌های داخلی‌ای که خوب با الگوریتم یادگیری و بایاس استقرایی‌اش کار می‌کنند ایجاد شوند. اگر گونه‌هایی از یادگیرها از الگوریتم یادگیری از که از بایاس استقرایی تیغ Occam استفاده می‌کنند، وجود داشته باشد، انتظار خواهیم داشت که تکامل نمایش‌های داخلی‌ای را ایجاد کند که تیغ Occam برایشان استراتژی موفقی است. نکته اصلی بحث در اینجا این است که تکامل نمایش‌های داخلی‌ای را ایجاد خواهد کرد که بایاس الگوریتم یادگیری نوعی خود توجیهی[۱۲] داشته باشد، زیرا که می‌تواند بسیار راحت‌تر از تغییر الگوریتم، نمایش را تغییر دهد.

فعلاً بحث مربوطه‌ی تیغ Occam را رها می‌کنیم. اما در فصل ۶، در قسمتی که قانون کمترین طول توضیح را بررسی خواهیم کرد، نسخه‌ای از تیغ Occam را که با چارچوب بیزی توجیه می‌شود را بررسی خواهیم کرد.

---

[۹] Occam’s razor

[۱۰] evolution

[۱۱] Natural selection

[۱۲] self-fulfilling

۳,۷ مشکلات یادگیری درختی

مشکلات کاربردی یادگیری درختی شامل، مشخص کردن حداکثر عمق درخت، چگونگی بررسی ویژگی‌های پیوسته، انتخاب معیار انتخاب ویژگی‌ها، یادگیری با داده‌هایی با بعضی ویژگی‌های مجهول، یادگیری با ویژگی‌های غیر هم هزینه و بهینه کردن محاسبات می‌شود. در ادامه 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).

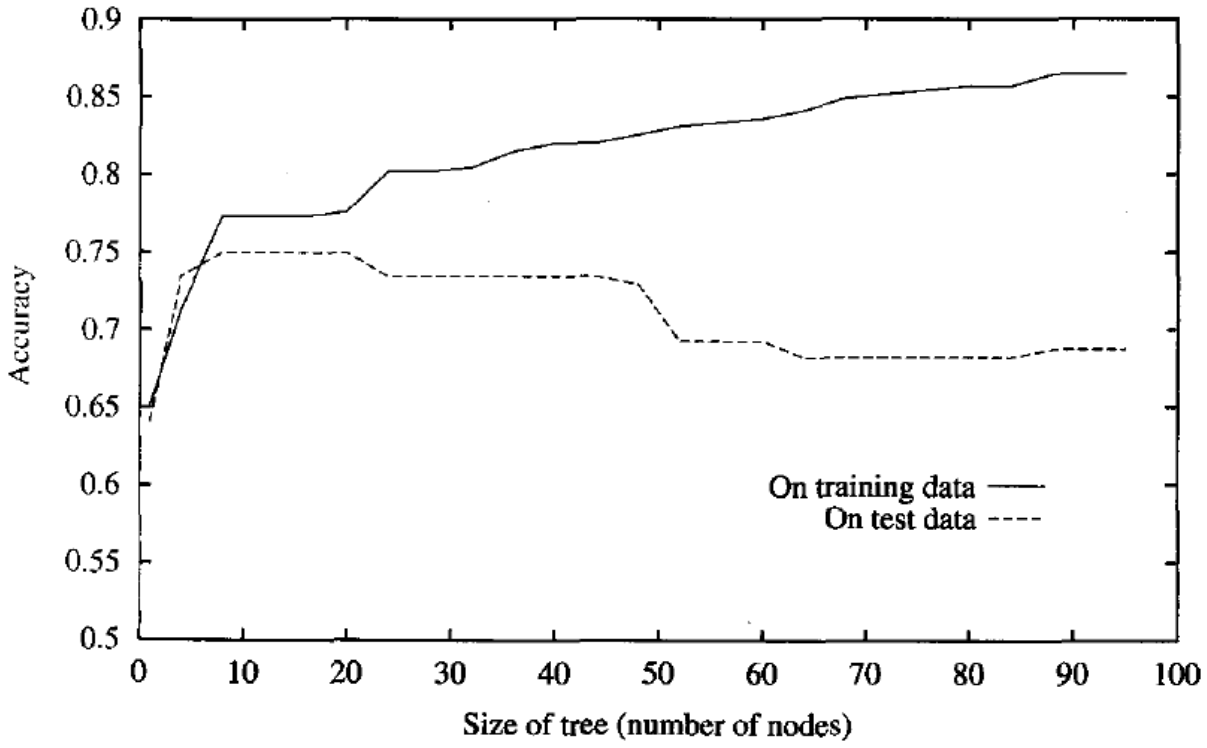
۳,۷,۱ حل مشکل overfit

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 مشکل‌ساز 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 خطچین میزان دقت را بر روی دسته‌ی دیگری از نمونه‌ها نشان می‌دهد (دسته‌ای به جز نمونه‌های آموزشی). همان طور که پیش‌بینی می‌شد دقت درخت بر روی نمونه‌های آموزشی با افزایش اندازه‌ی درخت افزایش می‌یابد. با این وجود، دقت دسته‌ی دیگر کاهش می‌یابد. همان طور 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 ۳٫۶ overfit در یادگیری درختی.

همین طور که ID3 گره‌های بیشتری برای رشد درخت به آن اضافه می‌کند، به طور مشابه دقت بر روی نمونه‌های آموزشی افزایش پیدا می‌کند. با این وجود، زمانی که دقت بر روی دسته‌ای از نمونه‌های غیر آموزشی بررسی می‌شود، ابتدا افزایش و سپس کاهش مشاهده می‌شود. برنامه و داده‌های استفاده شده برای این آزمایش در http://www.cs.cmu.edu/~tom/mllbook.html موجود است.

چگونه ممکن است درخت h که نسبت h′ عملکرد بهتری بر روی نمونه‌های آموزشی دارد، در کل نمونه‌ها عملکرد ضعیف‌تری داشته باشد؟ یکی از مواردی که چنین مشکلی ایجاد می‌شود مواقعی است که نمونه‌های آموزشی خطای تصادفی[۱۳] یا همان نویز داشته باشند. برای تصور، نمونه‌ی آموزشی مثبت زیر را در نظر بگیرید که اشتباهاً نمونه‌ی منفی در نظر گرفته شده:

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

با دادن داده‌های آموزشی بدون خطا به ID3 درخت نشان داده شده در شکل ۳٫۱ به دست می‌آید. با این وجود اگر این نمونه‌ی اشتباه را به نمونه‌های آموزشی اضافه کنیم، ID3 درخت پیچیده‌تری خروجی می‌دهد. در کل، نمونه‌ی جدید در برگ دوم شاخه‌ی سمت چپ شکل ۳٫۱ قرار می‌گیرد، 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 می‌دهد. البته، تا زمانی که نمونه‌ی اشتباه این گونه باشد (فقط مقدار تابع هدف اشتباه تعیین شده باشد)، ID3 ویژگی‌ای خواهد یافت تا نمونه غلت را از نمونه‌ی درست جدا کند. نتیجه این خواهد بود که ID3 درخت تصمیم‌گیری h ای را خروجی می‌دهد که پیچیده‌تر از درخت تصمیم‌گیری درست h′ است (شکل ۳٫۱). البته h بر روی نمونه‌های آموزشی دقت زیادی دارد در حالی که h′ ساده‌تر آن میزان دقت را ندارد. با این وجود، با داشتن گره‌ای که جدیداً اضافه شده و مستقیماً تأثیر نمونه‌ی خطادار بوده، انتظار داریم که h از h′ دقت کلی بهتری داشته باشد.

---

[۱۳] Random error

مثال بالا نشان داد که چگونه داده‌های خطادار آموزشی می‌توانند باعث overfit شوند. در واقع، overfit حتی زمانی که نمونه‌های آموزشی خطا ندارند نیز ممکن است اتفاق بیفتد، مخصوصاً زمانی که تعداد نمونه‌ها با تعداد برگ‌های درخت برابر می‌شود. در چنین شرایطی، دور از انتظار نیست که نظم‌های اتفاقی در درخت پدیدار شوند، در این نظم‌ها به نظر می‌رسد که ویژگی‌های خاصی در دسته‌بندی نمونه‌ها تأثیر بسیار زیادی دارند در حالی که آن ویژگی‌ها هیچ ربطی به تابع هدف ندارند. هر گاه چنین نظم‌های اتفاقی‌ای ایجاد می‌شود، احتمال overfit نیز بالا می‌رود.

مشکل overfit مشکل قابل‌توجهی در یادگیری درختی و بسیاری از متدهای یادگیری دیگر است . برای مثال، در یک مطالعه‌ی آزمایشی ID3 که بر روی ۵ کار یادگیری و با داده‌های خطادار و غیرقطعی[۱۴] انجام شد (Mingers 1989b)، در اکثر مسائل overfit دقت را بین ۱۰ تا ۲۵ درصد کاهش داد.

روش‌های بسیاری برای حل مسئله‌ی overfit در یادگیری درختی موجود است. این روش‌ها به دو دسته‌ی کلی تقسیم می‌شوند:

- روش‌هایی که جلوی رشد درخت را قبل از رسیدن به نقطه‌ای که تمامی نمونه‌ها را درست دسته‌بندی کند می‌گیرند،
- روش‌هایی که اجازه می‌دهند تا درخت به اندازه‌ی دلخواه رشد کند سپس درخت را هرس[۱۵] می‌کنند.

با وجود اینکه به نظر می‌رسد روش‌های دسته‌ی اول مستقیم‌ترند، اما روش‌های دسته‌ی دوم در کاربرد موفقیت بیشتری را از خود نشان داده‌اند. 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 اندازه‌ی اصلی می‌رسد، سؤال کلیدی این است که معیار درست اندازه نهایی ی درخت چیست؟ روش‌های زیر برای جواب به این سؤال پیشنهاد می‌شوند:

- استفاده از دسته‌ای از نمونه‌های اضافی (که با نمونه‌های آموزشی تداخل ندارند) برای تخمین کارایی گره‌ها و هرس آن‌ها.
- استفاده از تمام نمونه‌های موجود برای آموزش، استفاده از آزمونی آماری برای تخمین اینکه آیا رشد (یا هرس) یک گره از درخت تعمیمی را ایجاد می‌کند یا تنها باعث overfit می‌شود. برای مثال، (Quinlar 1986) از آزمون کای اسکوار (کی دو یا همان مجذور خی)[۱۶] برای جواب سؤال "آیا رشد یک گره به کارایی کلی درخت کمک می‌کند یا فقط باعث سازگاری با نمونه‌ی آموزشی می‌شود؟" استفاده می‌کند.
- استفاده از معیاری برای اندازه‌گیری پیچیدگی. در نظر گرفتن نوعی کد سازی برای درخت و متوقف کردن رشد درخت زمانی که این اندازه‌ی کد کمینه می‌شود. این روش مبتنی بر توجیه است که "کمترین طول توضیح"[۱۷] نامیده می‌شود و مفصلاً در فصل ۶ بررسی شده. برای اطلاعات بیشتر به (Quinlar and Rivest 1989) و (Mehta 1995) مراجعه کنید.

روش اول متداول‌ترین روش است و گاهی روش آموزش و مجموعه‌ی تأیید[۱۸] نیز نامیده می‌شود. در اینجا به دو نسخه‌ی اصلی این روش می‌پردازیم. در این روش، داده‌های موجود به دو دسته تقسیم می‌شوند: دسته‌ی آموزشی، که از آن‌ها برای آموزش درخت استفاده می‌شود، و

---

[۱۴] nondeterministic
[۱۵] post-prune
[۱۶] chi-square
[۱۷] Minimum Description Length principle
[۱۸] training and validation set

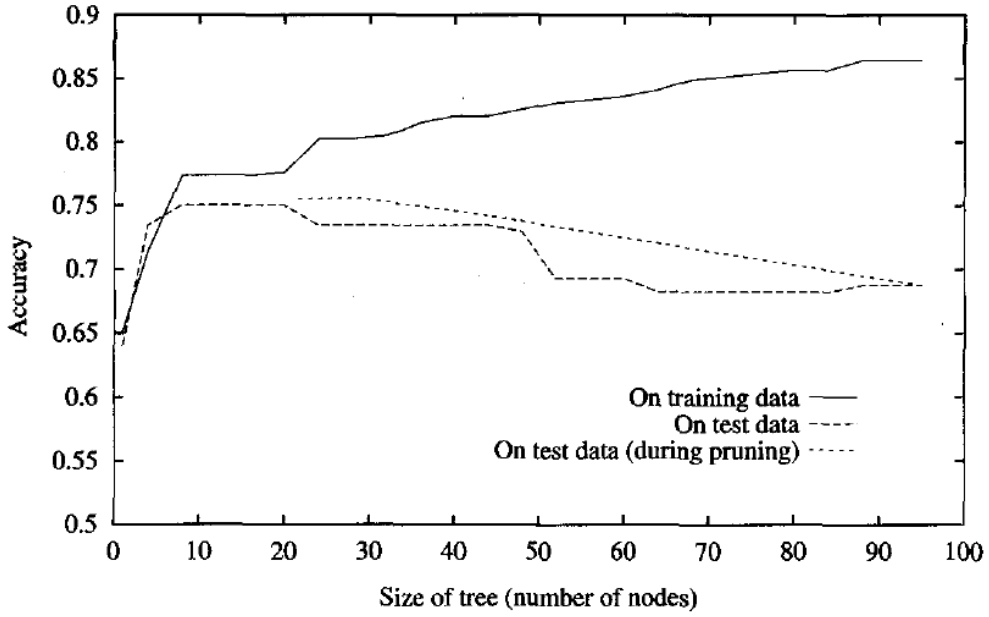
دسته‌ی تأیید[۱۹]، که از آن برای ارزیابی دقت فرضیه‌ها استفاده می‌شود، در کل، برای ارزیابی تأثیر هرس از این داده‌ها استفاده می‌شود. انگیـزه‌ی اولیه‌ی این روش این است که اگر چه ممکن است یادگیر با داده‌های خطادار گمراه شود، و به نظم‌های تصـادفی در میـان نمونـه‌های آموزشـی میل کند، اما احتمال اینکه دسته‌ی تأیید نیز همان نظم‌های اتفاقی که نمونه‌های آموزشی دارند را داشته باشد بسیار کم است. بنابراین، می‌تـوان انتظار داشت که دسته‌ی تأیید معیار خوب و مطمئنی در مقابل معیار غلت داده‌های آموزشی به ما بدهد. البته، مهم اسـت کـه انـدازه‌ی دسـته‌ی تأیید به اندازه‌ی کافی بزرگ باشد تا بتواند به تنهایی نمونه‌ی کاملی از نمونه‌ها را داشته باشد. یکی از ایده‌های معمول این است که یک‌سوم کل داده‌ها را برای دسته‌ی تأیید نگه می‌دارند و از دوسوم دیگر برای آموزش درخت استفاده می‌کنند.

۳,۷,۱,۱ کاهش خطا با هرس کردن

دقیقاً چگونه می‌توان با استفاده از یک دسته‌ی تأیید 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 یـک گـره تصـمیم باعـث حذف شدن زیر درخت متصل به گره مذکور و تبدیل آن به برگ تبدیل خواهد شد با علامت متداول‌ترین دسته‌بندی از داده‌های آموزشی می‌شود. گره‌هایی هرس خواهند شد که هرس شدنشان باعث تأثیر منفی در دسته‌بندی دسته‌ی تأیید نشود. این عمل باعث حذف برگ‌هایی کـه بـر اثـر نظم‌های تصادفی داده‌های آموزشی ایجاد شده‌اند می‌شوند، زیرا که احتمال تکرار همان نظم‌هـای تصـادفی در دسـته‌ی تأییـد نـاچیز اسـت. در چندین مرحله گره‌ها هرس می‌شوند، در هر مرحله گره‌ای که هرسش باعث حداکثر افزایش دقت درخت در دسته‌بندی دسـته‌ی تأییـد مـی‌شـود هرس می‌شود. هرس کردن آنقدر ادامه خواهد یافت تا زمانی که هرس کردن درخت اثر منفی داشته باشد (دقت دسته‌بندی دسته‌ی تأیید را کـم کند).

اثر reduce-error pruning بر روی دقت درخت تصمیم‌گیری در شکل ۳,۷ نشان داده شده است. مشابه شکل ۳,۶ دقـت درخـت بـرای نمونه‌های آموزشی و غیر آموزشی در هنگام هرس نشان داده شده است. منحنی اضافه شده دقت را بر روی دسته‌ی تست در درخت هرس شده نشان می‌دهد. زمانی که هرس کردن آغاز می‌شود درخت در حداکثر اندازه‌ی خودش است. با ادامه‌ی هرس کردن تعداد گره‌های درخت کـاهش و دقت بر روی دسته‌ی تست افزایش می‌یابد. در اینجا داده‌های موجود به سه دسته تقسیم شده‌اند: نمونه‌های آموزشی، نمونه‌های دسته‌ی تأیید، و دسته‌ی تست. از دسته‌ی آخر برای بررسی دقت درخت بر روی نمونه‌های جدید (قدرت تعمیم درخت) استفاده می‌شود. نمودار نشان داده شـده دقت را بر روی نمونه‌های آموزشی و دسته‌ی تست نشان می‌دهد. دقت دسته‌ی ارزیابی که برای هرس کردن از آن استفاده مـی‌شـود در شـکل نشان داده نشده است.

[۱۹] validation set



شکل ۳,۷ اثر reduced-error pruning در درخت تصمیم‌گیری.

شکل همان منحنی‌های دقت نمونه‌های آموزشی و دسته‌ی تست را نشان می‌دهد (شکل ۳,۶). 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 تأیید 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 مؤثری است. مانع اصلی این روش این است محدودیت تعداد داده‌هاست. گاهی اوقات کم کردن قسمتی از داده‌ها برای استفاده در دسته‌ی تأیید 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 موفقیت‌آمیز بوده است. تکنیک‌های دیگری نیز از جمله بخش‌بندی داده‌ها در دسته‌های متعدد با ترکیب‌های مختلف در دفعات متعدد و میانگین‌گیری در میان درخت‌ها، ارائه شده است. بررسی‌های تجربی دیگر متدهای هرس در (Migers 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[۲۰] 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 هم‌ارز 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 دسته‌بندی نمونه‌های جدید این سری را در نظر بگیر.

---

[۲۰] post pruning

برای تصور، دوباره درخت تصمیم‌گیری 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 دسته‌بندی برگ نیز، حکم قانون خواهد بود. برای مثال، برای چپ‌ترین مسیر درخت شکل ۳,۱ قانون زیر به دست می‌آید:

IF (Outlook = Sunny) /\ (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 اندازه‌گیری دقت قوانین استفاده از دسته‌ی تأیید 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[۲۱] این دقت تخمینی را با فرض توزیع دوجمله‌ای 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 تصمیم‌گیری 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 دست‌نخورده 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.

---

[۲۱] Standard deviation

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 تعریف پویای هم‌ارز 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 غلت است. حال این سؤال پیش می‌آید که بهترین روش تعیین مقدار آستانه‌ی c چیست؟

برای مثال، فرض کنید که قصد داریم ویژگی پیوسته‌ی Tempereture را در نمونه‌های آموزشی کار PlayTennis در جدول ۳,۲ اضافه کنیم. فرض کنید که برای گره خاصی از درخت نمونه‌هایی با Tempereture و ویژگی هدف PlayTennis زیر را داریم.

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

چه مقدار آستانه‌ای را باید برای Tempereture 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 دسته‌بندی تابع هدف، می‌توان مقدار آستانه‌های پیشنهادی به دست 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): 2/(48+60) و 2/(80+90). 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 بر گزید ($Temperature_{>54}$). این ویژگی منطقی پویای[۲۳] 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. (Ufgoff and Brodley 1991) و (Murthy 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 دسته‌بندی می‌کند. البته این درخت تصمیم‌گیری بر روی نمونه‌های جدید خیلی ضعیف عمل خواهد کرد، زیرا که با وجود اینکه تمامی نمونه‌های آموزشی را درست دسته‌بندی می‌کند اما قدرت پیش‌بینی بسیار ضعیفی دارد.

---

[۲۳] dynamic

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 پیش‌بینی‌های بسیار ضعیفی دارد.

یکی از روش‌های حل این مشکل، انتخاب ویژگی‌ها با معیاری دیگر (به غیر بهره‌ی اطلاعات) است. یکی از جایگزین‌های موفق نسبت بهره[۲۳] است (Quinlar 1986). معیار نسبت بهره ویژگی‌هایی چون تاریخ را با جمله‌ای به نام تقسیم اطلاعات[۲۴] جریمه می‌کند. این جمله به حجم و یکنواختی پخش نمونه‌ها حساس است:

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

در این رابطه S_1 تا S_c، c زیرمجموعه‌ی 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:

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

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 ی 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 قسمت‌بندی

---

[۲۳] gain ratio

[۲۴] split information

دادهایی که انجام می‌دهد و قسمت‌بندی بهینه[۲۵] (قسمت‌بندی‌ای که تمامی نمونه‌ها را درست دسته‌بندی می‌کند) سنجیده می‌شود و ویژگی‌ای که تشابه بسیاری به قسمت‌بندی 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 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.

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

۳,۷,٤ کار با نمونه‌های آموزشی‌ای که ویژگی‌های مجهول دارند

در بعضی موارد، در داده‌های موجود تمامی ویژگی‌ها معلوم نیست. برای مثال، در تشخیص بیماری‌ای که بیماری بر اساس دسته‌ای 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 تصمیم‌گیری محاسبه می‌شود تا تخمین زده شود که آیا برای این گره A بهترین ویژگی است یا خیر. فرض کنید که <x,c(x)> یکی از نمونه‌های 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 تصمیم‌گیری موجود استفاده کرد. این استراتژی مفصلاً در (Migers 1989a) توضیح داده شده است.

راه حل دومی نیز وجود دارد، می‌توان از فرایند پیچیده‌تری (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 نمونه‌های گره n مشخص کرد. برای مثال، اگر ویژگی A ویژگی‌ای 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 تقسیم‌بندی ۰,۶ نمونه‌هایی که 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 ویژگی‌ای معلوم نباشد) دوباره انجام گردد. همچنین می‌توان چنین نسبت‌هایی را بعد از یادگیری برای دسته‌بندی نمونه‌های جدیدی که بعضی ویژگی‌ها را ندارند اعمال کرد. در چنین حالتی این دسته‌بندی نمونه‌ی جدید محتمل‌ترین دسته‌بندی

---

[۲۵] perfect partition

خواهد بود، این محتمل‌ترین[۲۶] دسته‌بندی با جمع وزن‌دار دسته‌بندی‌های مختلف هر گره برگ درخت انجام خواهد گرفت. از این متد برای کار با نمونه‌های آموزشی‌ای که ویژگی‌های مجهول دارند در C4.5 مورد استفاده قرار گرفته است (Quinlan 1993).

۳,۷,۵ کار با ویژگی‌های غیر هم ارزش

در بعضی از کارهای یادگیری ممکن است نمونه‌ها ویژگی‌های غیر هم هزینه‌ای 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 کم‌هزینه‌تر را انجام دهد و آزمایش‌های پرهزینه 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 بهینه‌ای در هزینه پیدا کنند اما بایاسی در جستجو به سمت ویژگی‌های کم‌هزینه‌تر ایجاد می‌کنند.

(Tan and Schlimmer 1990) و (Tan 1993) روشی برای این کار ابداع کردند و در کار ادراک یک ربات به کاربردند، در این کار ربات یاد می‌گرفت که چگونه اشیاء مختلف را با حس کردن آن‌ها با بازوهایش دسته‌بندی 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 اندازه‌گیری می‌شد. آن‌ها ثابت کردند که با استفاده از معیار زیر می‌توان بدون کاهش دقت دسته‌بندی می‌توان مؤثرترین استراتژی تشخیص اشیا 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 مسائل مختلف مقایسه کرد.

۳,۸ خلاصه و منابع برای مطالعه‌ی بیشتر

نکات اصلی این فصل:

---

[۲۶] Most probable

- 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 فرضیه‌ای کاملی (فضای فرضیه‌ای تمامی درخت‌های تصمیم ممکن برای توابع گسسته مقدار) را جستجو می‌کند. به همین دلیل این الگوریتم مشکل اساسی که هنگام جستجوی دسته فضای فرضیه‌های محدود به وجود می‌آید، این احتمال که ممکن است تابع هدف در فضای فرضیه‌ای مفروض نباشد، را ندارد.
- بایاس استقرایی ID3 درخت‌های کوچک‌تر را ارجح می‌داند؛ به این معنا که درخت را فقط تا زمانی که نمونه‌های آموزشی را دسته‌بندی کند رشد می‌دهد.
- مسئله‌ی overfit بر روی نمونه‌های آموزشی مسئله‌ای مهم در یادگیری درختی است. زیرا که نمونه‌های آموزشی فقط نمونه‌ای از تمامی نمونه‌های ممکن هستند، ممکن است ما به درخت شاخه‌هایی را بیفزاییم که کارایی درخت را بر روی نمونه‌های آموزشی افزایش داده اما کارایی برای نمونه‌های خارج این مجموعه کاهش یابد. به همین دلیل متدهای 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 متدهای پس هرس درخت‌ها، کار با ویژگی‌های حقیقی مقدار، کار با نمونه‌های آموزشی‌ای که ویژگی‌های مجهول دارند، بازنگری در درخت با افزایش تعداد نمونه‌های آموزشی، استفاده از معیارهای دیگری به جای معیار gain برای انتخاب ویژگی‌ها و در نظر گرفتن هزینه اندازه‌گیری ویژگی‌های نمونه‌هاست.

در میان اولین کارهایی که بر روی درخت تصمیم انجام گرفته، (Hunt's Concept Learning System) یا CLS (Hunt et al. 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. پیاده‌سازی الگوریتم‌های استقرایی درختی هم اکنون به صورت تجاری روی بسیاری از سیستم‌عامل‌ها ارائه می‌شود.

برای مطالعه‌ی بیشتر روی استقرای یادگیری درختی، کتاب (Quinlan 1993) بسیاری از مسائل عملی را بررسی کرده و کدهای 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:

(a) $A \vee \neg B$

(b) $A \vee [B \wedge C]$

A XOR B (c)

[A∧B]∨[C∧D] (d)

۳,۲ مجموعه‌ی نمونه‌های آموزشی زیر را در نظر بگیرید:

شماره‌ی نمونه‌ی آموزشی	دسته‌بندی	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 دسته‌بندی تابع هدف چقدر است؟

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

۳,۳ عبارت زیر غلت 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 وD2 یک خاص سازی از D1 است اگر ID3 بتواند D1 را به D2 تامیم 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 به دست آمده چیست؟

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

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

(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 بریم. مجموعه‌های S و G را که از نمونه‌های آموزشی جدول ۲,۱ به دست 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 نمونه‌ی 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)

آنتروپی	Entropy
بایاس	Bias
بایاس استقرایی	inductive bias
بهره‌ی اطلاعات	information gain
تقسیم اطلاعات	split information
تئوری اطلاعات	information theory
جستجوی کم‌عمق	breath first search
حریصانه	Greedy
خاص	Specific
خاص‌ترین کلی سازی‌ها	maximal generalization
درخت تصمیم‌گیری	decision tree
دسته‌بندی	Classify
دسته‌ی تأیید	validation set
زیر درخت	Subtree
سازگار	Consistent
شامل	Expressive
علامت‌گذاری	Label
غیرقطعی	Nondeterministic
فرضیه	Hypothesis
فضای ویژه	space version
کلی	General
کلی‌ترین خاص سازی‌ها	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
هرس	post-prune
یادگیری درختی	decision tree learning