فصل یازدهم: یادگیری تحلیلی

متدهای یادگیری استقرایی مثل شبکه‌های عصبی و یادگیری درختی برای رسیدن به حدی از دقت در تعمیم نیاز به تعداد قابل‌توجهی نمونه آموزشی دارند، این نیاز در مرزهای تئوری و نتایج آزمایشگاهی تأثیر خواهد گذاشت. یادگیری تحلیلی از دانش قبلی و استدلال استنتاجی برای جمع‌آوری اطلاعات نمونه‌های آموزشی استفاده می‌کند، بنابراین به محدودیت تعداد نمونه‌های آموزشی وابسته نیست. در این فصل به متد یادگیری تحلیلی‌ای به نام یادگیری توضیحی[۱] (EBL) می‌پردازیم. در یادگیری توضیحی دانش قبلی برای بررسی و یا توضیح هر یک از نمونه‌های آموزشی مشاهده شده به کار می‌رود. سپس از توضیحات داده شده برای تشخیص ویژگی‌های مرتبط[۲] از ویژگی‌های غیر مرتبط[۳] نمونه‌های آموزشی استفاده می‌شود، سپس می‌توان تعمیم را بر اساس روش‌های منطقی[۴] به جای روش‌های آماری پیش برد. یادگیری توضیحی برای یادگیری قوانین کنترل جستجو و کارهای برنامه‌ریزی و انجام[۵] با موفقیت به کار گرفته شده است. در این فصل با فرض اینکه دانش قبلی همه‌جانبه[۶] و درست[۷] است به یادگیری توضیحی می‌پردازیم. و در فصل بعدی به ترکیب یادگیری تحلیلی و استقرایی برای مسائلی که دانش قبلی فقط تا حدی درست است می‌پردازیم.

۱۱,۱ معرفی

در فصول گذشته انواع مختلف متدهای یادگیری استقرایی، متدهایی که تابع را از طریق نمونه‌های آموزشی و پیدا کردن خواصی که یک نمونه را مثبت یا منفی می‌کند می‌آموزند، را بررسی کردیم. متدهای یادگیری مثل درخت تصمیم‌گیری، شبکه‌های عصبی، برنامه‌نویسی منطقی

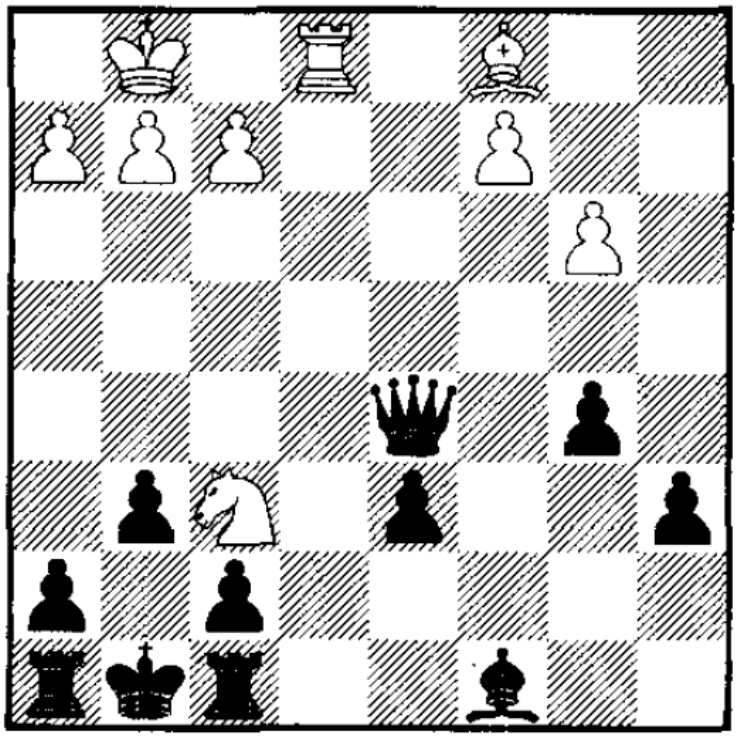
---

[۱] explanation-based learning
[۲] relevant
[۳] irrevelant
[۴] logical
[۵] planning and schedualing
[۶] complete
[۷] correct

استقرایی و الگوریتم‌های ژنتیک همگی نمونه‌های متدهای استقرایی هستند. ضعف کلیدی متدهای استقرایی در نیاز به تعداد قابل‌توجهی نمونه‌های آموزشی است. در واقع همان‌طور که در فصل ۷ نیز گفته شد، بررسی‌های تئوری نشان می‌دهد که محدودیت‌های پایه‌ای‌ای در دقت این نوع متدها هنگامی که تعدادی نمونه آموزشی محدود است وجود دارد.

آیا می‌تواند متدهای یادگیری‌ای ایجاد کرد که این محدودیت‌های پایه‌ای را نداشته باشد و با استفاده از داده‌های آموزشی موجود کار کند؟ جواب مثبت است به شرط اینکه در خود تعریف مسئله‌ی یادگیری تجدیدنظر کنیم. یکی از راه‌های ممکن ایجاد الگوریتم‌های یادگیری‌ای است که دانشی قبلی را علاوه بر نمونه‌های آموزشی دریافت کنند. یادگیری توضیحی چنین متدی است. این متد از دانش قبلی برای بررسی و توضیح هر یک از نمونه‌های آموزشی استفاده می‌کند تا بفهمد که کدام ویژگی‌های نمونه‌های آموزشی مربوط و کدام ویژگی‌ها نامربوط به تابع هدف‌اند. چنین اطلاعاتی باعث می‌شود تا این الگوریتم بتواند بهتر از سیستم‌های استقرایی که فقط مبتنی بر نمونه‌های آموزشی هستند، نمونه‌ها را بررسی کند. همان‌طور که در فصول گذشته نیز دیدیم، سیستم‌های برنامه‌نویسی منطقی استقرایی مثل Cigol دانشی اضافی نیز برای جهت‌دهی به یادگیری دارند. با این وجود، چنین سیستم‌هایی از دانش قبلی برای تغییر توصیفات نمونه‌های ورودی استفاده می‌کنند و پیچیدگی فضای فرضیه‌ای جستجو را افزایش می‌دهند. در مقابل، یادگیری توضیحی از این دانش قبلی کاهش پیچیدگی فضای فرضیه‌ای مورد جستجو کمک می‌گیرد، بنابراین، پیچیدگی نمونه‌ای کاهش یافته و دقت تعمیم یادگیر نیز افزایش خواهد یافت.

برای شهود لازم برای یادگیری توضیحی، یادگیری بازی شطرنج را در نظر بگیرید. در کل، فرض کنید که می‌خواهیم برنامه‌ی ما مجموعه‌ی پوزیسیون‌های مهمی را یاد بگیرد، مثل مفهوم هدف "پوزیسیون‌هایی که سیاه در دو حرکت وزیر خود را از دست خواهد داد". شکل ۱۱,۱ پوزیسیونی از این مفهوم هدف را نشان می‌دهد. البته می‌توان از یادگیری استقرایی برای یادگیری این تابع هدف استفاده کرد. با این حال، چون صفحه‌ی تقریباً پیچیده است (۳۲ مهره و ۶۴ خانه) و الگوهایی که به این مفهوم ختم می‌شوند تقریباً پیچیده‌اند (به مکان نسبی بسیاری از مهره‌ها در صفحه بستگی دارد)، باید هزاران نمونه‌ی آموزشی به یک سیستم استقرایی بدهیم تا بتوانیم انتظار داشته باشیم که آن نمونه‌های جدید را درست دسته‌بندی کند.



شکل ۱۱,۱ نمونه‌ای مثبت از مفهوم هدف "پوزیسیون‌هایی که سیاه در دو حرکت وزیر خود را از دست خواهد داد".
توجه داشته باشید که اسب همزمان به شاه و وزیر حمله کرده (چنگال اسب)، پس سیاه باید شاه خود را حرکت دهد و در حرکت بعدی سفید وزیر سیاه را خواهد گرفت.

یکی از نکات جالب در مورد یادگیری شطرنج این است که انسان‌ها فقط با دیدن چند مثال چنین توابع هدفی را یاد می‌گیرند! در واقع، بعد دیدن همین شکل ۱۱,۱ اکثر افراد فرضیه‌ی مناسب را پیدا می‌کنند "پوزیسیون‌هایی که در آن شاه و وزیر هم‌زمان زیر حمله قرار می‌گیرند". هیچ‌کس فرضیه‌ی دیگری مثل "پوزیسیون‌هایی که در آن چهار سرباز سفید در خانه‌ی اولیه‌ی خود باشند" را پیشنهاد نمی‌دهد. چگونه آدمی می‌تواند فقط با دیدن یک مثال تعمیم موفقی انجام دهد؟

جواب در این است که یادگیری آدمی متکی بر تفسیر نمونه‌ها با دانش قبلی است، در اینجا این دانش قبلی قوانین شطرنج در مورد نحوه‌ی حرکت مهره‌هاست. اگر از یادگیر انسانی بپرسید که چرا شکل ۱۱,۱ یک نمونه مثبت از مفهوم "پوزیسیون‌هایی که سیاه در دو حرکت وزیر خود را از دست خواهد داد" است اکثر افراد جوابی مشابه خواهند داد: "زیرا که اسب سفید همزمان به شاه و وزیر حمله کرده است، سیاه باید از کیش خارج شود، پس در حرکت بعدی اسب وزیر را خواهد گرفت". اهمیت چنین توضیحاتی به این است که اطلاعات لازم برای تعمیم موفق از جزئیات نمونه‌های آموزشی در پی بردن به تابع هدف را در خود دارند. ویژگی‌های نمونه‌ی آموزشی که در توضیح آمد (مثل مکان اسب سفید، شاه سیاه و وزیر سیاه) مرتبط به تابع هدف‌اند و باید در توضیح فرضیه‌ی تعمیم استفاده شوند. در مقابل، ویژگی‌هایی که در توضیح نیامده (مثل اینکه که سیاه ۴ سوار دارد) نباید توجه شود.

دانش لازم برای یادگیر ساختن چنین توضیحی برای این مثال شطرنجی چیست؟ فقط کافی است حرکت‌های قانونی مهره‌های شطرنج را بدانیم: بدانیم که اسب و بقیه‌ی مهره‌ها چگونه در صفحه حرکت می‌کنند، بدانیم که دو بازیکن نوبت به نوبت مهره‌های خود را حرکت می‌دهند، و بدانیم که اگر بازیکنی شاه حریف را بگیرد برنده می‌شود. توجه داشت باشید که فقط با این دانش قبلی در کل از نظر تئوری می‌توان حرکت بهینه را برای هر پوزیسیون پیدا کرد. با این وجود، در عمل این محاسبات می‌تواند بسیار پیچیده باشد، با وجود اینکه آدمی علم زیادی درباره‌ی

شطرنج دارد اما با این حال هنوز نمی‌تواند آن را به طرز بهینه بازی کند. پس اکثر یادگیری انسانی در شطرنج (و دیگر بازی‌هایی که نیاز به نقشه کشیدن دارد) نیاز به فرایندی بلند برای پیدا کردن نتایج دانش قبلی، و نمونه‌های آموزشی دارد.

در این فصل به توصیف الگوریتم‌های یادگیری‌ای می‌پردازیم که به طور خودکار چنین توضیحاتی را ساخته و بر اساس آن‌ها مفاهیمی یاد خواهند گرفت. در ادامه‌ی این قسمت مسئله‌ی یادگیری تحلیلی را به صورت دقیق‌تر تعریف خواهیم کرد. در قسمت بعد، به الگوریتم توضیحی خاصی به نام Prolog-EBG خواهیم پرداخت. و در قسمت‌های بعدی نیز خواص کلی این الگوریتم و رابطه‌ی آن را با الگوریتم‌های یادگیری استقرایی، را که در دیگر فصل‌ها آورده شده‌اند، بررسی خواهیم کرد. در آخر نیز به کاربرد یادگیری توضیحی برای بهبود کارایی در مسائلی با فضای حالت بزرگ[1] خواهیم پرداخت. در سراسر این فصل فرض می‌کنیم که توضیحات از دانش قبلی کاملاً درست (مثل دانش انسان‌ها) ناشی می‌شوند، مثل مثال شطرنج بالا. در فصل ۱۲ به حالتی کلی‌تر که در آن دانش قبلی تقریباً درست است خواهیم پرداخت.

۱۱,۱,۱ مسائل یادگیری استقرایی و یادگیری تحلیلی

تفاوت‌های اصلی بین متدهای یادگیری استقرایی و متدهای تحلیلی این است که آن‌ها دو تصور متفاوت از مسئله دارند:

- در یادگیری استقرایی، به یادگیر فضای فرضیه‌های H و دسته مثال‌های $D = \{< x_1, f(x_1) >, \dots, < x_n, f(x_n) >\}$ داده می‌شود تا فرضیه‌ای از H مثل h را که سازگار با نمونه‌های آموزشی D است پیدا کند.
- در یادگیری تحلیلی، ورودی یادگیر شامل همان فضای فرضیه‌ای H و مثال‌های D است. علاوه بر این ورودی‌ها، ورودی دیگری نیز به شکل روبرو به یادگیر داده می‌شود: یک تئوری قلمرو[2] مثل B که دانش قبلی‌ای به یادگیر می‌دهد تا بتواند نمونه‌های آموزشی را تجزیه‌وتحلیل کند. خروجی یادگیر فرضیه‌ای مثل h از H خواهد بود که با نمونه‌های آموزشی D و تئوری قلمرو B سازگار باشد.

برای تصور، در مثال ذکر شده x_i ها پوزیسیون‌ها هستند و $f(x_i)$ زمانی که پوزیسیون نمونه‌ی مثبتی از "پوزیسیون‌هایی که در آن شاه و وزیر هم‌زمان زیر حمله قرار می‌گیرند" درست و در غیر این صورت غلت است. فضای فرضیه‌ای H را می‌توان دسته گزاره‌های تعریف شده در فصل ۱۰ (قانون‌های if-then) که در آن شروط جای نسبی مهره‌ها در صفحه‌ی شطرنج است در نظر گرفت. تئوری قلمرو B را نیز می‌توان قوانین شطرنج در نظر گرفت. این قوانین شامل نحوه‌ی حرکت مهره‌ها، تغییر نوبت حرکت، و این حقیقت است که زمانی که کسی شاهش را از دست بدهد می‌بازد می‌شود.

توجه داشته باشید که در یادگیری تحلیلی، یادگیر باید فرضیه‌ای را به عنوان خروجی ارائه کند که هم با نمونه‌های آموزشی و هم با تئوری قلمرو سازگار باشد. زمانی که می‌گوییم فرضیه‌ی h با تئوری قلمروی B سازگار است که تئوری قلمروی B فرضیه‌ی h را رد نکند ($B \not\vdash \neg h$). این شرط آخر باعث می‌شود ابهام H در زمان‌هایی که یادگیر با نمونه‌های آموزشی‌ای مواجه است که H را به اندازه‌ی کافی محدود نمی‌کنند کم شود. اثر کلی که توسط تئوری قلمرو ایجاد می‌شود، دقت درستی فرضیه‌ی خروجی را افزایش می‌دهد.

بیایید مسئله‌ی دیگری از یادگیری تحلیلی را معرفی کنیم، مسئله‌ای که در ادامه‌ی فصل برای توضیحات به کار می‌رود. فضای نمونه‌های X را در نظر بگیرید که هر نمونه در این فضا جفتی از اشیاء است. هر یک از این دو شی با خواص رنگ، حجم، صاحب، جنس، نوع و چگالی و رابطه‌ی بین دو شی نیز با خاصیت On مشخص شده است. با معلوم بودن فضای نمونه‌ها می‌خواهیم مفهوم هدف "جفت اجسامی که یکی را

---

[1] large state-space search problems

[2] domain theory

می‌توان با اطمینان به روی دیگری گذاشت" را که با نماد SafeToStack(x,y) یاد بگیریم. یادگیری چنین مفهومی می‌تواند بسیار سودمند باشد، برای مثال، برای یک ربات که می‌خواهد اجسام را در فضای محدودی انبار کند این مفهوم بسیار مهم است. تعریف کامل این مسئله در جدول ۱۱,۱ آمده است.

ورودی‌های:

- فضای نمونه‌های X: هر نمونه جفتی از اجسام است که با خواص Type، Color، Volume، Owner، Material، Density و On مشخص می‌شوند.
- فضای فرضیه‌های H: هر فرضیه دسته‌ای از قوانین Horn clause است. سر هر Horn clause یک عبارت شامل SafeToStack است. در بدنه‌ی هر horn clause رابطه‌ای عطفی از عملگرهایی است که بر پایه‌ی همان خواص نمونه‌ها عمل می‌کنند. این عملگرها شامل توابع LessThan، Equal، و GreaterThan و توابعی مثل plus، minus و times می‌شود. برای مثال:

SafeToStack(x, y) ←Volume(x, vx) /\ Volurne(y, vy) /\ LessThan(vx, vy)

- مفهوم هدف: SafeToStack(x,y)
- نمونه‌های آموزشی: نمونه‌هایی مثل نمونه‌ی آموزشی زیر SafeToStack(Obj1,Obj2):

On(Obj1,Obj2)	Owner(0bj1 , Fred)
Type(0bj1 , Box)	Owner(Obj2, Louise)
Type(Obj2, Endtable)	Density(0bj1,0.3)
Color(Obj1, Red)	Material(Obj1, Cardboard)
Color(Obj2, Blue)	Material (Obj2, Wood)
Volume(Obj1,2)	

Domain Theory B:

SafeToStack(x, y) ←¬Fragile(y)

SafeToStack(x, y) ←Lighter(x, y)

Lighter(x, y) ← Weight(x, wx)/\ Weight(y, wy)/\ LessThan(wx, wy)

Weight(x, w) ← Volume(x, v) /\ Density(x,d) /\ Equal(w, times(v, d))

Weight(x,5) ← Type(x, Endtable)

Fragile(x) ← Material (x, Glass)

...

- خروجی:

فرضیه‌ای از H را پیدا کن که هم با نمونه‌های آموزشی و هم با تئوری قلمرو سازگار باشد.

جدول ۱۱,۱ مثالی از مسئله‌های یادگیری تحلیلی

همان‌طور که در جدول ۱۱,۱ نیز نشان داده شده است فضای فرضیه‌های H تمامی قوانین درجه اول if-then یا همان horn clause در نظر گرفته شده است (در سراسر این فصل از همان نمادگذاری و اصطلاحات تعریف شده در جدول ۱۰,۳ استفاده می‌کنیم). برای مثال، نمونه‌ی horn clause ی که در جدول ۱۱,۱ آمده است، بیان می‌کند که جسم x را می‌توان روی جسم y گذاشت اگر که حجم x کمتر از (LessThan) حجم y باشد (در این نمایش Vx و Vy به ترتیب حجم جسم x و حجم جسم y در نظر گرفته شده‌اند). توجه داشته باشید که در بیان horn clause ها می‌توانند از تمامی ویژگی‌های نمونه‌ها و تمامی توابع معرفی شده استفاده کرد. یک نمونه مثبت SafeToStack(Obj1,Obj2) نیز در جدول ۱۱,۱ آمده است.

برای فرمولی کردن این یادگیری تحلیلی باید ابتدا تئوری قلمروی‌ای را پیدا کنیم که توضیح دهد که چرا نمونه‌ی مثبت، نمونه‌ی مثبتی از مفهوم هدف مذکور است. در مثال پوزیسیون شطرنجی که در اول فصل بیان کردیم، توضیحی از اینکه چرا یک نمونه، نمونه‌ی مثبت است آورده شده بود. در این مثال نیز تئوری قلمرو باید توضیح دهد که چرا نمونه‌ی مذکور، نمونه‌ای مثبت است، به عبارت دیگر باید توضیح دهد که چرا اجسامی با چنین ویژگی‌هایی را می‌توان روی هم قرار داد. تئوری قلمروی بیان شده در جدول تعریف‌هایی همچون "جسم x را می‌توان روی جسم y قرار داد هر گاه که y شکننده (fragile) نباشد" و "جسم x زمانی شکننده است که جنس (material) آن شیشه (glass) باشد" ارائه شده است. مشابه فرضیه‌های یادگیر، تئوری قلمرو نیز با استفاده از دسته‌ای horn clause تعریف شده است تا یادگیر بتواند فرضیه‌ها را با تئوری قلمرو مقایسه کند. توجه داشته باشید که تئوری قلمرو ویژگی‌های جدیدی مثل Lighter و Fragile را نیز مورد استفاده قرار داده که جزو ویژگی‌های نمونه‌ها نیستند اما آن‌ها را نیز با استفاده از ویژگی‌های Material، Volume و Density تعریف کرده است. در آخر توجه داشته باشید که تئوری قلمروی نشان داده شده در جدول می‌تواند اثبات کند که نمونه‌ی مثبت یک نمونه‌ی مثبت است (توضیح لازم را ارائه می‌کند).

۱۱,۲ یادگیری با تئوری قلمروهای کامل: Prolog-EBG

همان‌طور که پیش‌تر نیز گفتیم، در این فصل به یادگیری توضیحی‌ای که بر اساس تئوری قلمروهای کامل‌اند[۳] می‌پردازیم. یعنی اینکه تئوری قلمروهای مورد استفاده هم درست[۴] هم همه‌جانبه‌اند زمانی می‌گوییم یک تئوری قلمرو درست است که تمامی رابطه‌هایش در جهان واقعی قابل‌اطمینان باشد و زمانی می‌گوییم یک تئوری قلمرو همه‌جانبه است که با توجه به فضای نمونه‌ها و تابع هدف تمامی نمونه‌های مثبت را پوشش دهد. به عبارت دیگر، اگر یک تئوری قلمرو تمامی نمونه‌های که تابع هدف را راضی می‌کند را توضیح دهد همه‌جانبه است. توجه داشته باشید که تعریف ما از همه‌جانبه بودن حرفی در مورد اینکه توضیحی برای نمونه‌های منفی داشته باشد نمی‌زند. با این وجود اگر قرار داد

[۳] perfect

[۴] correct

prolog را قبول کنیم، باید نمونه‌هایی که توضیحی برای مثبت بودن ندارند را منفی دسته‌بندی کنیم، بنابراین تئوری قلمرو تمامی نمونه‌ها را پوشش خواهند داد.

ممکن است این سؤال را بپرسید که آیا منطقی است که فرض کنیم که چنین تئوری قلمروی کاملی در اختیار یادگیر است؟ به هر حال، با وجود چنین تئوری قلمروی کاملی چرا باید به فکر یادگیری باشیم؟ دو جواب برای چنین سؤالی وجود دارد:

- اول اینکه حالاتی وجود دارد که می‌توان چنین تئوری قلمروی کاملی را پیدا کرد. مثال شطرنجی که در ابتدای فصل زدیم یکی از این حالات است که در آن نحوه‌ی حرکت مهره‌ها تئوری قلمرو کامل را تشکیل می‌دهد که با استفاده از آن می‌توان بازی کردن بهینه را آموخت. علاوه بر آن، با وجود اینکه می‌توان به راحتی نحوه‌ی حرکت مهره‌ها را روی کاغذ نوشت، تئوری قلمروی کامل، اما نمی‌توان به سادگی نحوه‌ی بهینه شطرنج بازی کردن را روی کاغذ نوشت. در چنین حالاتی، ما ترجیح می‌دهیم که تئوری قلمرو را برای یادگیر پیدا کنیم و فرموله کردن مفهوم هدف را به یادگیر واگذار کنیم تا توضیحی قابل‌استفاده (مثل "پوزیسیون‌هایی که در حرکت بعدی وزیرم را از دست خواهم داد") از تابع هدف بدهد و با بررسی و تعمیم از این نمونه‌های آموزشی تخمین خوبی از تابع هدف بدهد. در قسمت ۱۱,۴ کاربردهای موفقیت‌آمیز یادگیری توضیحی با تئوری قلمروهای کامل را در افزایش کارایی در مسائل متمرکز بر جستجوی برنامه‌ریزی و بهینه‌سازی آورده‌ایم.
- دوم اینکه، در بسیاری از حالات فرض اینکه یک تئوری قلمروی کامل وجود دارد بی‌دلیل است. حتی نوشتن یک تئوری درست و همه‌جانبه در مورد مسئله‌ی ساده‌ی SafeToStack سخت است. فرض واقع‌بینانه این است که فرض کنیم توضیحاتی قابل‌قبول[۵] از تئوری قلمروی غیر کامل داریم نه اینکه خود تئوری قلمرو را در دسترس داشته باشیم. با این وجود، با درک ایده‌ی تئوری قلمروهای کامل نقش توضیحات در یادگیری را خواهیم فهمید. در فصل ۱۲ یادگیری با تئوری قلمروهای غیر کامل را بررسی خواهیم کرد.

در این قسمت الگوریتم Prolog-EBG (Kedar-Cabelli and McCarty 1987) را ارائه خواهیم داد که پایه‌ی بسیاری از الگوریتم‌های یادگیری توضیحی است. Prolog-EBG الگوریتمی ترتیبی است (رجوع کنید به فصل ۱۰). به عبارت دیگر این الگوریتم با یک قانون horn clause شروع می‌کند و سپس نمونه‌های پوشانده شده‌ی قانون را حذف کرده و دوباره این فرایند را برای نمونه‌های مثبت باقیمانده ادامه می‌دهد تا تک‌تک نمونه‌های مثبت پوشش داده شوند. زمانی که تئوری قلمرو درست و همه‌جانبه باشد، Prolog-EBG تضمین می‌کند که فرضیه‌ای (دسته قوانین) را خروجی دهد که هم درست باشند و هم نمونه‌های مثبت آموزشی را پوشش دهد. برای هر دسته از نمونه‌های آموزشی فرضیه‌ی خروجی Prolog-EBG دسته‌ای از عبارات منطقی کافی را بر اساس تئوری قلمرو ایجاد می‌کند. Prolog-EBG تجدیدنظری در EBG است که توسط (Mitchell 1986) معرفی شده و شباهت زیادی به الگوریتم EGGS (DeJong and Mooney 1986) دارد. الگوریتم Prolog-EBG در جدول ۱۱,۲ آورده شده است.

۱۱,۲,۱ یک مثال

برای تصور دوباره نمونه‌های آموزشی و تئوری قلمروی جدول ۱۱,۱ را در نظر بگیرید، الگوریتم Prolog-EBG الگوریتمی ترتیبی است که مرحله به مرحله تعداد بیشتری از نمونه‌های آموزشی را پوشش می‌دهد. برای هر نمونه‌ی مثبت آموزشی که هنوز توسط horn clause ها پوشش نداده شده است، horn clause جدیدی تشکیل می‌شود: (۱) این horn clause باید توضیحی برای نمونه‌ی جدید داشته باشد، (۲)

---

[۵] plausible explanations

این توضیح باید بررسی شود تا تعمیم لازم انجام شود و (۳) horn clause جدید به فرضیه‌ی فعلی اضافه می‌شود تا فرضیه نمونه‌ی آموزشی جدید را علاوه بر نمونه‌های قبلی پوشش دهد. در زیر هر کدام از این مراحل را بررسی خواهیم کرد.

۱۱,۲,۱,۱ توضیح دادن نمونه‌های آموزشی

مرحله‌ی اول هر یادگیری نمونه‌ی آموزشی پیدا کردن توضیحی برای آن با استفاده از تئوری قلمرو است که توضیح دهد که چرا این نمونه‌ی مثبت مفهوم هدف را راضی می‌کند. زمانی تئوری قلمرو درست و همه‌جانبه است پس باید اثباتی برای اینکه چرا نمونه مفهوم هدف را راضی می‌کند داشته باشد. اما زمانی که تئوری قلمرو کامل نیست، نمایش توضیحات باید طوری تغییر کنند که قابل‌قبول باشد و مقادیر را نیز تخمین بزند (فقط به جای اثبات محض نباشد).

Prolog-EBG(TargetConcept, TrainingExamples, DomainTheory)

- {} ⟶ LearnedRules
- Pos ⟶ نمونه‌های مثبت از مجموعه‌ی TrainingExamples
- برای هر نمونه‌ی مثبت از Pos که توسط LearnedRules پوشانده نمی‌شوند:

۱. توضیح:

- Explanation ⟶ توضیحی را که چرا TargetConcept PossitiveExample را راضی می‌کند.

۲. بررسی:

- SufficientConditions ⟶ کلی‌ترین دسته ویژگی‌های PossitiveExample که برای راضی کردن TargetConcept بر اساس Explanation لازم است.

۳. بازنگری:

- NewHornClause + LearnedRules ⟶ LearnedRules ، در این رابطه NewHornClause به صورت زیر است:

"SufficientConditions ⟶ TargenConcept"

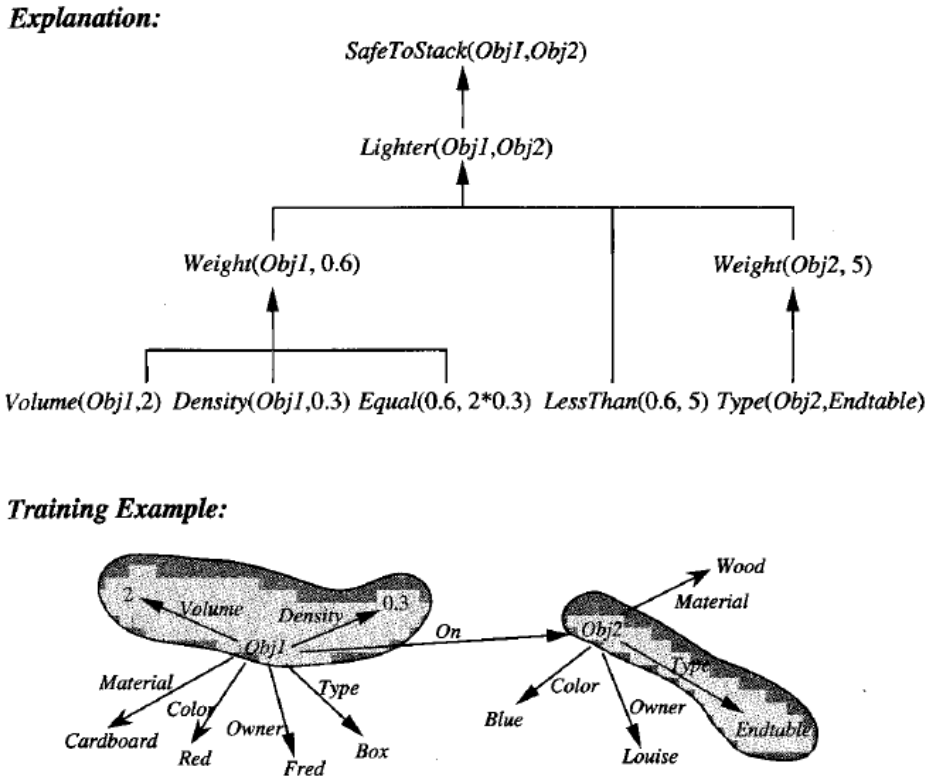
- مجموعه‌ی LearnedRules را خروجی بده.

جدول ۱۱,۲ الگوریتم Prolog-EBG.

در این الگوریتم برای هر نمونه‌ی مثبت که هنوز با دسته قوانین یاد گرفته شده (LearnedRules) سازگار نیستند، یک horn clause جدید ایجاد می‌شود. این horn clause جدید در سه مرحله ایجاد می‌شود. (۱) توضیح نمونه با استفاده از تئوری قلمرو (۲) بررسی توضیحات تا معلوم گردد کدام ویژگی‌های نمونه در مفهوم هدف دخیل‌اند و (۳) ساخت یک horn clause جدید که در این‌گونه نمونه‌ها با تابع هدف سازگار باشد.

توضیحات مربوط به نمونه‌ی ارائه شده در شکل ۱۱,۲ آمده است، توجه داشته باشید که شکل نشان داده شده در وسط نمود تصویری همان نمونه‌ای است که در جدول ۱۱,۱ آمده بود. بالای شکل توضیحات ساخته شده برای این نمونه را نشان می‌دهد. توجه داشته باشید که این توضیحات (یا همان اثبات مثبت بودن نمونه) برای گذاشتن Obj1 بر روی Obj2 آمده زیرا که Obj1 سبک‌تر از Obj2 است. این سبک‌تری Obj1 از Obj2 از وزن Obj1، که از چگالی (Density) و حجم (Volume) آن به دست آمده و وزن Obj2 که از وزن پیش‌فرض Endtable به دست آمده، ناشی شده است. Horn clause ی که از این توضیحات به دست می‌آید در جدول ۱۱,۱ آمده است. توجه داشته باشید که توضیح فقط از تعدادی از ویژگی‌های Obj1 و Obj2 استفاده کرده (ویژگی‌هایی که در شکل هاشور زده شده‌اند.)

با وجود اینکه در مثال ما فقط یک توضیح برای نمونه ممکن بود اما در کل ممکن است که از یک تئوری قلمرو چندین توضیح برای یک نمونه وجود داشته باشد. در چنین شرایطی چند تا یا کل توضیحات مورد استفاده قرار می‌گیرند. با وجود اینکه این توضیحات ممکن است horn clause را به تعمیم‌های مختلفی بکشانند اما با این حال همه در جهت تئوری قلمروند. در Prolog-EBG، مثل Prolog توضیحات با یک جستجوی زنجیروار معکوس[۶] انجام می‌شود. Prolog-EBG نیز مثل Prolog در اولین توضیح پیدا شده متوقف می‌شود و جستجو را ادامه نمی‌دهد.



شکل ۱۱,۲ توضیح یک نمونه‌ی آموزشی.
شبکه‌ی پایین شکل توضیح نمونه‌ی آموزشی جدول ۱۱,۱ را برای مفهوم SafeToStack(Obj1,Obj2) ارائه می‌دهد. قسمت بالایی شکل چگونگی راضی شدن مفهوم SafeToStack توسط این نمونه‌ی آموزشی را نشان می‌دهد. قسمت‌های هاشور خورده در شکل ویژگی‌های استفاده شده در توضیح را مشخص می‌کنند. ویژگی‌های دیگر تأثیری بر تعمیم فرضیه در مرحله‌ی بررسی ندارند.

۱۱,۲,۱,۲ بررسی توضیحات

نکته‌ی کلیدی در تعمیم نمونه‌های آموزشی جواب سؤال "از تمامی ویژگی‌های نمونه‌های مثبت کدام ویژگی‌ها ویژگی‌های تأثیر گذارند؟" است. توضیح ساخته‌شده‌ی یادگیر جواب مناسبی به این سؤال می‌دهد: دقیقاً ویژگی‌هایی که در توضیح مورد استفاده قرار گرفته‌اند. برای مثال توضیحی که در شکل ۱۱,۲ آمده از چگالی Obj1 (density) استفاده کرده در حالی که از صاحب (owner) آن استفاده‌ای نکرده. بنابراین فرضیه‌ای که برای SafeToStack(x,y) ارائه می‌شود باید از ویژگی Density(x,0.3) استفاده کند و کاری با ویژگی

[۶] backward chaining search

Owner(x,Fred) نداشته باشد. با جمع‌آوری اطلاعات مشخص شده در برگ‌های درخت شکل ۱۱,۲ و جایگزین کردن x و y به جای Obj1 و Obj2 می‌توان قانون کلی زیر را از تئوری قلمرو به دست آورد:

SafeToStack(x, y) ←Volume(x, ۲) ∧ density(y, 0.3) ∧ Type(y,EndTable)

قانون بالا تمامی موارد ذکر شده در برگ‌های شکل ۱۱,۲ را جز دو عبارت "Equal(0.6,times(2,0.3))" و "LessThan(0.6,5)" را در خود دارد. این دو برگ حذف شده‌اند زیرا که جدا از ویژگی‌های دو جسم x وy همیشه مقدار درست دارند.

با داشتن این قانون، برنامه می‌تواند توجیه[۷] متناسب را هم پیدا کند: توضیح نمونه‌ی آموزشی اثباتی برای درستی این قانون است. با وجود اینکه این توضیح برای پوشش نمونه‌ی آموزشی ساخته شده بود، اما توضیحی برای تمامی نمونه‌هایی که در آن صدق کنند نیز دارد.

قانون بالا تعمیمی قابل‌توجه روی نمونه‌ی آموزشی داده است، زیرا که بسیاری از ویژگی‌های دیگر نمونه را که مهم نبودند حذف کرده (ویژگی‌های مثل رنگ (color) دو شی). با این وجود با بررسی دقیق‌تر توضیحات، قوانین کلی‌تری را می‌توان به دست آورد. الگوریتم Prolog-EBG کلی‌ترین قانون را که توضیحات را توجیه کند به دست می‌آورد، این کار با استفاده از پیدا کردن ضعیف‌ترین پیش‌نویس[۸] توضیحات انجام می‌شود:

تعریف: ضعیف‌ترین پیش‌نویس یک نتیجه‌گیری مثل C با توجه به اثبات P، کلی‌ترین دسته فرض‌های اولیه‌ای مثل A است که بر اساس A با فرض درست بودن P بتوان C را نتیجه گرفت.

برای مثال مذکور ضعیف‌ترین پیش‌نویس مفهوم هدف SafeToStack(x,y)، با توجه به نمونه‌ی آموزشی جدول ۱۱,۱ به شکل قانون زیر بیان می‌شود. این کلی‌ترین قانونی است که می‌توان با توضیح ۱۱,۲ نوشت.

SafeToStack(x,y)←Volume(x,vx)∧ Density(x,dx)∧ Equal(wx,times(vx,dx))∧

LessThan(wx,5)∧ Type(y,EndTable)

توجه دارید که این قانون کلی دو مقدار حجم (volume) و (density) که در قانون اول آمده بود را نیاز ندارد. و به جای آن شرایطی کلی‌تر برای مقادیر این ویژگی‌ها می‌گذارد.

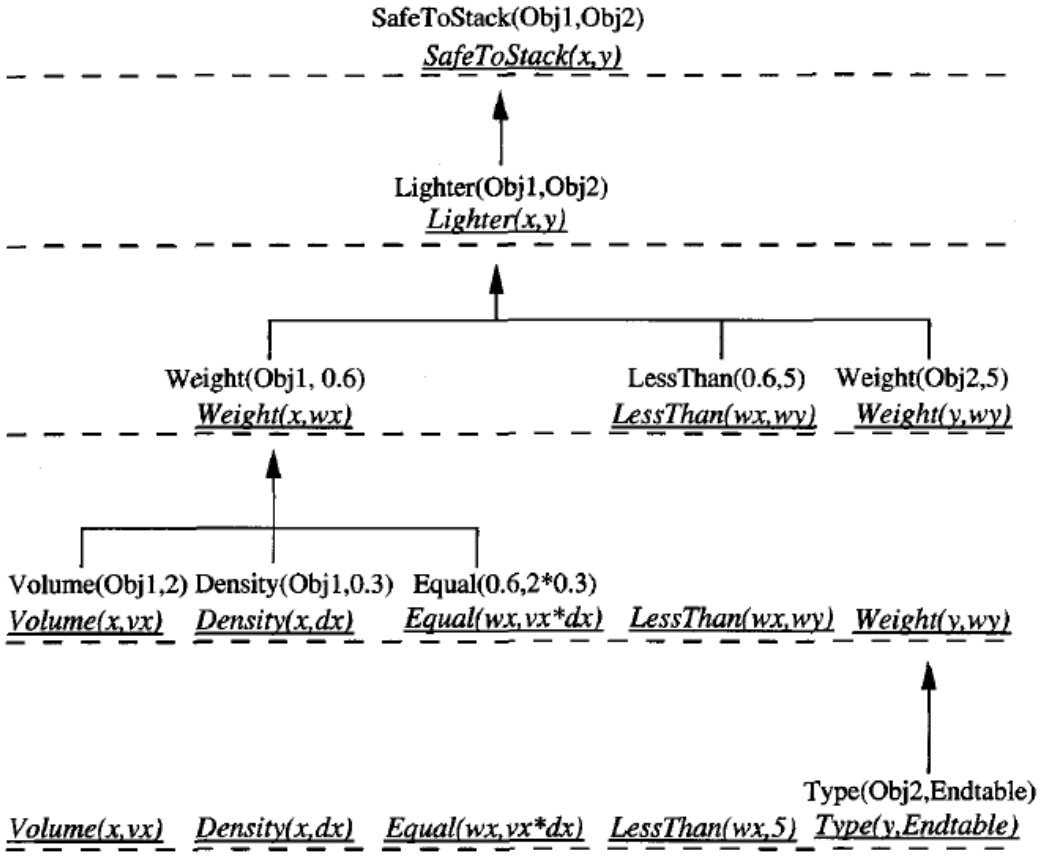
الگوریتم Prolog-EBG ضعیف‌ترین پیش‌فرض تابع مفهوم را بر اساس توضیحات و فرایندی به نام (Waldinger regression 1977) محاسبه می‌کند. فرایند regression بر روی تئوری قلمروهایی که بر اساس horn clause ها تعریف شده‌اند عمل می‌کند. این فرایند پله به پله توضیحات را بر عکس مورد بررسی قرار می‌دهد، ابتدا ضعیف‌ترین پیش‌فرض مفهوم هدف را بر اساس مرحله‌ی نهایی اثبات توضیحات محاسبه می‌کند، سپس ضعیف‌ترین پیش‌فرض عبارات حاصل را با توجه به قدم قبلی محاسبه می‌کند، و به همین ترتیب ادامه می‌دهد. این فرایند زمانی پایان می‌یابد که تمامی قسمت‌های توضیح بررسی شده باشند، یعنی ضعیف‌ترین پیش‌فرض مفهوم هدف با توجه به تمامی برگ‌های توضیح به دست آمده باشد.

---

[۷] justification

[۸] weakest preimage

مراحل طی شده‌ی فرایند regression در شکل ۱۱,۳ آمده است، در این شکل قسمت توضیح شکل ۱۱,۲ با خط غیرکج[۹] دوباره آورده شده است. مرز تعیین شده در هر مرحله‌ی فرایند regression با خط کج و زیرخط دار[۱۰] نوشته شده است. فرایند از ریشه‌ی درخت شروع به کار می‌کند و مرزی را برای مفهوم هدف کلی SafeToStack(x,y) بیان می‌کند. در مرحله‌ی اول، ضعیف‌ترین پیش‌فرض ممکن با توجه به عبارات و قسمت انتهایی درخت محاسبه می‌شود. در این مرحله قانون به دست آمده SafeToStack(x,y) ← Lighter(x,y) خواهد بود، پس ضعیف‌ترین پیش‌فرض ممکن Lighter(x,y) است. حال فرایند با مفهوم جدید {Lighter(x,y)} ادامه پیدا می‌کند. در مرحله‌ی بعدی و بررسی clause horn بعدی توضیح با فرایند به ضعیف‌ترین پیش‌نویس {Weight(x,wx),LessThan(wx,wy),Weight(y,wy)} می‌رسیم، یعنی تمامی x و y هایی که در آن‌ها wx (وزن x) از wy (وزن y) کمتر است. این فرایند به همین صورت پله به پله ادامه پیدا می‌کند تا بالاخره به دسته قوانینی که در برگ‌های درخت نوشته شده‌اند می‌رسد. مجموعه‌ی این قوانین که در انتهای شکل ۱۱,۳ نیز آمده قانون معرف ضعیف‌ترین پیش‌فرض خواهد بود.



شکل ۱۱,۳ محاسبه‌ی ضعیف‌ترین پیش‌فرض برای مفهوم SafeToStack(Obj1,Obj2) با توجه به توضیح موجود.
مفهوم هدف از ریشه (نتیجه) تا برگ‌های درخت بررسی می‌شود. در هر مرحله (که با خط‌چین جدا شده) دسته قوانین مرزی (با خط کج و زیرخط دار) یک مرحله در توضیحات عقب می‌رود. زمانی که فرایند پایان می‌پذیرد، عطف شرایط به دست آمده ضعیف‌ترین پیش‌فرض موجود برای توضیح و مفهوم هدف موجود خواهد بود. ضعیف‌ترین پیش‌فرض به دست آمده در انتها با خط کج و زیرخط دار نشان داده شده است.

---

[۹] nonitalic

[۱۰] italic underlined

قلب فرایند regression الگوریتم دیگری به نام regress است که در هر مرحلـه مرزهـا[11] فعلـی را بـا اسـتفاده از Horn clause ی از تئوری قلمرو پس می‌راند[12]. این الگوریتم در جدول ۱۱,۳ آورده شده است. مثال آمده در این جـدول مرحلـه‌ی اول فراینـد regression را در شکل ۱۱,۳ نشان می‌دهد. همان‌طور که در جدول هم آورده شده اسـت، الگـوریتم regress بـا پیـدا کـردن جانشـین ای بـرای سـر horn clause که با عبارت مرز یکی است و جایگزینی مرز با بدنه‌ی قانون، آن را پیش می‌برد.

قانون خروجی الگوریتم Prolog-EBG با فرمول زیر بیان می‌شود: بدنه‌ی قانون ضعیف‌ترین پیش‌فرض پیدا شده از طریق روش بالا خواهد بود و سر قانون نیز همان مفهوم هدف است که برای هر مرحله‌ی فرایند regression جانشینی‌ای (مثلاً جانشینی Θ_{hl} در جـدول ۱۱,۳) بـه آن اعمال شده است. این جانشینی از این جهت ضروری است که ثبات اسامی متغیرها در سر و بدنه‌ی حکم ایجاد شده را حفظ کرده و سر قانون را در مواقعی که توضیحات استفاده شده فقط یک حالت خاص از تابع هدف‌اند مشخص کند. همان‌طور که پیش‌تر نیز اشاره شد، برای مثال حاضـر قانون نهایی به شکل زیر خواهد بود،

SafeToStack(x,y)← Volume(x,vx)∧Density(x,dx)∧

Equal(wx,times(vx,dx))∧LessThan(wx,5)∧Type(y,Endtable)

Regress(Frontier,Rule,Literal, Θ_{hi})

Frontier: دسته‌ای از قوانین که باید با Rule، پسروی کنند (regress)

Rule: یک horn clause

Literal: عبارتی در Frontier که توسط Rule در توضیحات استنتاج می‌شود

Θ_{hi}: جانشینی‌ای که سر قانون Rule را با توجه به عبارت توضیحات یکتا می‌کند

این فرایند مجموعه‌ای از عبارات را که با هم ضعیف‌ترین پیش‌فرض Frontier را با توجه به Rule تشکیل می‌دهند را بر می‌گرداند

head→ سر Rule

body→ بدنه‌ی Rule

Θ_{hi}→ کلی‌ترین یکتا کننده‌ی head با Literal که جانشینی‌ای چون Θ_{hl} وجود داشته باشد که

$$\Theta_{hl}\left(\Theta_{hi}(head)\right) = \Theta_{hi}(head)$$

$\Theta_{hi}(Frontier - head + body)$ را برگردان

مثال (اولین مرحله‌ی regression در شکل ۱۱,۳)

---

[11] Frontier

[12] regress

Frontier = {Volume(x, us), Density(x, dx), Equal(wx, times(vx,dx)), LessThan(wx, wy),Weight(y, wy))

Rule = Weight(z,5) ← Type(z, Endtable)

Literal = Weight(y, wy)

Θ_{hi} = {z/Obj2}

head ← Weight (z, 5)

body ← Type(z, Endtable)

Θ_{hl} ← {z/y, wy/5}, where Θ_{li}= (y/Obj2)

Return {Volume(x, us), Density(x, dx), Equal (wx, times(vx, dx)), LessThan(wx, 5),

Type(y, Endtable)}

جدول ۱۱,۳ الگوریتم **regression** *برای مجموعه‌ای از عبارات با یک* **Horn clause**.
مجموعه‌ی عبارات **Frontier** *با* **Rule** *پس رانده می‌شوند.* **Literal** *عضوی از* **Frontier** *است که توسط* **Rule** *در توضیحات توجیه شده است. جانشینی* $\boldsymbol{\Theta_{hi}}$ *جانشینی از متغیرهاست که سر* **Rule** *را به عبارت مربوطه‌ی آمده در توضیحات مربوط می‌کند. الگوریتم ابتدا جانشینی* $\boldsymbol{\Theta_{hi}}$ *را که سر* **Rule** *و* **Literal** *را به صورتی که با جانشینی* $\boldsymbol{\Theta_{hl}}$ *سازگار باشد یکتا می‌کند محاسبه می‌کند. سپس جانشینی* $\boldsymbol{\Theta_{hi}}$ *را با توجه به* **Rule** *برای ایجاد پیش‌فرض* **Frontier** *اعمال می‌کند. علامت "+" و "−" در الگوریتم نشان‌دهنده‌ی اجتماع و اختلاف مجموعه‌ها هستند. نمادگذاری* **{z/y}** *نیز نشان‌دهنده‌ی جایگزینی* **y** *به جای* **z** *است. یک نمونه از چگونگی عملکرد الگوریتم آورده شده.*

۱۱,۲,۱,۳ تجدیدنظر در فرضیه‌های موجود

فرضیه‌ی فعلی در هر مرحله مجموعه‌ای از horn clause هاست که تا آن مرحله یاد گرفته شده‌اند. در هر مرحله، الگوریتم ترتیبی نمونه‌ی مثبت جدیدی را که هنوز پوشانده نشده انتخاب کرده و آن را توضیح می‌دهد و قانونی جدید بر اساس فرایندی که در بالا توضیح داده شد ایجاد می‌کند. توجه داشته باشید که طبق تعریفی که کردیم در این الگوریتم فقط نمونه‌های مثبت پوشانده خواهند شد و مجموعه‌ی horn clause های پوشانده شده فقط نمونه‌های مثبت را پیش‌بینی می‌کنند. اگر یک نمونه توسط قوانین فعلی یاد گرفته شده پیش‌بینی نشود آن نمونه منفی دسته‌بندی خواهد شد. این مطابق با روش منفی در زمان شکست (negation-as-failure) است که در سیستم Prolog توضیح داده شد.

۱۱,۳ نکاتی در مورد یادگیری توضیحی

همان‌طور که در بالا دیدیم، Prolog-EBG بررسی‌ای دقیق از تک نمونه‌های آموزشی برای تعیین بهترین راه تعمیم آن به یک فرضیه horn clause انجام می‌دهد. در زیر خواص کلیدی این الگوریتم آورده شده است،

بر خلاف روش‌های استقرایی، Prolog-EBG فرضیه‌های کلی توجیه شده توسط دانش قبلی را برای بررسی تک نمونه‌ها ارائه می‌دهد.

توضیح اینکه چگونه یک نمونه مفهوم هدف را راضی می‌کند مشخص می‌کند که کدام ویژگی‌های نمونه مربوط و کدام ویژگی‌ها نامربوط به تابع هدف‌اند: ویژگی‌های استفاده شده در توضیح ویژگی‌های مربوط هستند.

با بررسی بیشتر توضیح، پس راندن مفهوم هدف برای تعیین ضعیف‌ترین پیش‌فرض با توجه به توضیحات به ما اجازه می‌دهد تا قیدهای (constraint) کلی دیگری درباره‌ی ویژگی‌های مربوط پیدا کنیم.

هر horn clause یاد گرفته شده متناسب با شرط کافی برای راضی کردن تابع هدف است. مجموعه‌ی horn clause های یاد گرفته شده نمونه‌های آموزشی مثبتی را می‌پوشانند که یادگیر در طول یادگیری با آن‌ها مواجه شده است، اما دیگر نمونه‌هایی که این شرایط کافی را داشته باشند نیز پوشانده خواهند شد.

میزان تعمیم horn clause های یاد گرفته شده به فرمول تئوری قلمرو و ترتیب مشاهده‌ی نمونه‌های آموزشی وابسته است.

Prolog-EBG کاملاً فرض می‌کند که تئوری قلمرو درست و کامل است. اگر تئوری قلمرو درست یا کامل نباشد، مفهوم یاد گرفته شده نیز نادرست خواهد بود.

جنبه‌های مربوطه‌ی بسیاری در یادگیری توضیحی وجود دارد که به درک قابلیت‌ها و محدودیت‌های آن کمک می‌کند:

EBL به عنوان تعمیم نمونه‌ها با هدایت تئوری. EBL از تئوری قلمروش برای تعمیم نسبی نمونه‌ها با مشخص کردن ویژگی‌های مربوط و نامربوط نمونه‌ها استفاده می‌کند. با این روش این الگوریتم از مرزهای پیچیدگی نمونه‌ای که کاملاً در یادگیری استقرایی ایجاد می‌شود اجتناب می‌کند. این جنبه‌ای است که به طور ضمنی در توضیحات بالا درباره‌ی الگوریتم Prolog-EBG آمده است.

EBL به عنوان بازنویسی تئوری‌ها بر اساس نمونه‌ها. به الگوریتم Prolog-EBG می‌توان به صورت متدی برای بازنویسی تئوری قلمرو به فرمی کاربردی‌تر نگاه کرد. در کل، تئوری قلمرو اصلی با ایجاد قوانینی که (a) که از تئوری قلمرو نتیجه‌گیری خواهد شد و (b) نمونه‌های آموزشی مشاهده شده را تنها در یک مرحله استنباطی دسته‌بندی می‌کند، بازنویسی می‌شود. بنابراین، قوانین یاد گرفته شده را می‌توان بازنویسی تئوری قلمرو به صورت دسته‌ای از قوانین حالت خاص که می‌توانند نمونه‌های مفهوم هدف با یک استنتاج دسته‌بندی کنند دانست.

EBL به "فقط" عنوان بازنویس آنچه یادگیر "می‌داند". از نظری، یادگیر مثال SafeToStack با دانشی کامل از مفهوم SafeToStack شروع می‌کند. بدین معنا که، اگر تئوری قلمروی اولیه برای توضیح تمامی نمونه‌های آموزشی کافی است، پس برای پیش‌بینی دسته‌بندی‌ها نیز کافی خواهد بود. بنابراین، پس از چه نظر، این یادگیر ویژگی یادگیری دارد؟ یکی از جواب‌ها این است که تفاوت بین اینکه چیزی را به طور کلی بدانیم و اینکه چیزی را به طور بهینه محاسبه کنیم ممکن است بسیار زیاد باشد، و در چنین شرایطی این نوع "بازنویسی دانش" می‌تواند فرم مهمی از یادگیری باشد. برای مثال، در بازی شطرنج، قوانین بازی تئوری قلمروی‌ای کامل هستند، که در کل برای بازی بهینه‌ی شطرنج کافی‌اند. با این وجود، افراد برای یادگیری خوب بازی کردن نیاز به تجربه دارند. این دقیقاً وضعیتی است که تئوری قلمروی‌ای همه‌جانبه و کامل برای یادگیر (بازیکن) معلوم است و یادگیری فقط بازنویسی این دانش به فرمی است که بتوان آن را به طور کارآمد برای تعیین حرکت مناسب به کار برد. یک مسیر در حال ایجاد در فیزیک نیوتونی همین خاصیت را نشان می‌دهد، قوانین ساده‌ی فیزیک به راحتی نوشته می‌شوند اما دانشجویان قسمت عمده‌ای از ترم را بر روی نتایج آن کار می‌کنند در حالی که آن‌ها این دانش را در فرم کاربردی‌تر دارند و نیازی به بدست آوردن راه‌حل مسئله از قوانین اولیه در امتحان نهایی نخواهند داشت. Prolog-EBG این بازنویسی دانش را با نگاشت مستقیم دسته قوانین یاد گرفته‌اش روی ویژگی‌های نمونه‌های مشاهده شده و دسته‌بندی متناسب با مفهوم هدف به طوری که سازگار با تئوری قلمروی مربوطه باشد

انجام می‌دهد---. از آنجایی که دسته‌بندی نمونه‌ی دلخواه به مراحل متعددی استنتاج و جستجوی قابل‌توجهی در تئوری قلمرو دارد، قوانین یاد گرفته شده نمونه‌های مشاهده شده را در یک مرحله استنتاج دسته‌بندی می‌کنند.

بنابراین، فرم خالص EBL، بازنویسی تئوری قلمرو به صورت دسته قوانینی که نمونه‌های آموزشی را در یک استنباط دسته‌بندی می‌کنند است. این نوع بازنویسی دانش گاهی گرداوری دانش[۱] نامیده می‌شود، بدین معنا که این تبدیل کارایی دانش را Same درستی دانش افزایش می‌دهد.

می‌دهد.

۱۱,۳,۱ پیدا کردن خواص جدید

یکی از قابلیت‌های جالب Prolog-EBG توانایی آن در فرمولی کردن خواص جدیدی که به صورت صریح در توضیح نمونه‌های آموزشی نیامده‌اند اما برای توصیف قانون کلی نمونه‌های آموزشی لازم‌اند است. این قابلیت در بررسی عملکرد الگوریتم و قانون یاد گرفته شده‌ی قسمت قبل نشان داده شده است. در اینجا، قانون یاد گرفته شده نشان می‌دهد که قید روی Volume و Density ی x این است که ضربشان باید کمتر از ۵ باشد. در واقع، نمونه‌های آموزشی توصیفی از چنین حاصل‌ضربی یا مقداری که باید داشته باشد ندارند، در مقابل این قید توسط یادگیر به صورت اتوماتیک ایجاد می‌شود.

توجه دارید که این "ویژگی"[۲] مشابه ویژگی‌های واحدهای لایه پنهان شبکه‌های عصبی است؛ از این جهت که، این ویژگی یکی از خواص بسیار بسیار زیاد قابل‌محاسبه روی ویژگی‌های نمونه‌هاست. مشابه الگوریتم backpropation، Prolog-EBG نیز به طور خودکار چنین ویژگی‌هایی را حین تلاش برای تناسب با داده‌های آموزشی پیدا می‌کند. با این وجود، برخلاف فرایند آماری که ویژگی‌های واحدهای پنهان شبکه‌های عصبی را پیدا می‌کرد، Prolog-EBG از فرایندی تحلیلی برای ایجاد ویژگی جدید بر اساس تحلیل نمونه‌ی آموزشی استفاده می‌کند. در بالا، Prolog-EBG فرم تحلیلی ویژگی Density·Volume>5 را از نمونه‌سازی خاص تئوری قلمرو برای توضیح یک تک نمونه‌ی آموزشی ایجاد کرد. برای مثال، نمادگذاری حاصل‌ضرب Density و Volume از آن جهت اهمیت دارد که از یکی از قوانین تئوری قلمرو که Weight را تعریف می‌کند ایجاد می‌شود. این تفکر که این حاصل‌ضرب باید کمتر از ۵ باشد از دو قانون تئوری قلمرو که ادعا می‌کنند Obj1 باید Lighter و Endtable باشد و اینکه وزن Endtable (Weight)، ۵ است ناشی می‌شود. بنابراین، این ترکیب خاص و نمونه‌برداری و عبارات اولیه‌ی تئوری قلمرو است که به تعریف این خواص جدید کمک می‌کند.

مبحث یادگیری خودکار خواص مفید برای تغییر نمایش روی نمونه‌ها مبحث مهمی در یادگیری ماشین است. مشتقل تحلیلی این خواص جدید در یادگیری توضیحی و مشتقل استقرایی این خواص در لایه ی پنهان شبکه‌های عصبی دو روش مجزا را ارائه می‌کنند. زیرا که آنها به منابع مختلف اطلاعات (نظم های آماری روی نمونه‌های زیاد در مقابل تحلیل یک نمونه با استفاده از تئوری قلمرو) بررسی روشهایی که از ترکیب این دو منبع استفاده می‌کنند مفید خواهد بود.

---

[۱] knowledge compilation

[۲] feature

۱۱,۳,۲ یادگیری استنتاجی

در فرم خالص، Prolog-EBG بیشتر شبیه روشی استنتاجی است تا روشی استقرایی. بدین معنا کـه بـا محاسـبه‌ی ضـعیف‌ترین پـیش‌فـرض توضیحات فرضیه مثل h ایجاد می‌شود که از تئوری قلمروی B که نمونه‌های آموزشی D را پوشش می‌دهد نتیجه‌گیری خواهد شد. بـه عبـارت دقیق‌تر، Prolog-EBG فرضیه‌ای مثل h را خروجی خواهد داد که در دو شرط زیر صدق می‌کند:

$$(\forall < x_i, f(x_i) > \in D)(h \wedge x_i) \vdash f(x_i) \qquad (11.1)$$

$$D \wedge B \vdash h \qquad (11.2)$$

در این روابط داده‌های آموزشی D شامل مجموعه‌ای از نمونه‌های آموزشی است که در آن x_i، i امین نمونه‌ی آموزشی و $f(x_i)$ مقدار هـدف آن است (f تابع هدف است). توجه داشته باشید که شرط اول فقط تعبیر ریاضی شرط کلی یادگیری ماشین است، اینکه فرضیه‌ی h بایـد مقـدار هدف $f(x_i)$ را برای هر نمونه‌ی x_i در D به درستی پیش‌بینی کند (البته در اینجا ---). البته در کل، فرضـیه‌های جـایگزین بسـیاری وجـود دارند که در شرط اول صدق می‌کنند. شرط دوم تأثیر تئوری قلمرو در Prolog-EBG را نشان می‌دهد: فرضیه‌ی خروجی مجبور است از تئوری قلمرو و داده‌های آموزشی نتیجه گرفته شده باشد. این شرط دوم ابهام یادگیر در انتخاب فرضیه را کاهش می‌دهد. بنابراین، تأثیر تئـوری قلمـرو کاهش مؤثر اندازه‌ی فضای فرضیه و بنابراین کاهش پیچیدگی نمونه‌ها در یادگیری است.

با نمایشی دیگر می‌توان نوع دانشی که Prolog-EBG به عنوان تئوری قلمرو نیاز دارد را معلوم کرد. در کل، Prolog-EBG فرض می‌کنـد که تئوری قلمروی B را می‌توان از دسته‌بندی نمونه‌های داده‌های آموزشی نتیجه گرفت:

$$(\forall < x_i, f(x_i) > \in D)(B \wedge x_i) \vdash f(x_i) \qquad (11.3)$$

این شرط برای تئوری قلمروی B معادل این است که فرض کنیم که B بتواند برای هر نمونه مثبت توضیحی ارائه دهد.

مقایسه‌ی تعریف مسئله‌ی یادگیری Prolog-EBG و برنامه‌نویسی استقرایی منطقی (ILP) که در فصل ۱۰ بحث شد جالـب اسـت. در فصـل ۱۰ فرض کردیم که تعمیمی از یادگیری استقرا معمول انجام می‌دهیم با این تفاوت که دانش قبلی‌ای مثل B′ در اختیار یادگیر است. (از B′ بـه جای B برای دانش قبلی ILP استفاده کردیم تا نشان دهیم که B′ در رابطه‌ی ۱۱,۳ صدق نمی‌کنـد). ILP یـک سیسـتم اسـتقرایی اسـت امـا Prolog-EBG یک سیستم استنتاجی است. ILP از دانش قبلی B′ برای افزایش اندازه‌ی مجموعه‌ی فرضیه‌های ممکـن اسـتفاده می‌کنـد در حالی که Prolog-EBG از دانش قبلی B برای کاهش مجموعه‌ی فرضیه‌های قابل‌قبول استفاده می‌کند. همان‌طور که در رابطه‌ی 10.2 نیـز آمد، ILP فرضیه‌ای مثل h خروجی می‌دهد که در شرط زیر صدق می‌کند،

$$(\forall < x_i, f(x_i) > \in D)(B' \wedge h \wedge x_i) \vdash f(x_i)$$

به رابطه‌ی بین این شرط و شروط h در Prolog-EBG (که در روابط ۱۱,۱ و ۱۱,۲ آمده) توجه کنید. این شرط ILP برای h حالت ضعیف‌تری از شرط رابطه‌ی ۱۱,۱ است، در ILP فقط لازم است که $(B' \wedge h \wedge x_i) \vdash f(x_i)$ در حالی که در Prolog-EBG شـرط بسـیار محکم‌تـر است، $(h \wedge x_i) \vdash f(x_i)$. همچنین توجه داشته باشید که ILP شرطی در مقابل شرط رابطه‌ی ۱۱,۲ Prolog-EBG ندارد.

۱۱,۳,۳ بایاس استقرایی یادگیری توضیحی

با توجه به آنچه در فصل ۲ گفته شد، بایاس استقرایی یک الگوریتم یادگیری مجموعه‌ای از فرض‌هاست که با آن به همراه نمونه‌های آموزشی می‌توان پیش‌بینی‌های یادگیر را نتیجه‌گیری کرد. اهمیت بایاس استقرایی در این است که چگونگی تعمیم یادگیر بر روی نمونه‌های آموزشی مشاهده شده را مشخص می‌کند.

اما بایاس استقرایی Prolog-EBG چیست؟ همان‌طور که در رابطه‌ی ۱۱,۲ نیز گفته شده، در Prolog-EBG فرضیه‌ی خروجی h را می‌توان از $B \wedge D$ نتیجه گرفت. بنابراین، تئوری قلمرو B مجموعه‌ای از فرض‌هاست که به همراه نمونه‌های آموزشی می‌توان از آن فرضیه‌ی خروجی را نتیجه‌گیری کرد. با دانستن اینکه پیش‌بینی‌های یادگیر فقط از این فرضیه‌ی h ناشی می‌شود، به نظر می‌رسد که بایاس استقرایی Prolog-EBG فقط تئوری قلمروی B است. در واقع این گفته است اما باید یک توضیح کوچک را در نظر گرفت که: مجموعه‌ی بزرگی از horn clause ها را می‌توان از تئوری قلمرو نتیجه‌گیری کرد. پس عناصر باقیمانده‌ی بایاس استقرایی ناشی از بایاس در انتخاب Prolog-EBG در میان horn clause های ممکن است. همان‌طور که در بالا دیده می‌شود، Prolog-EBG از الگوریتمی ترتیبی که تا اتمام تمامی نمونه‌های مثبت قانون تولید می‌کند استفاده می‌کند. علاوه بر آن، هر horn clause کلی‌ترین حکم ممکن (ضعیف‌ترین پیش‌فرض) بر اساس توضیحات نمونه‌ی آموزشی فعلی است. بنابراین، در بین این horn clause ها که از تئوری قلمرو نتیجه‌گیری می‌شوند، می‌توانیم بایاس استقرایی Prolog-EBG را ارجحیت مجموعه‌ی کوچکی از کلی‌ترین horn clause ها بدانیم. در واقع، الگوریتم حریص Prolog-EBG فقط یک تقریب برای الگوریتم جستجوی همه‌جانبه است که واقعاً کوتاه‌ترین مجموعه از کلی‌ترین horn clause ها را پیدا می‌کند. با این وجود، بایاس استقرایی Prolog-EBG را می‌توان تقریباً آنچه گفته شد در نظر گرفت.

تقریب بایاس استقرایی Prolog-EBG: تئوری قلمروی B به علاوه‌ی ارجحیت مجموعه‌های کوچک‌تر horn clause های کلی‌تر.

مهم‌ترین نکته در اینجا این است که بایاس استقرایی Prolog-EBG (خط‌مشی‌ای که برای تعمیم روی داده‌های آموزشی پیش می‌گیرد) به شدت به تئوری قلمروی ورودی وابسته است. این مخالف عملکرد تمامی دیگر الگوریتم‌های یادگیری (مثل شبکه‌های عصبی و درخت یادگیری) است، در دیگر الگوریتم‌های یادگیری بایاس استقرایی خاصیتی ثابت از الگوریتم یادگیری بود که معمولاً به نمایش فرضیه‌ها وابسته بود. چرا وابستگی بایاس استقرایی Prolog-EBG به ورودی و ثابت نبودن آن اهمیت دارد؟ زیرا که، همان‌طور که در فصل ۲ و دیگر فصول گفته شد، بایاسی وجود ندارد که در همه‌ی مسائل کارایی لازم را داشته باشد، همچنین یادگیری بدون بایاس بیهوده است. بنابراین، هر تلاش برای ساخت متد کلی یادگیری حداقل لازم است که بایاس استقرایی متغیری داشته باشد تا بتواند متناسب با مسئله تغییر کند. در سطح کاربردی‌تر، در بسیاری از کارها طبیعی است که دانشی مربوط به قلمرو مسئله (مثل، دانش درباره‌ی Weight در مسئله‌ی SafeToStack) که در چگونگی تعمیم یادگیر بر روی داده‌های آموزشی تأثیرگذار است در دسترس است. در مقابل، بایاس کردن فرم فرضیه‌ها (مثل ترجیح درخت‌های کوتاه‌تر در یادگیری درختی) طبیعی نیست. در آخر، اگر مشکل بزرگ چگونگی اینکه عامل خودکار[۱] توانایی‌هایش در یادگیری را در طول زمان بهبود می‌بخشد را در نظر بگیریم، بنابراین جذاب خواهد بود که الگوریتم یادگیری‌ای داشته باشیم که قابلیت‌های تعمیمش با جمع آوردی دانش بیشتر از قلمرو بهبود یابد.

---

[۱] autonomous

۱۱,۳,٤ یادگیری مرتبه‌ی دانش[۱]

همان‌طور که در رابطه‌ی ۱۱,۲ اشاره شد، فرضیه‌ی خروجی h از الگوریتم Prolog-EGB از تئوری قلمروی B و نمونه‌های آموزشی D نتیجه‌گیری می‌شود. در واقع، با بررسی الگوریتم Prolog-EBG فهمیدن اینکه h به طور مستقیم از B نتیجه‌گیری می‌شود و مستقل از D است بسیار ساده خواهد بود. یکی از راه‌های پی بردن به این حقیقت بررسی الگوریتم Lemma-Enumerator است. الگوریتم Lemma-Enumerator، خیلی ساده، تمامی درخت‌های اثبات[۲] شامل مفهوم هدف بر اساس فرض‌ها در تئوری قلمروی B را می‌شمارد. برای هر درخت اثبات، Lemma-Enumerator ضعیف‌ترین پیش‌فرض را محاسبه می‌کند و horn clause ی بر اساس آنچه در Prolog-EBG گفته شد تشکیل می‌دهد. تنها تفاوت Lemma-Enumerator با Prolog-EBG این است که Lemma-Enumerator کاری با داده‌های آموزشی ندارد و فقط تمامی درخت‌های اثبات را می‌شمارد.

توجه دارید که Lemma-Enumerator مجموعه توانی‌ای (superset) از horn clause ها را که خروجی Prolog-EBG هستند را خروجی می‌دهد. با دانستن این حقیقت سؤال‌های بسیاری مطرح می‌گردد. ابتدا اینکه، اگر فرضیه‌های خروجی Prolog-EBG مستقیماً و تنها از تئوری قلمرو نتیجه‌گیری می‌شوند، پس نقش داده‌های آموزشی در Prolog-EBG چیست؟ جواب در این است که نمونه‌های آموزشی الگوریتم Prolog-EBG را بر ایجاد قوانینی که توزیع نمونه‌ها در عمل را پوشش می‌دهد متمرکز می‌کند. برای مثال، در مثال شطرنج ما، مجموعه‌ی تمامی (lemma) ها بسیار بزرگ است، در حالی که مجموعه‌ی پوزیسیون‌هایی که در بازی واقعی شطرنج رخ می‌دهد فقط قسمت کوچکی از آن مجموعه‌ی بسیار بزرگ است. بنابراین، با تمرکز بر نمونه‌های آموزشی تولیدی در تمرین‌ها، برنامه با مجموعه‌ی کوچک‌تری از پوزیسیون‌ها سروکار خواهد داشت.

سؤال دوم مطرح این است که چگونه Prolog-EBG می‌تواند فرضیه‌ای را یاد بگیرد که به صورت ضمنی در تئوری قلمرو آورده شده است را یاد بگیرد. به عبارت دیگر، آیا این الگوریتم (با این فرض که الگوریتم در میزان محاسبات هیچ محدودیتی نداشته باشد) می‌تواند دسته‌بندی نمونه‌ای را که نمی‌توان به تنهایی با تئوری قلمرو دسته‌بندی کرد را یاد خواهد گرفت؟ متأسفانه، الگوریتم نمی‌تواند. اگر B⊢h، پس هر دسته‌بندی‌ای که از h نتیجه‌گیری خواهد شد را می‌توان نتیجه‌گیری‌ای مستقیم از B دانست. آیا این محدودیت در یادگیری توضیحی یا استنتاجی ذاتی است؟ خیر، این محدودیت ذاتی نیست، در زیر مثالی در این باره آورده شده است.

برای ایجاد نمونه‌ای از یادگیری استنتاجی که در آن فرضیه‌ی یاد گرفته شده‌ی h را مستقیماً نتوان از B نتیجه‌گیری کرد، باید حالتی ایجاد کنیم که $B \nvdash h$ اما داشته باشیم $B \wedge D \vdash h$ (که در رابطه‌ی ۱۱,۲ نیز آمده بود). یکی از حالات جالب حالتی است که B شامل فرض‌هایی چون فرض "اگر x تابع هدف را راضی کند، پس g(x) نیز تابع هدف را راضی خواهد کرد" است. این فرض به تنهایی دسته‌بندی هیچ‌یک از نمونه‌ها را مشخص نخواهد کرد، با این وجود، زمانی که یک نمونه‌ی مثبت مشاهده می‌شود، تعمیم استنتاجی روی نمونه‌های مشاهده نشده ممکن خواهد شد. برای مثال، یادگیری مفهوم هدف PlayTennis را در نظر بگیرید. فرض کنید که هر روز فقط با ویژگی Humidity توصیف شود و تئوری قلمرو فرض "اگر مفهوم بازی تنیس در Humidity x درست باشد، این مفهوم در Humidity های کمتر نیز درست خواهد بود" است. به فرم رسمی‌تر می‌توان این تئوری قلمرو را به صورت زیر نوشت،

---

[۱] knowledge level learning

[۲] proof tree

$$(\forall x)\ \text{IF}\ \big((\text{PlayTennis} = \text{Yes}) \leftarrow (\text{Humidity} = x)\big)$$

$$\text{THEN}\ ((\text{PlayTennis} = \text{Yes}) \leftarrow (\text{Humidity} \leq x))$$

توجه دارید که از روی این تئوری قلمرو نمی‌توان هیچ دسته‌بندی را در مورد مثبت یا منفی بودن یک نمونه‌ی PlayTennis را نتیجه گرفت. با این وجود، زمانی که یک یادگیر نمونه‌ی مثبتی با Humidity=.3 را مشاهده می‌کند، با توجه به تئوری قلمرو، فرضیه کلی زیر را نتیجه می‌گیرد،

$$(\text{PlayTennis} = \text{Yes}) \leftarrow (\text{Humidity} \leq .3)$$

به طور خلاصه اینکه، این مثال تصوری حالتی از $B \nvdash h$ را نشان می‌دهد که در آن داریم $B \wedge D \vdash h$. فرضیه یاد گرفته شده در این حالت پیش‌بینی‌هایی را می‌تواند نتیجه‌گیری کند که مستقیماً از دانش قبلی نتیجه‌گیری نمی‌شوند. برای این نوع یادگیری، یادگیری‌ای که در آن پیش‌بینی‌ها فقط به تئوری قلمرو وابسته نیستند، "یادگیری مرتبه دانش" می‌گویند. مجموعه‌ی تمامی پیش‌بینی‌های نتیجه‌گیری شده از فرض Y را بستگی استنتاجی[۱] Y می‌نامند. برتری کلیدی در اینجا این است که یادگیری سطح دانش بستگی استنتاجی B در اینجا زیرمجموعه‌ای از بستگی استنتاجی B+h است.

نمونه‌ی دوم یادگیری تحلیلی سطح دانش با نوعی از فرض که تعیین[۲] نامیده می‌شود ایجاد می‌شود. این نوع یادگیری سطح دانش با جزئیات در (Russel 1989) آمده است. تعیین، فرض می‌کند که یکی از ویژگی‌های نمونه کاملاً به دیگر ویژگی‌هایش وابسته است اما نوع وابستگی را مشخص نمی‌کند. برای مثال، یادگیری مفهوم هدف "افرادی که پرتقالی حرف می‌زنند" را در نظر بگیرید و همچنین فرض کنید که تئوری قلمرو فرض تعیین "زبانی که افراد به آن حرف می‌زنند وابسته به ملیت آن‌هاست" است. این تئوری قلمرو به تنهایی هیچ دسته‌بندی‌ای بر روی نمونه‌ها انجام نمی‌دهد، با این وجود، اگر با نمونه‌ی مثبت "جو، ۲۳ ساله، چپ‌دست، برزیلی، پرتقالی حرف می‌زند" برخورد کنیم، از روی این نمونه و تئوری قلمرو می‌توان گفت که "تمامی برزیلی‌ها پرتقالی حرف می‌زنند".

هر دو مثال آورده شده نشان می‌دهند که چگونه با روش استنتاجی می‌توان فرضیه‌ای را خروجی گرفت که فقط نتیجه‌گیری شده از تئوری قلمرو نباشد. در هر دو حالت فرضیه خروجی h در رابطه‌ی $B \wedge D \vdash h$ صدق کرده اما در $B \vdash h$ صدق نمی‌کند. در هر دو حالت یادگیر فرضیه‌ای توجیه شده که نمی‌توان آن را به تنهایی از تئوری قلمرو و یا داده‌های آموزشی به دست آورد استنتاج[۳] می‌کند.

۱۱,٤ یادگیری توضیحی ی دانش کنترل جستجو

همان‌طور که در بالا گفته شد، محدودیت کاربرد عملی الگوریتم Prolog-EBG نیاز آن به تئوری قلمروی درست و همه‌جانبه است. مجموعه‌ای از مسائل یادگیری که در آن‌ها این نیاز به سادگی بر طرف می‌شود مسائل یادگیری افزایش سرعت جستجوی برنامه‌های پیچیده تعیین[۴] است. در واقع، بزرگ‌ترین مسئله‌ای که در آن از یادگیری توضیحی استفاده می‌شود، مسئله‌ی یادگیری کنترل جستجو است، به این

---

[۱] deductive closure
[۲] determination
[۳] deducts
[۴] speed up complex search programs

مسئله گاهی مسئله‌ی افزایش سرعت[۱] نیز گفته می‌شود. برای مثال، بازی کردن بازی‌هایی چون شطرنج نیاز به جستجوی میان فضایی بزرگ از حرکات ممکن و پوزیسیون‌های مختلف برای پیدا کردن بهترین حرکت دارد. بسیاری از مسائل عملی برنامه‌ریزی و بهینه‌سازی[۲] را می‌توان به راحتی به صورت مسائل جستجو برای پیدا کردن حرکتی به سوی وضعیت هدف بیان کرد. در چنین مسائلی، تعریف عملگرهای جستجوی قانونی به همراه تعریف هدف جستجو تئوری قلمروی‌ای کامل و درست برای یادگیری کنترل جستجو ایجاد خواهد کرد.

اما دقیقاً چگونه باید مسئله‌ی یادگیری کنترل جستجو را بیان کنیم تا بتوان یادگیری توضیحی را در آن به کار برد؟ مسئله‌ای کلی از یادگیری کنترل جستجویی را در نظر بگیرید که در آن S مجموعه‌ی تمامی وضعیت‌های ممکن و O مجموعه‌ی تمامی عملگرهای ممکن برای تبدیل یک وضعیت به وضعیتی دیگر و G نیز عبارتی تعریف شده روی S است که مشخص می‌کند کدام وضعیت‌ها وضعیت هدف هستند. مسئله در کل پیدا کردن ترتیبی از عملگرهاست که وضعیت اولیه‌ی دلخواه s_i را به وضعیت آخر s_f ببرد که عبارت G را راضی می‌کند. یکی از راه‌های بیان این مسئله تغییر سیستم یادگیری برای یادگیری تابع هدفی مستقل برای هر یک از عملگرهای O است. در کل، برای هر عملگر مثل o در O ممکن است مفهوم هدفی به شکل "مجموعه‌ی وضعیت‌هایی که o به یک وضعیت هدف ختم می‌شود" تعریف شود. البته برای انتخاب دقیق اینکه کدام مفهوم هدف را یاد بگیریم به ساختار داخلی حلال مسئله[۳] بستگی دارد که از دانش یاد گرفته شده استفاده خواهد کرد. برای مثال، اگر حلال مسئله سیستمی با هدف‌های میانی[۴] است که مسئله را با ایجاد و حل زیر هدف‌ها[۵] حل می‌کند، ممکن است بخواهیم در عوض مفهوم "مجموعه وضعیت‌هایی که باید در آن‌ها زیر هدف A زودتر از زیر هدف B عملی شود" را یاد بگیریم.

یکی از سیستم‌هایی که از یادگیری توضیحی برای بهبود جستجویش استفاده می‌کند Prodigy (Carbonell 1990) است. Prodigy سیستم تصمیم‌گیری‌ای[۶] مستقل از قلمرو[۷] است که تعریف قلمروی مسئله را با وضعیت‌های S و عملگرهای O دریافت می‌کند. سپس مسئله را در فرم "ترتیبی از عملگرها بیابید که وضعیت اولیه‌ی s_i را به وضعیتی که در G صدق می‌کند برساند" حل می‌کند. ***Prodigy*** از برنامه‌ریز[۸] اهداف میانی که مسئله را به زیر هدف‌های تجزیه می‌کند و آن‌ها را حل می‌کند کمک می‌گیرد و در آخر نیز این روش‌های حل را ترکیب کرده و راه‌حلی برای کل مسئله می‌یابد. بنابراین، در طی این جستجو برای حل مسئله ***Prodigy*** مرتباً با سؤالاتی نظیر "به کدام زیر هدف را باید در قدم بعدی رسید؟" و "کدام عملگر برای حل این زیر هدف باید در نظر گرفته شود؟" مواجه است. ***(Mintor 1988)*** مجتمع سازی یادگیری توضیحی در ***Prodigy*** را با تعریف مجموعه‌ای از مفاهیم هدف متناسب با این نوع کنترل تصمیم که مرتباً با آن مواجهیم را توضیح می‌دهد. برای مثال، یکی از مفاهیم هدف "مجموعه‌ی وضعیت‌هایی است که زیر هدف ***A*** باید قبل از زیر هدف ***B*** حل شود". مثالی از قانون یاد گرفته شده توسط ***Prodigy*** برای این مفهوم هدف در قلمروی مثال ساده‌ی روی هم گذاشتن مکعب‌ها به صورت زیر است،

IF One subgoal to be solved is On(x,y) and

One subgoad to be solved is On(y,z)

---

[۱] speedup
[۲] scheduling and optimization
[۳] problem solver
[۴] means-ends
[۵] subgoal
[۶] planning
[۷] domain-independent
[۸] *planner*

THEN Solve the subgoal On(y,z) before On(x,y)

برای درک این قانون دوباره مثال روی هم گذاشتن مکعب‌ها را که در شکل ۹،۳ توضیح داده شده بود را در نظر بگیرید. در این مثال، هدف چیدن مکعب‌ها به صورتی است که کلمه‌ی "universal" را نمایش دهد. Prodigy این هدف را به زیر هدف‌هایی تقسیم می‌کند، زیر هدف‌هایی نظیر On(U,N) و On(N,I) و ... . توجه دارید که قانون بالا بدین معناست که زیر هدف On(U,N) باید قبل از زیر هدف On(N,I) حل شود. توجیه این قانون (و توضیح آن توسط Prodigy برای یادگیری این قانون) این است که اگر زیر هدف‌ها را در ترتیب عکس حل کنیم، به تضاد خواهیم رسید و باید حل On(U,N) را برای رسیدن به هدف On(N,I) ضایع کنیم. Prodigy در ابتدا با چنین تضادی برخورد می‌کند سپس این تضاد را توضیح می‌دهد و قانونی مثل بالا برای آن ایجاد می‌کند. اثر برآیند این است که Prodigy از دانش مستقل از قلمروش برای تشخیص زیر هدف‌های ناسازگار و از دانش مخصوص قلمروی‌اش از عملگرهای قلمرو (برای مثال، این حقیقت که ربات در هر بار فقط یک جعبه را می‌تواند بلند کند) برای یادگیری قوانین لازمه در این قلمرو برای حل مشابه بالا استفاده می‌کند---.

استفاده از یادگیری توضیحی برای به دست آوردن دانش کنترل برای Prodigy در بسیاری از قلمروهای مسائل شامل مسئله‌ی چیدن مکعب‌های بالا و مسائل پیچیده‌تر برنامه‌ریزی و اجرا موفق ظاهر شده‌اند. (Minton 1988) آزمایش‌هایی در سه قلمروی مسئله، که قوانین یاد گرفته شده به حل مسئله با ضریب ۲ تا ۴ بهبود بخشیده‌اند را معرفی می‌کند. علاوه بر این، کارایی قوانین یاد گرفته شده نیز با قوانین دست‌نویس این تئوری‌های قلمرو قابل‌مقایسه بوده است. Minton همچنین تعدادی فرایند گسترش یافته برای یادگیری ساده‌ی توضیحی ارائه می‌کند که کارایی یادگیری کنترل جستجو را بهبود می‌بخشد. این فرایندها شامل متدهای ساده‌سازی قوانین یاد گرفته شده و حذف قوانین یادگیری‌ای که سودشان کمتر از هزینه‌شان است می‌شود.

مثال دوم ساختار حل مسئله‌ی کلی که از یادگیری توضیحی کمک می‌گیرد سیستم Soar (Laird et al. 1986; Newell 1990) است. Soar دامنه‌ی وسیعی از استراتژی‌های حل مسئله شامل روش هدف‌های میانی Prodigy را پشتیبانی می‌کند. مشابه Prodigy، با وجود اینکه، Soar با توضیح وضعیت‌ها یاد می‌گیرد---. زمانی که این سیستم با جستجویی که جوابی قطعی برایش ندارد مواجه می‌شود، (مثل این سؤال که از کدام عملکرد در گام بعدی استفاده کند)، سیستم --- با استفاده از متدهای ضعیف نظیر (generate-and-test) برای تعیین مسیر درست عملیات استفاده می‌کند. ---

Prodigy و Soar نشان می‌دهند که متدهای مبتنی بر یادگیری توضیحی را می‌توان برای یادگیری دانش کنترل جستجو در قلمروی مسائل مختلف به کاربرد. با این وجود، تعداد زیادی یا اکثر برنامه‌های جستجو همچنان از توابع ارزیابی عددی، مشابه آنچه در فصل ۱ توصیف شد، به جای قوانین به دست آمده از یادگیری توضیحی استفاده می‌کنند. اما دلیل چیست؟ در واقع مسائل کاربردی مهمی در استفاده از EBL برای یادگیری کنترل جستجو وجود دارد. اول اینکه در بسیاری از موارد تعداد قوانین کنترل به باید یاد گرفته شوند بسیار زیادند (مثلاً در حد هزاران قانون). همین‌طور که سیستم قوانین کنترل بیشتری را برای بهبود جستجویش یاد می‌گیرد باید هزینه‌ی بیشتر و بیشتری در هر مرحله برای جفت کردن[۱] این مجموعه از قوانین و وضعیت فعلی بپردازد. توجه دارید که این مشکل به یادگیری توضیحی منوط نمی‌شود؛ این مشکل برای تمامی سیستم‌هایی که دانش یادگرفتنشان را با دسته‌ای از قوانین که تعدادشان افزایش می‌یابد وجود دارد. الگوریتم‌های جفت سازی می‌توانند این مشکل را تا حدی حل کنند، اما این مشکل به طور کامل از بین نمی‌رود. (Minton 1988) استراتژی‌های تخمین تجربی این هزینه‌ی محاسباتی و سود هر قانون را بررسی کرد و قوانین را یادگیر می‌گیرد که میزان سود تخمینی از میزان هزینه‌ی تخمینی کمتر باشد و همچنین

---

[۱] match

قوانینی که اثر منفی دارند را حذف می‌کند. وی نحوه‌ی استفاده از این تحلیل تأثیر قوانین[۱] را برای تعیین تأثیرگذاری یادگیری توضیحی در Prodigy توصیف می‌کند. برای مثال، در سری‌ای از مسائل چیدن مکعب، Prodigy ۳۲۸ فرصت یادگیری قوانین جدید دارد اما فقط ۶۹ قانون از این قوانین را استخراج می‌کند و سرانجام این مجموعه را به ۱۹ قانون کاهش می‌دهد، و قوانین کم‌کاربرد را حذف می‌کند. (Tambe et al. 1990 و Doorenbos 1993) چگونگی تشخیص انواع قوانین که در کل جفت شدنشان هزینه‌بر خواهد بود، را به همراه متدهای بازنویسی چنین قوانینی در فرم‌های کاراتر و متدهایی برای الگوریتم‌های بهینه‌سازی جفت سازی ارائه می‌کنند. (Doorenbos 1993) چگونگی اینکه این متدها به Soar امکان جفت شدن با 100,000 قانون یاد گرفته شده بدون افزایش قابل‌توجه در هزینه جفت شدن بر حالت در قلمروی مسئله‌ای را می‌دهند را مورد بحث و بررسی قرار می‌دهد.

مشکل کاربردی دیگر یادگیری توضیحی در یادگیری کنترل جستجو این است که در بسیاری از موارد حتی ساختن توضیحات برای مفهوم هدف بسیار به طور رام نشدنی‌ای سخت است. برای مثال، در شطرنج ممکن است بخواهیم مفهوم هدفی چون "وضعیت‌هایی که عملگر A ما را به سمت راه‌حل بهینه می‌برد را یاد بگیریم". متأسفانه برای اثبات یا توضیح اینکه چرا A ما را به سمت راه‌حل بهینه می‌برد نیاز دارد که نشان دهیم تمامی دیگر عملگرها نتیجه‌ای ضعیف‌تر خواهند داشت. این کار معمولاً تلاشی نمایی در عمق جستجو را نیاز خواهد داشت. (Chien 1993 و Tadepalli 1990) متدهایی برای توضیحات "تنبل"[۲] یا افزایشی[۳] که در آن روشی ابتکاری برای ایجاد توضیحات جزئی و تخمینی اما قابل‌اجرا (tractable) را معرفی می‌کنند. قوانین از این توضیحات ناکامل مشابه زمانی که توضیحات کامل بودند استخراج می‌شود. البته این قوانین یاد گرفته شده ممکن است به خاطر توضیحات ناکامل نادرست باشند. سیستم این مشکل را با نظارت بر کارایی قانون بر روی وضعیت‌های بعدی اصلاح می‌کند. اگر قانونی در مشاهدات بعدی اشتباه کند، آنگاه توضیح اصلی به صورت توانی ایجاد خواهد شد تا وضعیت جدید را بپوشاند و قانونی بازنگری شده از این توضیح توانی استخراج خواهد شد.

تلاش‌های بسیار تحقیقاتی دیگری درباره‌ی کاربرد یادگیری توضیحی برای بهبود حلال‌های مسائل جستجویی انجام گرفته است (Mitchell 1981; Silver 1983; Shavlik 1990; Mahadevan et al. 1993; Gervasio and DeJong 1994; DeJong 1994. (Bennett and DeJong 1996) یادگیری توضیحی را برای مسائل برنامه‌ریزی ربات که سیستم تئوری قلمروی ناکاملی دارد که جهان واقعی و حرکات را توصیف می‌کند را بررسی کرده‌اند. (Dietterich and Flann 1995) اجتماع یادگیری توضیحی را با روش‌های یادگیری تقویتی بحث شده در فصل ۱۳ را بررسی کرده‌اند. (Mitchell and Thrun 1993) نیز کاربرد شبکه‌های عصبی مبتنی بر یادگیری توضیحی (به الگوریتم EBNN در فصل ۱۲ مراجعه کنید) را برای مسائل یادگیری تقویتی بررسی کرده‌اند.

۱۱,۵ خلاصه و منابع برای مطالعه‌ی بیشتر

نکات اصلی این فصل شامل موارد زیر می‌شود:

بر خلاف متدهای یادگیری استقرایی محض که به دنبال فرضیه‌ای می‌گردند که با داده‌های آموزشی متناسب باشد، متدهای تحلیلی محض به دنبال فرضیه‌ای می‌گردند که با دانش اولیه‌ی یادگیر تطبیق داشته و نمونه‌های آموزشی را نیز پوشش دهد. انسان‌ها نیز

---

[۱] utility analysis
[۲] lazy
[۳] incremental

گاهی از دانش قبلی برای هدایت بیان فرضیه‌های جدید استفاده می‌کنند. این فصل روش‌های تحلیلی محض را بیان می‌کند، در فصل آتی به روش‌های یادگیری ترکیبی تحلیلی استقرایی می‌پردازیم.

یادگیری مبتنی بر توضیحات نوعی یادگیری تحلیلی است که یادگیر هر نمونه‌ی آموزشی را (۱) با استفاده از تئوری قلمرو توضیح داده (۲) این توضیحات را برای تعیین شرط کلی درستی توضیحات بررسی کرده و (۳) فرضیه‌اش را برای تطابق با این شروط کلی بازنگری می‌کند.

الگوریتم Prolog-EBG الگوریتم یادگیری مبتنی بر توضیحاتی است که از horn clause های درجه اول برای نمایش تئوری قلمروی ورودی و فرضیه‌ی یاد گرفته استفاده می‌کند. در Prolog-EBG توضیح، یک اثبات Prolog است و فرضیه استخراجی از این توضیح ضعیف‌ترین پیش‌نویس این اثبات است. نتیجه، اینکه فرضیه‌ی خروجی Prolog-EBG را می‌توان از تئوری قلمروی‌اش نتیجه‌گیری کرد.

روش‌های یادگیری تحلیلی مثل Prolog-EBG، ویژگی‌های میانی مفیدی را به عنوان اثر جانبی بررسی نمونه‌های آموزشی ایجاد می‌کند. این روش تحلیلی برای ایجاد ویژگی مکمل روش آماری ایجاد ویژگی‌های میانی در متدهایی چون Backpropagation است. (ویژگی‌های واحدهای پنهان)

با وجود اینکه Prolog-EBG فرضیه‌ای را که از محدوده‌ی نتیجه‌گیری تئوری قلمروی‌اش تجاوز کند خروجی نمی‌دهد، اما روش‌های یادگیری استنتاجی چنین قابلیتی را دارا هستند. برای مثال، تئوری قلمروی ---

یکی از کلاس‌های مهم مسائلی که تئوری قلمروی کامل و درست برایشان موجود است، کلاس مسائل جستجوی فضاهای حالت بزرگ است. سیستم‌هایی چون Prodigy و SOAR کارایی متدهای یادگیری مبتنی بر توضیحات را برای پیدا کردن خودکار دانش کنترل جستجوی‌ای که سرعت حل مسائل را در چنین مسائلی ممکن می‌سازد اثبات کرده‌اند.

برخلاف وضوح کارایی متدهای یادگیری توضیحی برای انسان‌ها، پیاده‌سازی استنتاجی محض مثل Prolog-EBG ضعف‌هایی، نظیر اینکه فرضیه‌ی خروجی فقط زمانی که تئوری قلمرو صحیح است درست است، دارد. در فصل بعدی به روش‌هایی خواهیم پرداخت که متدهای یادگیری تحلیلی و استقرایی را ترکیب کرده تا از تئوری قلمروی ناکامل و داده‌های آموزشی محدود یادگیری ممکن شود.

ریشه‌های متدهای یادگیری تحلیلی را می‌توان در میان کارهای اولیه‌ی Fikes (1972) در یادگیری عملگرهای کلی (macro-operators) در بررسی عملگرهای ABSYTIPS و کار Soloway (1977) در استفاده‌ی دانش قبلی محض در یادگیری پیدا کرد. روش‌های یادگیری مبتنی بر توضیحات، مشابه آنچه در طول فصل به آن پرداختیم، اولین بار در تعدادی از سیستم‌های طراحی شده در اوایل دهه‌ی ۸۰ شامل DeJong (1981); Mitchell (1981); Winston et al. (1983); Silver (1983) ظهور پیدا کرد. DeJong and Mooney (1986) و Mitchell et al. (1986) توضیحات کلی لازم ----

مهم‌ترین تلاش‌های استفاده از یادگیری مبتنی بر توضیحات با تئوری قلمروهای کامل در محدوده‌ی یادگیری کنترل جستجو یا یادگیری افزایش سرعت (speedup) بوده است. سیستم Soar که توسط Laird et al. (1986) و Prodigy که توسط Carbonell et al. (1990) توصیف شد در میان پیشرفته‌ترین سیستم‌هایی هستند که از متدهای یادگیری توضیحی برای یادگیری حل مسائل استفاده می‌کنند. Rosenbloom and Laird (1986) رابطه‌ی نزدیک بین متد یادگیری Soar (یا همان chunking) و دیگر متدهای یادگیری توضیحی را بحث می‌کنند. اخیراً، Dietterich and Flann (1995) ترکیب روش‌های مبتنی بر توضیحات را با یادگیری تقویتی برای کنترل جستجو بررسی کرده‌اند.

با وجود اینکه هدف عمده‌ی ما در اینجا مطالعه‌ی یادگیری الگوریتم‌های یادگیری ماشین است، خالی از لطف نیست که اشاره شود که تحقیقات بر روی یادگیری انسان به این حدس گرویده که یادگیری انسان نیز مبتنی بر توضیحات است. برای مثال، Ahn et al. (1987) و Qin et al. (1992) مدارکی مبنی بر این حدس که انسان‌ها از روش‌های مبتنی بر توضیحات استفاده می‌کنند ارائه می‌کنند. (Wisniewski and Medin 1995) تحقیقاتی را بر روی یادگیری انسان‌ها که اثر متقابل غنی‌ای از دانش قبلی و داده‌های مشاهده شده برای تأثیر در فرآیند یادگیری را نشان می‌دهد را گزارش می‌کنند. Kotovsky and Baillargeon (1994) تحقیقاتی را توصیف کرده که نشان می‌دهد حتی بچه‌های ۱۱ ماهه نیز از دانش قبلی‌ای که یاد گرفته‌اند کمک می‌گیرند.

بررسی انجام گرفته بر روی یادگیری توضیحی مشابه نوع خاصی از متدهای بهینه‌سازی برنامه‌هاست که توسط برنامه‌های Prolog به کار می‌رود، مثل (Partition evaluation). (Harmelen and Bundy 1988) به رابطه‌ی بین این دو می‌پردازد.

تمرینات

۱۱,۱ مسئله‌ی یادگیری مفهوم هدف "زوج افرادی که در یک خانه زندگی می‌کنند" را در نظر بگیرید که با HouseMates(x,y) نمایش داده می‌شود، در زیر نمونه‌ای از این مفهوم آورده شده است:

HouseMates(Joe,Sue)

Person(Joe) Person(Sue)

Sex(Joe,Male) Sex(Sue,Female)

HairColor(Joe,Black) HairColor(Sue,Brown)

Height(Joe,Short) Height(Sue,Short)

Nationality(Joe,US) Nationality(Sue,US)

Mother(Joe,Marry) Mother(Sue,US)

Age(Joe,8) Age(Sue,6)

درباره‌ی این مفهوم هدف تئوری قلمروی زیر موجود است:

HouseMates(x,y)←InSameFamily(x,y)

HouseMates(x,y)←FraternityBrothers(x,y)

InSameFamily(x,y) ← Married(x,y)

InSameFamily(x,y) ← Youngster(x)/\ Youngster(y)/\SameMother(x,y)

SameMother(x,y) ← Mother(x,z)/\Mother(y,z)

Youngster(x) ← Age(x,a)/\LessThan(a,10)

از Prolog-EBG برای یادگیری این مفهوم با تئوری قلمرو و داده‌های آموزشی بالا استفاده کنید. در کل،

(a) رفتار Prolog-EBG را به طور دستی دنبال کنید؛ توضیحات ایجاد شده برای نمونه‌های آموزشی و حاصل برازش مفهوم هدف را از این توضیحات و دسته horn clause های حاصل را ثبت کنید.

(b) مفهوم "افرادی که با Joe در یک خانه زندگی می‌کنند" را به جای "زوج افرادی که در یک خانه زندگی می‌کنند" را در نظر بگیرید. این مفهوم هدف را با فرمول‌های بالا بازنویسی کنید. همان نمونه‌ی آموزشی و همان تئوری قلمرو را در نظر بگیرید، چه horn clause هایی توسط Prolog-EBG تولید می‌شود؟---

۱۱,۲ همان‌طور که در بخش ۱۱,۳,۱ نیز اشاره شد، Prolog-EBG می‌تواند ویژگی‌های جدید مفیدی که به طور صریح در نمونه‌ها بیان نشده اما بر حسب ویژگی‌های در تعمیم روی نمونه‌ها مفید است را ایجاد کند. این ویژگی‌ها بر اثر بررسی توضیحات نمونه‌های آموزشی ایجاد می‌شود. متد دیگری که برای پیدا کردن ویژگی‌های مفید غیرصریح Backpropagation در شبکه‌های عصبی است، در این متد ویژگی‌های جدید در واحدهای پنهان بر اساس ویژگی‌های آماری تعداد زیادی از نمونه‌ها ایجاد می‌شود. آیا می‌توانید راهی برای ترکیب این روش‌های آماری و استقرایی برای ایجاد ویژگی‌های جدید ارائه کنید؟ (توجه: این مسئله هنوز جزو قسمت‌های آزاد تحقیق محسوب می‌شود)

≺¬≠≰]∏∧∨

فرهنگ لغات تخصصی فصل (فارسی به انگلیسی)

افزایش سرعت	Speedup
افزایش سرعت جستجوی برنامه‌های پیچیده تعیین	speed up complex search programs
افزایشی	Incremental
برگ	Leaf nodes
برنامه‌ریز	Planner
برنامه‌ریزی و انجام	planning and schedualing
برنامه‌نویسی منطقی استقرایی	inductive logic programming
بستگی استنتاجی	deductive closure
پس راندن	Regress
تحلیل تأثیر قوانین	utility analysis
تعیین	Determination
تنبل	Lazy
توجیه	Justification
تئوری قلمرو	domain theory
جستجوی زنجیروار معکوس	backward chaining search
جفت کردن	Match
حلال مسئله	problem solver
خودکار	Autonomous

درخت‌های اثبات	proof tree
درست	Correct
زیر هدف	Subgoal
ضعیف‌ترین پیش‌نویس	weakest preimage
غیر مرتبط	Irrevelant
فرض‌ها	Assertion
فضای وضعیتی	state-space
قید	Constraint
کامل	Perfect
گرداوری دانش	knowledge compilation
الگوریتم ترتیبی	sequential covering algorithm
مجموعه توانی‌ای	Superset
مرتبط	Relevant
مسائل متمرکز بر جستجوی برنامه‌ریزی و بهینه‌سازی	search-intensive planning and optimization problems
مستقل از قلمرو	domain-independent
منطقی	Logical
منفی در زمان شکست	negation-as-failure
ویژگی	Feature
هدف‌های میانی	means-ends
همه‌جانبه	Complete
یادگیری تحلیلی	analytical learning
یادگیری توضیحی	explanation-based learning (ELB)
یادگیری مرتبه‌ی دانش	knowledge level learning
عملگرهای کلی	macro-operators