دانشکده علوم ریاضی
دانشگاه صنعتی شریف

نظریه یادگیری محاسباتی

امید اعتصامی، محمدهادی فروغمنداعرابی
بهار ۱۳۹۳

مدل یادگیری احتمالا تقریبا درست (PAC)

جلسه‌های دوم تا ششم

نگارنده: ابوالفضل طاهری

نظریه‌ی محاسباتی یادگیری[۱] به بررسی پیچیدگی‌های محاسباتی الگوریتم‌های یادگیری ماشین می‌پردازد. مدل‌های مختلفی برای آنالیز و تحلیل این الگوریتم‌ها معرفی شده است. در این نوشتار به مدل یادگیری احتمالا تقریبا درست[۲] یا به طور خلاصه یادگیری PAC می‌پردازیم. بنیان‌گذار این مبحث دانشمند بریتانیای لسلی والینت[۳] است که در مقاله‌ی A Theory of the Learnable به آن می‌پردازد. در مقدمه‌ی این مقاله آمده است:

> هدف اصلی این مقاله این است که نشان دهد آیا ممکن است ماشینی برای یادگیری طراحی کرد که سه خاصیت زیر را داشته باشد:
>
> ۱) بتوان اثبات کرد که ماشین تمام کلاس‌های مفهوم را فرامی‌گیرد. بعلاوه، این کلاس‌ها را می‌تواند از هم تمیز دهد.
>
> ۲) کلاس‌های مفهوم مناسب و نابدیهی است و برای یک دانش کلی است.

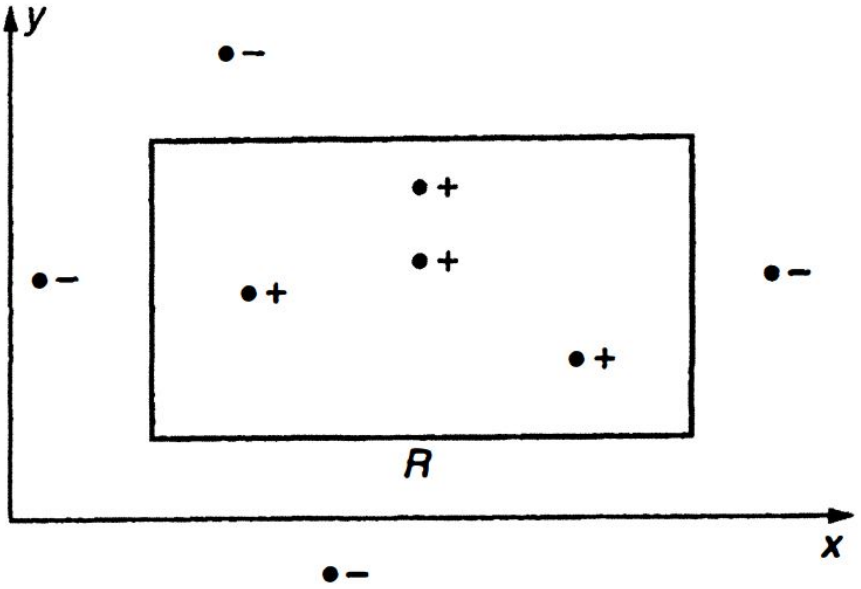
[۱] Computational learning theory
[۲] Probability Approximately Correct Learning
[۳] Leslie Valiant

۳) تعداد مراحل و عملیات مورد نیاز برای این‌کار توسط ماشین باید معقول (چندجمله‌ای) باشد.

در ادامه‌ی این مقاله و کارهای بعدی سعی بر این بود که سه خاصیت فوق برقرار باشد. با یک مثال شروع می‌کنیم.

۱ بازی یادگیری مستطیل

این بازی یک بازی یک نفره است و هدف یادگیری مستطیل R است که اضلاع آن موازی محورهای مختصات در صفحه‌ی $\mathbb{R}^2$ است. مستطیل R را مستطیل هدف[۴] می‌گوییم. تنها اطلاعاتی که به بازیکن در مورد مستطیل R داده می‌شود به این شکل ا
[illegible] له به بازیکن اعلام شده و
گ
م
مجهول R با نمونه‌هایی از



شکل ۱: مستطیل هدف در صفحه با نمونه‌های مثبت و منفی

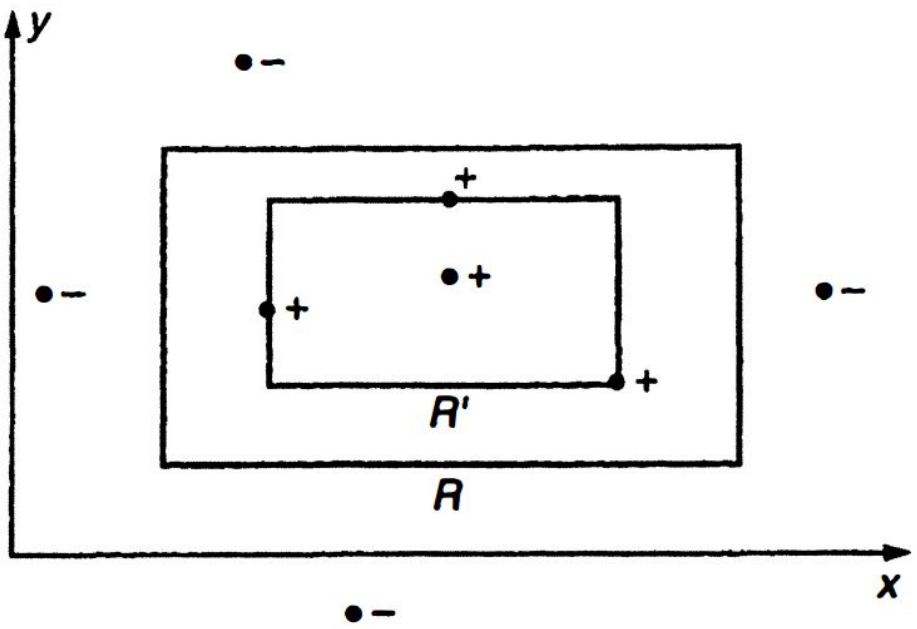
هدف بازیکن این است که با استفاده از تعداد کمی مثال و محاسباتی که از لحاظ عملی ممکن باشد، مستطیل R' را معرفی کند که تقریب مناسبی از R باشد. به R' یک فرضیه[۵] می‌گوییم. فرضیه‌ای که توسط بازیکن ارائه می‌شود توسط نقاط تصادفی نسبت به توزیع احتمال $\mathcal{D}$ سنجیده می‌شود و فرضیه‌ای که بازیکن ارائه داده است باید تا حدودی خوبی به درستی مشخص کند که نقاط جدید درون مستطیل هدف قرار دارند یا خارج از آن هستند. بنابراین خطای فرضیه‌ی R' را می‌توان احتمال قرار گرفتن نقاط در ناحیه‌ای که فرضیه با مستطیل هدف تفاوت دارد، نسب به توزیع $\mathcal{D}$ در نظر گرفت. به عبارتی احتمال قرار گرفتن نقاط در ناحیه‌ی $R\Delta R' = (R - R') \cup (R' - R)$.

قبل از این‌که به ادامه‌ی بازی یادگیری مستطیل بپردازیم، مثالی ملموس‌تر از آن ارائه می‌دهیم: فرض کنید می‌خواهیم مفهوم "هیکل متوسط" را در یک جامعه یاد بگیریم. فرض کنید قد و وزن افراد برای این مفهوم در یک بازه قرار می‌گیرند. به طور مثال قد باید بین ۱۷۰ تا ۱۹۰ سانتی‌متر و وزن بین ۶۵ تا ۹۵ کیلوگرم باشد. هر فرد در جامعه را می‌توان با یک نقطه در صفحه نمایش داد که مولفه‌ی اول قد و مولفه‌ی دوم وزن فرد است. حال یادگیری هیکل متوسط معادل یادگیری مستطیلی است که اضلاع آن موازی محورهای مختصات است. در هر مرحله یک فرد انتخاب شده، وزن و قد آن اعلام می‌شود و گفته می‌شود که هیکل شخص متوسط است یا خیر. به این ترتیب باید مستطیلی ارائه دهیم که با تقریب خوبی هیکل متوسط در جامعه را مشخص می‌کند.

[۴] target
[۵] hypothesis

حال فرض کنید یادگیرنده (بازیکن) هر شخص در جامعه را با احتمال برابری در دسترس دارد. حتی با این فرض، نقاط ارائه شده در صفحه لزوما توزیع یکنواخت نخواهند داشت (زیرا تمام قد و وزن‌های به احتمال یکسانی در جامعه نیستند و به مولفه‌های زیادی وابسته‌اند. به طور مثال فرض کنید جامعه‌ی ما را بسکتبالیست‌های یک شهر تشکیل می‌دهند.)، با این حال توزیع $\mathcal{D}$ ثابت است و تغییر نمی‌کند. به همین خاطر در بازی یادگیری اجازه می‌دهیم توزیع $\mathcal{D}$ دلخواه باشد اما همواره در طول یادگیری و تست ثابت باشد و هر مثال انتخاب شده مستقلا با این توزیع داده می‌شود. بنابراین در مورد مساله‌ی هیکل متوسط حتی لازم نیست فرض کنیم که انتخاب افراد به طور یکنواخت صورت می‌گیرد.

یک استراتژی ساده و کارا برای بازیکن در بازی یادگیری مستطیل وجود دارد: بعد از این‌که به تعداد کافی بزرگ، m مثال



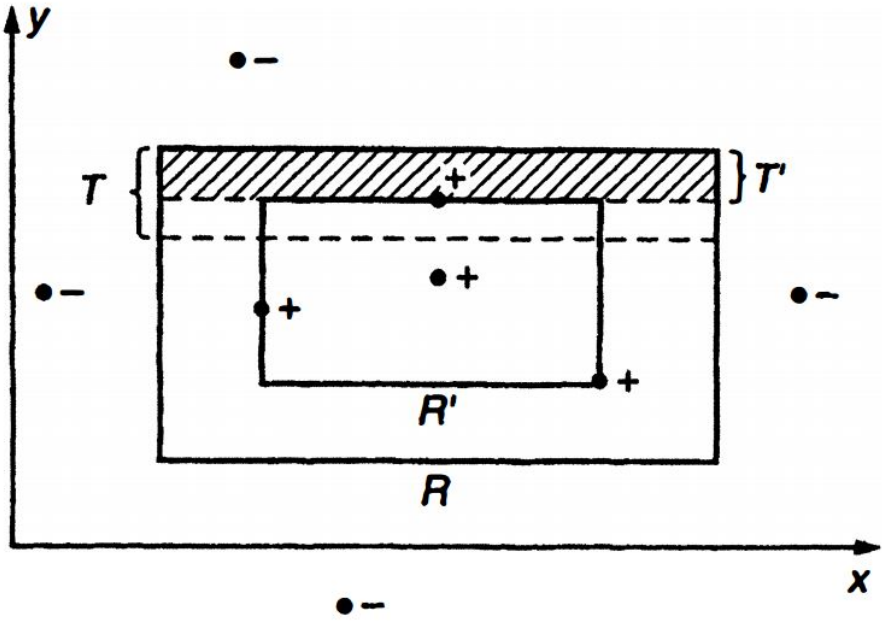
شکل ۲: کوچکترین مستطیل R' که توسط نمونه‌ها تعریف می‌شود.

حال نشان می‌دهیم برای مستطیل هدف R و هر توزیع دلخواه $\mathcal{D}$ و برای تمام مقادیر کوچک $0 < \epsilon, \delta \leq \frac{1}{2}$ می‌توان تعداد نمونه‌ها، m، را به گونه‌ای مشخص کرد که با احتمال حداقل $1 - \delta$ کوچکترین مستطیل نسبت به R و $\mathcal{D}$ حداکثر خطای ϵ داشته باشد.

قبل از هر چیز توجه کنید که کوچکترین مستطیل R' همواره درون مستطیل هدف R قرار می‌گیرد بنابراین $R \Delta R' = R - R'$. می‌توانیم تفاضل $R - R'$ را به صورت اجتماع چهار نوار مستطیلی مشخص کنیم. به عنوان مثال، نوار بالایی در شکل ۳ به صورت هاشورخورده با T' نمایش داده شده است. به همین ترتیب می‌توان نوارهایی پایینی، چپ و راست را مشخص کرد. این نوارها در چهارگوشه اشتراک‌هایی دارند. بنابراین اگر نشان دهیم وزن هر کدام از این نوارها تحت $\mathcal{D}$ حداکثر $\frac{\epsilon}{4}$ است در این صورت می‌توان نتیجه گرفت خطای R' حداکثر $4(\frac{\epsilon}{4}) = \epsilon$ خواهد بود. (در این‌جا هر کدام از ناحیه‌های اشتراک را دوبار در محاسبه‌ی خطا آورده‌ایم).

سعی می‌کنیم وزن T' را به گونه‌ای مورد تحلیل قرار دهیم. مستطیل T را به گونه‌ای انتخاب می‌کنیم که تحت $\mathcal{D}$ دقیقا وزن $\frac{\epsilon}{4}$ داشته باشد. برای این‌کار از یال بالایی R شروع می‌کنیم و به سمت پایین حرکت می‌کنیم تا زمانی که وزن آن دقیقا برابر $\frac{\epsilon}{4}$ شود. شکل ۳ را ببینید. به وضوح T' وزنی بیشتر از T دارد اگر و تنها اگر شامل T باشد (این اتفاق در شکل ۳ رخ نداده است). بعلاوه T' شامل T است اگر و تنها اگر هیچ نقطه‌ای از نمونه‌ی S در T قرار نگیرد. زیرا اگر S شامل نقطه‌ی $p \in T$ باشد، در این صورت چون p در R قرار دارد پس مثبت است و با توجه به انتخاب R'، p باید در R' قرار گیرد و

[۶] tightest fit



شکل ۳: تحلیل خطا به کمک نوار هاشورخورده‌ی بالایی T'. نوار T تحت $\mathcal{D}$ دقیقا وزنی برابر $\frac{\epsilon}{۴}$ دارد.

بنابراین می‌توانیم R' را از بالا به توی T گسترش دهیم.

با توجه به تعریف T، احتمال این‌که یک نقطه‌ی انتخاب شده تحت توزیع $\mathcal{D}$ خارج از T قرار گیرد دقیقا برابر $۱ - \frac{\epsilon}{۴}$ است. بنابراین احتمال این‌که m نقطه‌ی مستقل که تحت $\mathcal{D}$ انتخاب شده‌اند همگی خارج از ناحیه‌ی T قرار بگیرند برابر است با $(۱ - \frac{\epsilon}{۴})^m$. در این‌جا از این واقعیت استفاده شده است که احتمال رخ دادن اتفاقات مستقل با یکدیگر برابر حاصل‌ضرب احتمال رخ دادن هر کدام است. این تحلیل برای هر یک از سه ناحیه‌ی دیگر از $R - R'$ برقرار است. بنابراین احتمال این‌که هر یک از چهار نوار $R - R'$ وزنی حداکثر از $\frac{\epsilon}{۴}$ داشته باشند کران بالای $۴(۱ - \frac{\epsilon}{۴})^m$ را به دست می‌دهد. این کران از این‌جا به دست می‌آید که اگر A و B دو رویداد باشد در این صورت

$$\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$$

بنابراین کافی است m را به گونه‌ای ارائه دهیم که $۴(۱ - \frac{\epsilon}{۴})^m \leq \delta$ برقرار باشد، در این صورت با احتمال $۱ - \delta$ با m مثال، وزن ناحیه‌ی $R - R'$ دارای کران ϵ است که همان اداعای موردنظر است. با توجه به نامساوی

$$(۱ - x) \leq e^{-x}$$

نتیجه می‌شود اگر m در نامساوی $۴e^{-\epsilon m/۴} \leq \delta$ صدق کند در نامساوی قبلی نیز صدق می‌کند. از این نامساوی به دست می‌آوریم

$$m \geq (\frac{\epsilon}{۴}) \ln(\frac{۴}{\delta})$$

به طور خلاصه الگوریتم tightest-fit با استفاده از حداقل $(\frac{۴}{\epsilon}) \ln(\frac{۴}{\delta})$ مثال، فرضیه‌ی R' را ارائه می‌دهد که می‌توانیم ادعا کنیم با احتمال حداقل $۱ - \delta$، R' می‌تواند نقاط جدید را با خطای حداکثر ϵ تفکیک کند.

الگوریتم فوق برای هر توزیع احتمال برقرار است. تنها کافی است تنها کافی است نقاط به طور مستقل انتخاب شوند. با افزایش اندازه‌ی نمونه می‌توان فرضیه را بهتر کرد. در واقع برای افزایش دقت[۷] ϵ باید کاهش یابد و برای افزایش اطمینان[۸] δ باید کاهش یابد. الگوریتم فوق کاراست: اندازه‌ی نمونه تابعی برحسب $\frac{۱}{\epsilon}$ و $\frac{۱}{\delta}$ است که به آهستگی رشد می‌کند (به ترتیب، به طور خطی و لگاریتمی). بعلاوه برای یک نمونه‌ی داده شده محاسبات انجام شده برای بدست آوردن فرضیه، محاسباتی است که زمان زیادی نمی‌گیرد.

[۷] accuracy
[۸] confidence

سؤال. *اگر توزیع $\mathcal{D}$ مشخص باشد، آیا می‌توان الگوریتم بهتری ارائه کرد؟*

سؤال. *اگر شرط موازی بودن بال‌ها با محورهای مختصات را نداشته باشیم، آیا باز هم الگوریتمی برای یادگیری می‌توان یافت؟*

۲ مدل کلی

در این بخش مدل یادگیری را ارائه می‌دهیم که بخش اصلی اشیاء مورد مطالعه در این‌جاست: مدل یادگیری احتمالا تقریبا درست یا مدل یادگیری PAC. ویژگی‌های زیادی در بازی یادگیری مستطیل و راه‌حل آن وجود دارد که در مدل PAC ضروری به نظر می‌رسد. قبل از ارائه‌ی تعریف کلی این ویژگی‌ها را برمی‌شمریم:

- هدف از بازی، یادگیری مجموعه‌ی مجهول هدف بود، اما مجموعه‌ی هدف دلخواه نبود. در واقع یک محدودیت قوی روی مجموعه‌ی هدف وجود داشت: یک مستطیل در صفحه است که اضلاع آن موازی محورهای مختصات‌اند.
- یادگیری بر اساس تنظیمات احتمالاتی صورت می‌گرفت. مثال‌های مستطیل هدف به طور تصادفی از توزیع احتمالی انتخاب می‌شد که ناشناخته بود و هیچ محدودیتی روی آن وجود نداشت.
- فرضیه‌ی بازیکن نسبت به تنظیمات انجام شده در مرحله‌ی یادگیری انجام می‌شود و این به ما اجازه داده می‌شد تنها تقریبی از هدف را ارائه دهیم. استراتژی کوچکترین مستطیل، به طور دقیق مستطیل هدف را مشخص نمی‌کند اما می‌توان با احتمال بسیار کمی نسبت هدف آن را مشخص کرد.
- پاسخ موردنظر کارا بود: برای داشتن خطای کم و اعتماد زیاد نیاز به تعداد زیادی مثال نبود و محاسبات سریع انجام می‌شد.

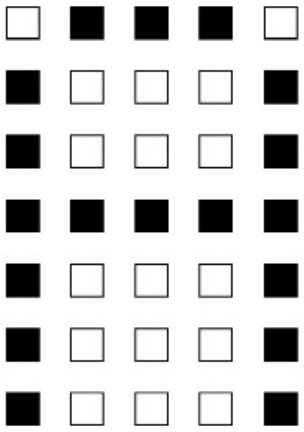
حال می‌خواهیم مدلی برای یادگیری ارائه دهیم که خواص فوق را دربر گیرد.

۱.۲ تعریف مدل PAC

فرض کنید X مجموعه نقاط مورد بحث باشد که به آن فضای نمونه[۹] می‌گوییم. می‌توان X را به عنوان مجموعه‌ی کد شده‌ی نمونه‌ها یا اشیاء مورد بحث در فضای یادگیری درنظر گرفت. در بازی یادگیری مستطیل، فضای نمونه، فضای تمام نقاط در صفحه‌ی اقلیدسی، $\mathbb{R}^2$ است. به عنوان مثالی دیگر می‌توان مساله‌ی تشخیص کاراکتر[۱۰] را مطرح کرد. هر کاراکتر را می‌توان با یک آرایه‌ی دو بعدی از صفر و یک نمایش داد که صفر نمایانگر رنگ سفید و یک نمایانگر رنگ سیاه است که می‌توان برای یک نمایش استاندارد طول و عرض را مشخص کرد و همگی نمایش‌ها را در چنین طول وعرضی بیان نمود. بنابراین در این‌جا فضای نمونه، فضای تمام آرایه‌های دوبعدی از صفر و یک با طول و عرض مشخص است. شکل ۴ یک نمایش از حرف A را به صورت آرایه‌ی دوبعدی نمایش می‌دهد.

یک مفهوم[۱۱] روی X یک زیرمجموعه‌ی $c \subset X$ از فضای نمونه است. در بازی مستطیل، مفهوم‌ها نواحی مستطیلی موازی محورهای مختصات‌اند. در شناسایی کاراکترها، یک مفهوم می‌تواند مجموعه‌ی تمام آرایه‌هایی باشد که حرف A را نمایش می‌دهند.

[۹] instance space
[۱۰] character recognition
[۱۱] concept



شکل ۴: نمایش کاراکترها به صورت آرایه‌های دوبعدی

یک مفهوم را می‌توان به عنوان مجموعه‌ی تمام نمونه‌های مثبت در نظر گرفت. معادلا می‌توانیم یک مفهوم را به صورت یک نگاشت بولی $c : X \to \{۰,۱\}$ بیان کرد به این ترتیب که $c(x) = ۱$ است اگر x مثالی مثبت باشد و $c(x) = ۰$ اگر مثالی منفی باشد. به خاطر این نوع نمایش گاهی اوقات X را فضای ورودی[۱۲] می‌گویند.

یک کلاس مفهوم $\mathcal{C}$ روی X گردایه‌ای از مفاهیم روی X است. در بازی مستطیل، مستطیل هدف از کلاس $\mathcal{C}$، تمام مستطیل‌های موازی محورهای مختصات انتخاب می‌شود. کلاس‌های مفهوم مورد علاقه‌ی ما به اندازه‌ی کافی مشخص و شامل یک دانش کلی هستند. به عنوان مثال، بر پایه‌ی ساختار منطقی، فرض کنید مجموعه‌ی $x_۱, \ldots, x_n$، n متغیر بولی باشند و X مجموعه‌ی تمام گزاره[۱۳]های براساس این متغیرها باشد ($X = \{۰,۱\}^n$). فرض کنید مفهوم c روی $\{۰,۱\}^n$ تمام مثال‌های مثبتی باشد که در فرمول بولی f_c روی $x_۱, \ldots, x_n$ صدق می‌کنند. در این صورت یک کلاس مفهوم که مورد علاقه‌ی ما است می‌تواند به این شکل تعریف شود: تمام فرمول‌های بولی f_c که در تعدادی محدودیت طبیعی نحوی صدق می‌کند؛ مثلا به شکل فرم نرمال فصلی[۱۴] (DNF) است و تعداد مولفه‌های آن کم است. (یک فرم نرمال فصلی به صورت ترکیب "یا" از چندین بخش است که هر کدام از بخش‌ها به صورت "و" از چند متغیر یا نقیض آن تشکیل شده است. به عنوان مثال $(A \wedge \neg B \wedge \neq C) \vee (\neg D \wedge E \wedge F)$ یک فرم نرمال است[۱۵]).

در این مدل، الگوریتم یادگیری به مثال‌های مثبت و منفی از مفهوم مجهول هدف c که از کلاس شناخته شده‌ی مفاهیم $\mathcal{C}$ انتخاب شده است، دسترسی دارد. الگوریتم یادگیری باید بتواند فرضیه‌ای برای مفهوم ارائه دهد که مقدار مثبت و منفی را با دقت خوبی جدا کند. دقت کنید که در مدل، کلاس هدف $\mathcal{C}$ برای الگوریتم یادگیری مشخص است؛ به عبارتی برای طراح الگوریتم تضمین شده است که مفهوم هدف از کلاس $\mathcal{C}$ انتخاب می‌شود با این حال الگوریتم باید برای هر $c \in \mathcal{C}$ پاسخ صحیحی ارائه دهد.

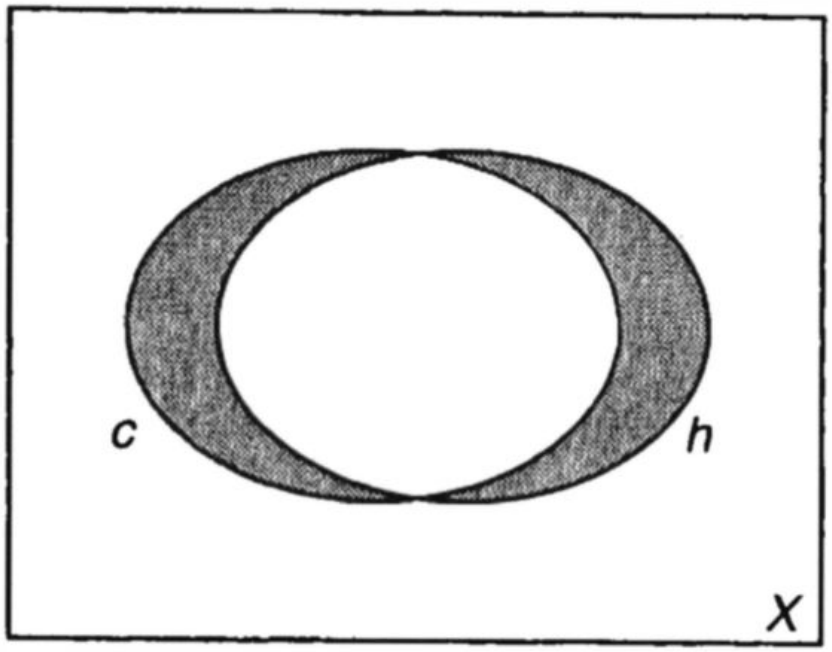
فرض کنید $\mathcal{D}$ توزیع احتمالی روی فضای نمونه، X باشد که در نظر گرفته شده است. این توزیع را توزیع هدف[۱۶] می‌نامیم. اگر h مفهوم دلخواهی روی X باشد، در این صورت $\mathcal{D}$ این امکان را به ما می‌دهد که بتوانیم خطای بین h و مفهوم هدف c را محاسبه کنید: تعریف می‌کنیم

$$\text{error}(h) = \Pr_{x \in \mathcal{D}}[c(x) \neq h(x)]$$

دقت کنید که مفاهیم c و h توابع بولی هستند و منظور از اندیس $x \in \mathcal{D}$ در فرمول فوق به این معنی است که x به طور تصادفی

[۱۲] input space
[۱۳] assignment
[۱۴] disjunctive normal form
[۱۵] وی‌کی‌پدیا
[۱۶] target distribution

نسبت به $\mathcal{D}$ انتخاب می‌شود. بنابراین $\text{error}(h)$ به طور مستقیم به c و $\mathcal{D}$ وابسته است.



شکل ۵: نمودار ون از دو مفهوم. ناحیه‌ی هاشورخورده تفاضل متقارن را نشان می‌دهد.

فرض کنید $\text{EX}(c, \mathcal{D})$ تابعی باشد که در زمان واحد اجرا شده و در هر مرتبه صدا زدن آن مثال برچسب‌دار $< x, c(x) >$ را برمی‌گرداند که x به طور تصادفی نسبت به $\mathcal{D}$ انتخاب شده است. این متد را اوراکل[۱۸] می‌گوییم. الگوریتم یادگیری به این اوراکل وقتی مفهوم هدف c باشد دسترسی دارد. با توجه به مطالب گفته شده، در حالت ایده‌آل انتظار داریم الگوریتم یادگیرید در خواص زیر صدق کند:

- تعداد فراخوانی‌های $\text{EX}(c, \mathcal{D})$ کوچک باشد؛ به وسیله‌ی یک چندجمله‌ای از پارامترها کران‌دار باشد.
- محاسبات انجام شده کم باشد.
- خروجی الگوریتم فرضیه‌ای باشد مانند مفهوم h به طوری که $\text{error}(h)$ کوچک باشد.

دقت کنید که تعداد فراخوانی‌های $\text{EX}(c, \mathcal{D})$ توسط الگوریتم یادگیری به وسیله‌ی زمان اجرای الگوریتم یادگیری محدود شده است.

با توجه به مطالبی که تا این‌جا گفته شد می‌توانیم تعریف اولیه‌ای برای مدل یادگیری احتمالا تقریبا درست ارائه دهیم. این تعریف، تنها یک تعریف اولیه است و به زودی شروط دیگری به آن اضافه خواهد شد.

تعریف ۱. مدل PAC، تعریف اولیه. *فرض کنید $\mathcal{C}$ یک کلاس مفهوم روی X باشد. می‌گوییم $\mathcal{C}$ قابل یادگیری PAC[۱۹] است اگر الگوریتم L وجود داشته باشد به طوری که: برای هر مفهوم $c \in \mathcal{C}$، برای هر توزیع $\mathcal{D}$ روی X، و برای هر $0 < \epsilon, \delta < \frac{1}{2}$، اگر L به اوراکل $\text{EX}(c, \mathcal{D})$ دسترسی داشته باشند و ϵ و δ به عنوان ورودی داده شده باشند، در این صورت با احتمال حداقل $1 - \delta$، L فرضیه $h \in \mathcal{C}$ را به عنوان خروجی می‌دهد به طوری که $\text{error}(h) \leq \epsilon$. این احتمال به وسیله‌ی مثال‌های تصادفی که با فراخوانی $\text{EX}(c, \mathcal{C})$ انتخاب می‌شود و هر تصادفی‌سازی که درون الگوریتم L وجود دارد، حاصل می‌شود.*

[۱۷] Venn diagram
[۱۸] oracle
[۱۹] PAC learnable

اگر L در زمان چندجمله‌ای برحسب $\frac{1}{\epsilon}$ و $\frac{1}{\delta}$ اجرا شود می‌گوییم $\mathcal{C}$ به طور کارا قابل یادگیری PAC[۲۰] است. در تعریف فوق به ϵ پارامتر خطا[۲۱] و به δ پارامتر اطمینان[۲۲] می‌گوییم.

فرضیه‌ی $h \in \mathcal{C}$ در الگوریتم یادگیری PAC، با احتمال بالایی تقریبا درست است به همین خاطر به آن یادگیری احتمالا تقریبا درست می‌گویند.

دو نوع اتفاق ناخوشایند در الگوریتم یادگیری PAC ممکن است رخ دهد که توسط ϵ و δ کنترل می‌شوند و به همین خاطر این پارامترها مورد نیاز است. دو فرضیه ممکن است در مقادیر ناچیزی با هم متفاوت باشد که در نمونه‌های کوچک دیده نمی‌شود و نمی‌توان بین آن‌ها تفاوتی قائل شد، این یکی از دلایلی است که به پارامتر خطا نیاز داریم. پارامتر اطمینان نیز از این جهت مورد نیاز است که ممکن است الگوریتم به شدت بدشناس باشد و نمونه‌های خوبی از مفهوم هدف انتخاب نشود؛ به طور مثال ممکن است همواره یک نقطه انتخاب شود (در هر بار فراخوانی اوراکل همان یک نقطه ارائه شود).

دقت کنید که الگوریتم یادگیری PAC برای هر توزیع $\mathcal{D}$ کار می‌کند و تنها واقعیتی که از آن استفاده می‌شود این است که فرضیه‌ی ارائه شده توسط الگوریتم هم با همین توزیع $\mathcal{D}$ سنجیده می‌شود. به عنوان مثال در بازی یادگیری مستطیل اگر وزن بخش‌هایی از صفحه نسبت به توزیع $\mathcal{D}$ ناچیز باشد، الگوریتم یادگیری لازم نیست چندان توجهی به این نواحی داشته باشد.

با توجه به تعریف ارائه شده و گفته‌های بخش قبل در رابطه با بازی یادگیری مستطیل، می‌توانیم قضیه‌ی زیر را بیان کنیم. در این قضیه و در ادامه کار، فرض می‌کنیم مدل محاسباتی ما اجازه می‌دهد که یک عدد حقیقی را در یک ناحیه از حافظه ذخیره کنیم و برای عمل‌گرهای ابتدایی (جمع، ضرب یا تقسیم) بین اعداد حقیقی، زمان محاسباتی لازم برابر واحد است.

قضیه ۲. *کلاس مفهوم مستطیل‌های موازی محورهای مختصات در صفحه‌ی اقلیدسی $\mathbb{R}^2$ به طور کارا قابل یادگیری PAC است.*

۲.۲ اندازه‌ی نمایش و بعد نمونه

یک موضوع مهم در تعریف یادگیری PAC باقی مانده است. یک تفاوت بنیادین بین مفهوم (که تنها یک مجموعه یا یک تابع بولی است) و نمایش (کدینگ مجموعه یا تابع به وسیله‌ی نمادها) وجود دارد. کلاس مفاهیم از فرمول‌های بولی که مجموعه‌ای از گزاره‌ها در آن صدق می‌کند را در نظر بگیرید. یک مفهوم در این کلاس را می‌توان با یک فرمول f، یا با یک جدول ارزش[۲۳] یا با فرمول دیگری مانند f' که به طور منطقی با f معادل است نمایش داد. با این که تمام این موارد نمایش یک مفهوم است، اما در اندازه‌ی نمایش متفاوت هستند.

به طور مثال تابع زوجیت $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ که در آن $\oplus$ عمگلر یای انحصاری[۲۴] است را در نظر بگیرید. این تابع را می‌توان با گیت‌های $\wedge$، $\vee$ و $\neg$ به صورت یک مدار منطقی ارائه داد که تعداد گیت‌های استفاده شده به وسیله‌ی یک چندجمله‌ای از n کران‌دار است. اما اگر بخواهیم همین تابع را به صورت فرم نرمال وصفی نمایش دهیم، اندازه‌ی تابع به صورت تابعی نمایی از n خواهد بود. زیرا برای نمایش هر حالت به هر n متغیر نیاز داریم. اگر مثلا x_1 در یک فرمول $x_{i_1} \wedge \cdots \wedge x_{i_k}$ قرار نداشته باشد در این صورت اگر فرمول برای مقادیری از $x_{i_1}, \ldots, x_{i_k}$ مقدار مثلا ۱ را خروجی دهد، این مقدار مستقل از x_1 است در صورتی که می‌دانیم اگر f به ازای مقدار $x_1 = 1$ و مقادیری مشخص برای متغیرهای دیگر خروجی ۰ یا ۱ را بدهد برای $x_1 = 0$ و همان مقادیر قبلی برای دیگر متغیرها به ترتیب مقدارهای ۱ و ۰ را می‌دهد. بنابراین اگر بخواهیم f را نمایش دهیم، باید تمامی حالت‌های ممکن را در فرمول بیاوریم که نمایش آن از $O(2^n)$ خواهد بود.

به عنوان مثالی دیگر فضای اقلیدسی $\mathbb{R}^n$ با بعد بالا را در نظر بگیرید. یک چندوجهی[۲۵] محدب را می‌توان به وسیله‌ی

[۲۰] efficiently PAC learnable
[۲۱] error parameter
[۲۲] confidence parameter
[۲۳] truth table
[۲۴] exclusive-or operation
[۲۵] polytope

راس‌هایش یا یک ترکیب خطی از وجه‌های آن نمایش داد. این دو نمایش از لحاظ اندازه به طور نمایی با یکدیگر متفاوت خواهند بود. به طور خاص مکعب مستطیل nـ بعدی را در نظر بگیرید. تعداد راس‌های آن برابر 2^n است و برای نمایش آن به کمک راس‌ها اندازه‌ی نمایی داریم. اما تعداد وجه‌های آن $2n$ خواهد بود که به صورت چندجمله‌ای است.

از آن‌جایی که الگوریتم‌های یادگیری PAC تنها به مثال‌هایی از رفتار مفهوم هدف دسترسی دارد، هیچ اطلاعاتی در مورد نمایش مفهوم هدف ندارد در صورتی که در واقعیت نمایش‌های متفاوتی ممکن است برای آن وجود داشته باشد. با این حال برای ارائه‌ی فرضیه باید به نحوی آن را نمایش داد. اما نوشتن نمایش فرضیه، جزو زمان لازم برای اجرای الگوریتم محسوب می‌شود بنابراین کران پایینی برای زمان اجرای الگوریتم است. بنابراین باید به نحوی نمایش فرضیه را در تعریف بگنجانیم.

یک الگوی نمایش[۲۶] برای کلاس مفهوم $\mathcal{C}$ تابعی مانند $\mathcal{R} : \Sigma^* \to \mathcal{C}$ است که Σ یک الفبای متناهی از نمادها است. در صورتی که برای نمایش مفهوم به اعداد حقیقی نیاز داشته باشیم، مانند بازی یادگیری مستطیل، اجازه می‌دهیم تابع به صورت $\mathcal{R} : (\Sigma \cup \mathbb{R})^* \to \mathcal{C}$ بیان شود. هر رشته[۲۷] مانند $\sigma \in \Sigma^*$ را یک نمایش[۲۸] برای c (تحت $\mathcal{R}$) می‌گوییم در صورتی که داشته باشیم $\mathcal{R}(\sigma) = c$. دقت داشته باشید برای یک مفهوم c ممکن است نمایش‌های متعددی تحت $\mathcal{R}$ وجود داشته باشد.

برای به دست آوردن مفهوم اندازه‌ی نمایش، به همراه $\mathcal{R}$ نگاشت دیگری مانند $\text{size} : \Sigma^* \to \mathbb{N}$ را در نظر می‌گیریم که به هر نمایش $h \in \Sigma^*$ عدد طبیعی $\text{size}(h)$ را نسبت می‌دهد. دقت کنید که اجازه می‌دهیم $\text{size}(.)$ هر نگاشتی باشد؛ نتایج به دست آمده تحت یک تعریف خاص برای $\text{size}(.)$ تنها در صورتی معنی‌دار خواهد بود که تعریف طبیعی باشد. شاید واقعی‌ترین دید این باشد که وقتی $\Sigma = \{0,1\}$ (بنابراین یک کدگزاری دودویی از مفهوم داریم)، $\text{size}(h)$ را تعداد بیت‌های به کار رفته در h در نظر بگیریم (برای نمایش‌هایی که از اعداد حقیقی استفاده می‌شود، اغلب، به طور طبیعی برای هر عدد حقیقی یک واحد اندازه در نظر می‌گیریم). هرچند ممکن است از تعاریف دیگری برای اندازه‌ی نمایش در حالت دودویی استفاده کنیم با این حال تعریف اندازه همواره یک چندجمله‌ای از طول رشته‌ی دودویی تعریف است. برای مثال، می‌توانیم اندازه‌ی یک درخت تصمیم[۲۹] را تعداد گره‌های درخت در نظر بگیریم که همواره با یک عامل چندجمله‌ای از طول رشته‌ی دودویی می‌توان درخت را کدگذاری کرد.

تا این‌جا مفهوم اندازه را تنها برای نمایش‌ها (رشته‌های $h \in \Sigma^*$) ارائه کردیم. حال می‌خواهیم این تعریف را برای اندازه‌ی مفهوم هدف $c \in \mathcal{C}$ گسترش دهیم. از آن‌جایی که الگوریتم‌های یادگیری تنها به رفتار ورودی‌ـ خروجی c دسترسی دارد، در بدترین حالت باید فرض کنیم ساده‌ترین ساختار ممکن این رفتار را تولید می‌کند. بنابراین تعریف می‌کنیم:

$$\text{size}(c) \min_{\mathcal{R}(\sigma)=c} \{\text{size}(\sigma)\}$$

به عبارت دیگر اندازه c، اندازه‌ی ساده‌ترین نمایش ممکن برای مفهوم c تحت الگوی نمایش $\mathcal{R}$ است. به طور شهودی، زمانی که $\text{size}(c)$ بزرگ باشد، مفهوم c نسبت به الگوی نمایش پیچده‌تر خواهد بود. بنابراین طبیعی است که به الگوریتم‌های یادگیری اجازه‌ی انجام محاسبات بیشتری را برای این مفاهیم قائل شویم.

برای کلاس مفهوم $\mathcal{C}$، کلاس نمایش $\mathcal{C}$ را برای الگوی نمایش $\mathcal{R}$ در نظر می‌گیریم که از قبل مشخص شده است. در واقع، همواره کلاس مفهوم را تعریف می‌کنیم و آن را با الگوی نمایش مورد مطالعه قرار می‌دهیم.

برای اعضای فضای نمونه نیز، می‌توان مفهومی به نام اندازه یا بعد تعریف کرد. به عنوان مثال اگر فضای نمونه X_n، فضای اقلیدسی nـ بعدی، $\mathbb{R}^n$ باشد در این صورت هر مثال نمونه با n عدد حقیقی مشخص می‌شود و بنابراین طبیعی است که اندازه‌ی این نمونه را n در نظر بگیریم. همین روند را می‌توان برای فضای نمونه $X_n = \{0,1\}^n$ در نظر گرفت. به نظر می‌رسد در مطالعات ما تنها همین دو فضای نمونه مورد استفاده قرار می‌گیرد و بنابراین بعد آن یعنی n را نیز در مسائل

[۲۶] representation scheme
[۲۷] string
[۲۸] representation
[۲۹] decision tree

یادگیری مدنظر می‌گیریم. برای مثال اگر بخواهیم مساله‌ی یادگیری مستطیل را در فضای $\mathbb{R}^n$ بعد در زمان چندجمله‌ای انجام دهیم، بعد فضا در محاسبات تاثیرگذار است. اگر $\mathcal{C}_n$ را کلاس مفاهیم روی X_n در نظر بگیریم و قرار دهیم $X = \cup_{n\geq 1} X_n$ و $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$ آنگاه X و $\mathcal{C}$ خانواده‌ای نامتناهی از مسائل یادگیری را تعریف می‌کند که بعد آن‌ها افزایش می‌یابد.

با توجه به مطالبی که گفته شد، باید اندازه‌ی مفهوم هدف و بعد فضای نمونه را در مدل یادگیری در نظر بگیریم. پس تعریف جدیدی نیاز داریم:

تعریف ۳. مدل PAC، تعریف اصلاح شده. *فرض کنید $\mathcal{C}_n$ کلاس نمایش روی X_n باشد (که X_n برابر $\{0,1\}^n$ یا فضای اقلیدسی nـ بعدی است.) و $X = \cup_{n\geq 1} X_n$ و $\mathcal{C} = \cup_{n\geq 1}\mathcal{C}_n$. مدل یادگیری PAC اصلاح شده، همان تعریف قبل است با این تفاوت که اجازه می‌دهیم الگوریتم یادگیری در زمان چندجمله‌ای بر حسب n، $\text{size}(c)$، $\frac{1}{\epsilon}$ و $\frac{1}{\delta}$ برای مفهوم هدف $c \in \mathcal{C}_n$ اجرا شود.*

چون فضای X_n مورد مطالعه‌ی ما معمولا $\{0,1\}^n$ یا $\mathbb{R}^n$ است، مقدار n به طور صریح در نمونه‌ای که $\text{EX}(c,\mathcal{D})$ ارائه می‌دهد بیان می‌شود. بنابراین فرض می‌کنیم مقدار $\text{size}(c)$ به عنوان ورودی برای یادگیرنده فراهم است.

نکته‌ی قابل توجه دیگر این است که در بسیاری از کلاس‌های مفهوم تعریف طبیعی $\text{size}(c)$ به وسیله‌ی یک چندجمله‌ای از n کران‌دار است و بنابراین به دنبال الگوریتم‌هایی بگردیم که در زمان چندجمله‌ای تنها بر حسب n، $\frac{1}{\epsilon}$ و $\frac{1}{\delta}$ اجرا شوند. برای مثال اگر کلاس نمایش تمام فرمول‌های DNF با حداکثر ۳ مولفه را در نظر داشته باشیم، هر فرمول طولی حداکثر برابر با $3n$ دارد بنابراین چندجمله‌ای وابسته به اندازه‌ی فرمول هدف همان فرمولی است که به n وابسته است.

تمرین ۱. *با توجه به تعریف جدید، نشان دهید بازی یادگیری مستطیل دارای الگوریتم کارای PAC است (در واقع برای بعد دلخواه n باید نشان داد که الگوریتم کارای PAC وجود دارد.).*

۳ یادگیری فرمول‌های منطقی عطفی

به عنوان دومین نتیجه در مدل PAC نشان می‌دهیم عطف از لیترال‌های منطقی[۳۰] قابل یادگیری کارای PAC است. در این‌جا فضای نمونه $X_n = \{0,1\}^n$ است. هر $a \in X_n$ به عنوان یک گزاره از n متغیر بولی $x_1, \ldots, x_n$ تعبیر می‌شود و از نمادگذاری a_i برای نمایش بیت iام a استفاده می‌کنیم. فرض کنید کلاس نمایش $\mathcal{C}_n$ کلاس تمام عطف‌های حاصل از لیترال‌های روی $x_1, \ldots, x_n$ باشد (یک لیترال یک متغیر x_i یا نقیض آن $\bar{x}_i$ است). بنابراین عطف $x_1 \wedge \bar{x}_3 \wedge x_4$ نمایش مجموعه‌ی $\{a \in \{0,1\}^n : a_1 = 1, a_3 = 0, a_4 = 1\}$ خواهد بود. طبیعی است که اندازه‌ی گزاره‌ی عطفی را برابر طول لیترال‌های آن تعریف کنیم. بنابراین به وضوح برای هر $c \in \mathcal{C}_n$ داریم $\text{size}(c) \leq 2n$ (دقت کنید که در کدگزاری دودویی استاندارد، هر گزاره‌ی عطفی $c \in \mathcal{C}_n$ طولی از $O(n \log n)$ دارد). بنابراین برای این مساله باید الگوریتمی بیابیم که در زمان چندجمله‌ای برحسب n، $\frac{1}{\epsilon}$ و $\frac{1}{\delta}$ اجرا شود.

قضیه ۴. *کلاس نمایش تمام عطف‌های از لیترال‌های منطقی قابل یادگیری کارای PAC است.*

اثبات. در ابتدای الگوریتم فرض می‌کنیم فرضیه به صورت

$$h = x_1 \wedge \bar{x}_1 \wedge \cdots \wedge x_n \wedge \bar{x}_n$$

است. دقت کنید که هیچ گزاره‌ای در فرضیه‌ی اولیه صدق نمی‌کند. الگوریتم تمام مثال‌های منفی که توسط $\text{EX}(c,\mathcal{D})$ ارائه می‌شود را نادیده می‌گیرد. فرض کنید $< a, 1 >$ نمونه‌ی مثبتی باشد که توسط $\text{EX}(c,\mathcal{D})$ ارائه شده است. در این حالت

[۳۰] conjunctions of boolean literals

الگوریتم h را به این شکل اصلاح می‌کند: برای هر i، اگر $a_i = ۰$ باشد x_i را از h حذف می‌کنیم و اگر $a_i = ۱$ باشد $\bar{x}_i$ را از h حذف می‌کنیم. بنابراین الگوریتم تمامی لیترال‌هایی که با داده‌های مثبت تناقض دارد را حذف می‌کند. (اگر دقت کنیم الگوریتم مشابه الگوریتم یادگیری مستطیل رفتار می‌کند).

اولین نکته‌ای که در تحلیل الگوریتم باید به آن توجه کرد این است که مجموعه‌ی لیترال‌های در h همواره شامل مجموعه لیترال‌های در مفهوم هدف c می‌باشد. زیرا با فرضیه‌ای شروع کردیم که شامل تمام لیترال‌ها بود و یک لیترال تنها در صورتی حذف می‌شد که باعث ۰ شدن یک مثال مثبت می‌شد؛ که به وضوح این لیترال نباید در c باشد. این واقعیت که لیترال‌های h همواره شامل c می‌باشد نتیجه می‌دهد که h هیچ‌گاه در مثال‌های منفی دچار خطا نمی‌شود (به عبارتی h خاص‌تر از c است).

بنابراین لیترال z را در نظر بگیرید که در h وجود دارد ولی در c نیست. بنابراین z باعث می‌شود که h در مثال‌های مثبتی از c که $z = ۰$ است دچار خطا شود. هم‌چنین توجه داشته باشید که تنها چنین مثال‌های مثبتی است که باعث می‌شود الگوریتم z را از h حذف کند. فرض کنید $p(z)$ رخ دادن چنین نمونه‌های تحت توزیع $\mathcal{D}$ باشد، به عبارتی:

$$p(z) = \Pr_{a \in \mathcal{D}}[c(a) = ۱ \wedge a \text{ در } z = ۰]$$

چون هر خطای h حداقل به خاطر وجود یک لیترال چون z در h رخ می‌دهد بنابراین کران بالای $\text{error}(h) \leq \sum_{z \in h} p(z)$ به دست می‌آید. می‌گوییم لیترال z بد است اگر $p(z) \geq \frac{\epsilon}{۲n}$ باشد. اگر h شامل هیچ لیترال بدی نباشد در این صورت $\text{error}(h) \leq \sum_{z \in h} p(z) \leq ۲n(\frac{\epsilon}{۲n}) = \epsilon$. بنابراین کافی است یک کران بالا برای احتمال رخ دادن لیترال‌های بد در h بیابیم.

برای لیترال z، احتمال این‌که این لیترال بعد از m بار فراخوانی $\text{EX}(c, \mathcal{D})$ در الگوریتم، از h حذف نشود حداکثر برابر با $(۱ - \frac{\epsilon}{۲n})^m$ خواهد بود، زیرا احتمال این‌که لیترال z با یک بار فراخوانی $\text{EX}(c, \mathcal{D})$ حذف شود برابر $p(z)$ است که برای لیترال‌های بد، حداقل $\frac{\epsilon}{۲n}$ بود. بنابراین از این نتیجه می‌شود احتمال وجود لیترال‌های بد در h بعد از m بار فراخوانی حداکثر $۲n(۱ - \frac{\epsilon}{۲n})^m$ خواهد بود.

بنابراین برای کامل کردن تحلیل، باید mای را پیدا کنیم که $۲n(۱ - \frac{\epsilon}{۲n})^m \leq \delta$ که $۱ - \delta$ پارامتر اطمینان است. با استفاده از نامساوی $۱ - x \leq e^{-x}$ نتیجه می‌شود که کافی است m را طوری بگیریم که رابطه‌ی $۲ne^{-m\epsilon/۲n} \leq \delta$ که نتیجه می‌شود $m \geq (\frac{۲n}{\epsilon})(\ln(۲n) + \ln(\frac{۱}{\delta}))$.

بنابراین اگر تعداد مثال‌ها را برابر کران فوق بگیریم در این صورت با احتمال حداقل $۱ - \delta$ گزاره‌ی عطفی h نسبت به c و $\mathcal{D}$ حداکثر خطای ϵ خواهد داشت. چون الگوریتم برای بررسی هر مثال زمانی خطی نیاز دارد، زمان اجرا دارای کران mn خواهد بود و بنابراین یک چندجمله‌ای برحسب n، $\frac{۱}{\epsilon}$ و $\frac{۱}{\delta}$ خواهد بود. □

سؤال. *اطلاعاتی که درباره‌ی این مساله وجود داشت به نسبت اطلاعات زیاد بود که به نظر می‌رسد به طور کامل از آن‌ها استفاده نشد. آیا می‌توان کران بهتری برای مساله یافت؟ یا نشان داد که کران بهتری وجود ندارد؟*

۴ ناکارآمدی یادگیری فرمول‌های DNF سه مولفه‌ای

در ادامه نشان می‌دهیم که کلاس نمایش گزاره‌های عطفی منطقی در حالت کلی مساله‌ای است که از لحاظ یادگیری PAC کارآمد نیست. به طور خاص نشان می‌دهیم کلاس گزاره‌های وصفی از سه گزاره‌ی بولی عطفی (که با نام فرم نرمال وصفی سه مولفه‌ای[۳۱]، DNF شناخته می‌شود) قابل یادگیری کارای PAC نیست حداقل تا زمانی که مسائل NP را نتوان در بدترین حالت به صورت کارا با الگوریتم‌های تصادفی حل کرد به عبارتی برای هر زبان[۳۲] A در NP الگوریتم تصادفی وجود ندارد که رشته‌ی α و پارامتر $\delta \in [۰,۱]$ را به عنوان ورودی بگیرید و با احتمال حداقل $۱ - \delta$ در زمان چندجمله‌ای بر حسب طول α

[۳۱] 3-term disjunctive normal form
[۳۲] language

و $\frac{1}{\delta}$ تشخیص دهد که α متعلق A است یا خیر؟ این نتیجه که یادگیری DNF با سه مولفه سخت است بر پایه‌ی این فرض صورت می‌گیرد که $\mathrm{RP} \neq \mathrm{NP}$.

کلاس نمایش $\mathcal{C}_n$ از فرمول‌های DNF سه مولفه‌ای، مجموعه‌ی تمام گزاره‌های فصلی به شکل $T_1 \vee T_2 \vee T_3$ است که هر T_i یک گزاره‌ی عطفی روی لیترال‌های از متغیرهای بولی $x_1, \ldots, x_n$ است. اندازه‌ی هر نمایش را مجموع تعداد لیترال‌هایی که در مولفه‌ها وجود دارد تعریف می‌کنیم (که همواره به وسیله‌ی یک چندجمله‌ای از تعداد بیت‌های موردنیاز رشته کران‌دار است اگر از کدگزاری استاندارد استفاده کنیم). بنابراین برای هر $c \in \mathcal{C}_n$ داریم $\mathrm{size}(c) \leq 6n$ چون هر مولفه حداکثر $2n$ لیترال دارد و سه مولفه هم داریم. بنابراین یک الگوریتم یادگیری کارا برای این مساله باید در زمان چندجمله‌ای برحسب n، $\frac{1}{\epsilon}$ و $\frac{1}{\delta}$ اجرا شود.

قضیه ۵. *با فرض $\mathrm{RP} \neq \mathrm{NP}$ کلاس نمایش فرمول‌های DNF با سه مولفه، قابل یادگیری کارا در مدل PAC نمی‌باشد.*

اثبات. ایده‌ی اصلی اثبات، کاهش یک زبان NP ـ کامل به مساله‌ی یادگیری PAC فرمول‌های DNF سه مولفه‌ای است. کاهش مساله باید نگاشتی کارا باشد که هر رشته‌ی α که می‌خواهیم عضویت آن را در زبان A بررسی کنیم، را به مجموعه‌ی S_α از مثال‌های برچسب‌دار، می‌نگارد. اندازه‌ی $|S_\alpha|$ باید به وسیله‌ی یک چندجمله‌ای از طول رشته $|\alpha|$ کران‌دار باشد. نشان می‌دهیم اگر الگوریتم یادگیری PAC، L برای فرمول DNF سه مولفه‌ای وجود داشته باشد، می‌توان L را روی S_α اجرا کرد و براساس نتیجه‌ی آن با احتمال بالایی مشخص کرد که α در A قرار دارد یا خیر.

نکته‌ی اصلی که در این نگاشتن α به S_α اتفاق بیفتد این است که $\alpha \in A$ اگر و تنها اگر S_α با مفهومی مانند $c \in \mathcal{C}$ سازگار[۳۳] باشد. منظور از سازگاری، عبارت است از:

تعریف ۶. *فرض کنید $S = \{< x_1, b_1 >, \ldots, < x_m, b_m >\}$ مجموعه‌ای از نمونه‌های برچسب‌دار باشد که $x_i \in X$ و $b_i \in \{0,1\}$. می‌گوییم مفهوم c روی X با S سازگار است (معادلا S با c سازگار است) اگر برای هر $1 \leq i \leq m$ داشته باشیم $c(x_i) = b_i$.*

قبل از این‌که به انتخاب زبان NP ـ کامل و نحوه‌ی نگاشتن α به S_α بپردازیم، ببینیم اگر این حکم برقرار باشد که $\alpha \in A$ اگر و تنها اگر S_α با مفهومی در $\mathcal{C}$ سازگار باشد، چه اتفاقی رخ می‌دهد. حال نشان می‌دهیم از الگوریتم یادگیری PAC، L برای $\mathcal{C}$ می‌توان استفاده کرد و در صورت وجود، مفهومی در $\mathcal{C}$ را یافت که با احتمال بالا با S_α سازگار باشد. به این ترتیب عمل می‌کنیم: پارامتر خطا را برابر $\epsilon = \frac{1}{2|S_\alpha|}$ قرار می‌دهیم که $|S_\alpha|$ تعداد اعضای S_α است و هر درخواستی که از L می‌آید را با یک انتخاب یک مثال برچسب‌دار $< x_i, b_i >$ با توزیع یکنواخت از S_α پاسخ می‌دهیم (توزیع $\mathcal{D}$ را توزیع یکنواخت در نظر می‌گیریم). در این حالت، با انتخابی که برای ϵ داشتیم تضمین می‌کنیم هر فرضیه‌ی h که خطای کمتر از ϵ داشته باشد باید با S_α سازگار باشد. زیرا اگر h حتی در یک نمونه از S_α دچار اشتباه شود، در این صورت نسبت به c و $\mathcal{D}$ حداقل دارای خطای $\epsilon > 2\epsilon = \frac{1}{|S_\alpha|}$. از طرف دیگر اگر هیچ مفهومی در $\mathcal{C}$ وجود نداشته باشد که با S_α سازگار باشد، L نمی‌تواند مفهومی بیابد. بنابراین خروجی L، در صورت وجود با احتمال $1 - \delta$ با S_α سازگار است.

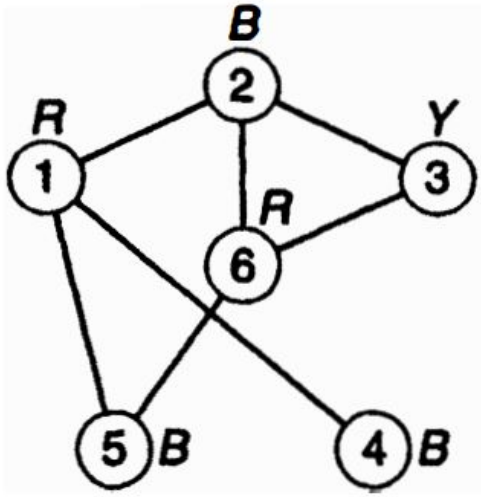
با ترکیب موضوع فوق با نگاشتی که α را به مجموعه‌ی S_α می‌نگارد، می‌توانیم (با احتمال حداقل $1 - \delta$) عضویت α در A را مشخص کنیم. بنابراین کافی است زبان NP ـ کامل A را به طور مناسب انتخاب کنیم و مجموعه مثال‌های برچسب‌دار S_α را برای $\alpha \in A$ بسازیم. برای نشان دادن ناکارامدی یادگیری فرمول‌های DNF سه مولفه‌ای از مساله‌ی سه رنگ‌پذیری گراف استفاده می‌کنیم:

مساله‌ی سه رنگ‌پذیری گراف: گراف بدون جهت $G = (V, E)$ با مجموعه راس‌های $V = \{1, \ldots, n\}$ و مجموعه رئوس $E \subset V \times V$ به عنوان ورودی داده شده است. آیا می‌توان راس‌های G را با سه رنگ متفاوت به گونه‌ای رنگ‌آمیزی کرد که اگر $(i, j) \in E$ باشد آن‌گاه رئوس i و j رنگ‌های متفاوتی داشته باشند؟

[۳۳] consistent

بنابراین باید نگاشتی از نمونه‌ی $G = (V, E)$ به مجموعه‌ی S_G از مثال‌های برچسب‌دار تعریف کنیم. S_G شامل دو مجموعه S_G^+ شامل مثال‌های با برچسب مثبت و S_G^- شامل مثال‌های با برچسب منفی است، بنابراین $S_G = S_G^+ \cup S_G^-$. برای هر $1 \le i \le n$، S_G^+ شامل مثال $< v(i), 1 >$ است که $v(i) \in \{0,1\}^n$ برداری است که مولفه‌ی iام آن ۰ و مابقی مولفه‌ها ۱ است. در واقع در این‌جا مثال‌ها، کدگذاری شده‌ی رئوس گراف G است. برای هر یال $(i, j) \in E$، مجموعه‌ی S_G^- شامل مثال

ای دیگر ۱ است.
ه رنگ‌آمیزی از G



Graph G

S_G^+	S_G^-
<011111,1>	<001111,0>
<101111,1>	<011011,0>
<110111,1>	<011101,0>
<111011,1>	<100111,0>
<111101,1>	<101110,0>
<111110,1>	<110110,0>
	<111100,0>

$T_R = x_2 \wedge x_3 \wedge x_4 \wedge x_5$

$T_B = x_1 \wedge x_3 \wedge x_6$

$T_Y = x_1 \wedge x_2 \wedge x_4 \wedge x_5 \wedge x_6$

شکل ۶: گراف G با یک رنگ‌آمیزی قابل قبول، نمونه‌های متناظر و گزاره‌های تعریف شده به وسیله‌ی رنگ‌آمیزی فوق

نشان می‌دهیم G سه رنگ‌پذیر است اگر و تنها اگر S_G با یک فرمول DNF سه مولفه‌ای سازگار باشد. ابتدا فرض کنید G سه رنگ‌پذیر باشد و یک سه رنگ‌آمیزی برای G در نظر بگیرید. فرض کنید R مجموعه‌ی تمام راس‌های گراف با رنگ قرمز باشد و T_R گزاره‌ی عطفی حاصل از تمام متغیرهای $x_1, \ldots, x_n$ باشد که اندیس‌شان در R قرار ندارد (تمام راس‌های با دو رنگ دیگر، شکل ۶ را ببینید). بنابراین برای هر $i \in R$، $v(i)$ باید در T_R صدق کند چون متغیر x_i در T_R نیست. بعلاوه هیچ $e(i, j) \in S_G^-$ نمی‌تواند در T_R صدق کند زیرا هر دوی i و j نمی‌توانند قرمز باشند و بنابراین یکی از x_i یا x_j در T_R قرار دارد. به همین ترتیب می‌توانیم T_B و T_Y را تعریف کنیم که هیچ‌کدام از مثال‌های منفی توسط آن‌ها پذیرفته نمی‌شود.

برای عکس مطلب فوق، فرض کنید فرمول $T_R \vee T_B \vee T_Y$ فرمولی سازگار با S_G باشد. یک رنگ‌آمیزی برای G به این شکل تعریف می‌کنیم: رنگ راس i را قرمز قرار می‌دهیم اگر $v(i)$ در T_R صدق کند، آبی قرار می‌دهیم اگر $v(i)$ در T_B صدق کند و آن را با زرد رنگ می‌کنیم اگر $v(i)$ در T_Y صدق کند (اگر $v(i)$ در بیش از یک شرط فوق صدق کند، شرط اول را در نظر می‌گیریم). از آن‌جایی که فرمول با S_G سازگار است، هر کدام از $v(i)$ها حداقل در یکی از مولفه‌های فرمول صدق می‌کند و بنابراین با روش فوق به هر راس یک رنگ نسبت داده‌ایم. کافی است نشان دهیم رنگ‌آمیزی فوق، یک سه رنگ‌آمیزی مجاز است. برای دیدن این مطلب، فرض کنید برای $i \neq j$ یک رنگ‌آمیزی یکسان، مثلا قرمز داشته باشیم. در این صورت $v(i)$ و $v(j)$ در T_R صدق می‌کنند. از آن‌جایی که iامین بیت $v(i)$ صفر و iامین بیت $v(j)$ برابر ۱ است، نتیجه می‌شود که هیچ‌کدام از x_i و $\bar{x}_i$ نمی‌تواند در T_R وجود داشته باشد. از طرفی چون $v(j)$ و $e(i, j)$ تنها در iامین بیت با هم تفاوت دارند اگر $v(j)$ در T_R صدق کند، آن‌گاه $e(i, j)$ نیز صدق می‌کند که نتیجه می‌شود $e(i, j) \notin S_G^-$ پس $(i, j) \notin E$. □

بنابراین دیدیم که فرمول‌های DNF سه مولفه‌ای تحت این فرض که مسائل NP ـ کامل با احتمال بالا در زمان چندجمله‌ای قابل حل نیستند ($\mathrm{RP} \neq \mathrm{NP}$)، قابل یادگیری کارای PAC نیست. با انجام تغییراتی تکنیکی در روش فوق می‌توان دید که

فرمول‌های DNF دو مولفه‌ای، و به طور کلی برای $k \geq ۲$ ثابت، فرمول‌های DNF با k مولفه قابل یادگیری کارای PAC نیست.

اما توجه داشته باشید که روش فوق بسیار متکی به تعریف یادگیری PAC و این واقعیت بود که فرضیه‌ی خروجی الگوریتم از همان کلاس نمایشی باشد که هدف انتخاب شده است. در بخش بعدی خواهیم دید که این واقعیت برای ناکارآمد بودن یادگیری فرمول‌های DNF سه مولفه‌ای مورد نیاز است و با تغییر این شرط نتایج دیگری به دست می‌آید که ما را به اصلاح مجدد تعریف سوق می‌دهد.

سؤال. *آیا مساله‌ی فوق الگوریتم یادگیری غیرکارای PAC دارد؟*

۵ استفاده از ۳-CNF برای جلوگیری از ناکارآمدی

در این بخش نشان می‌دهیم اگر الگوریتم یادگیری اجازه داشته باشد از نمایش‌های دیگری برای نمایش فرضیه استفاده کند، در این صورت کلاس فرمول‌های DNF سه مولفه‌ای به طور کارا قابل یادگیری است. این نتیجه با نتیجه‌ای که از قضیه‌ی ۵ حاصل شد، باعث می‌شود که بار دیگر تعریف را اصلاح کرده و تعریف نهایی را ارائه دهیم.

با استفاده از جبر بولی می‌دانیم که $\vee$ نسبت به $\wedge$ خاصیت توزیع‌پذیری دارد.به طور مثال برای متغیرهای بولی u, v, w, x می‌توان نوشت

$$(u \wedge v) \vee (w \wedge x) = (u \vee w) \wedge (u \vee x) \wedge (v \vee w) \wedge (v \vee x)$$

از این واقعیت می‌توانیم استفاده کنیم و فرمول‌های DNF سه مولفه‌ای روی متغیرهای $x_۱, \ldots, x_n$ را با فرم نرمال عطفی[۳۴]، CNF معادل بنویسیم به طوری که هر فصل[۳۵] شامل حداکثر ۳ لیترال است (که با آن فرمول 3-CNF می‌گوییم):

$$T_۱ \vee T_۲ \vee T_۳ \equiv \bigwedge_{u \in T_۱, v \in T_۲, w \in T_۳} (u \vee v \vee w)$$

که عطف روی تمام فصل‌هایی است که از هر مولفه یک لیترال دارند.

مساله‌ی یادگیری PAC فرمول‌های 3-CNF را می‌توان به مساله‌ی یادگیری گزاره‌های عطفی، که در حال حاضر الگوریتم یادگیری کارا برای آن داریم، کاهش داد. ایده‌ی اصلی به این شکل است: اوراکلی داده شده است که به طور تصادفی مثال‌هایی از یک فرمول 3-CNF ناشناخته را خروجی می‌دهد. یک راه ساده و کار وجود دارد که هر مثال مثبت یا منفی را به مثال مثبت یا منفی معادل آن از یک گزاره‌ی عطفی ناشناخته (با مجموعه متغیرهای بیشتر) تبدیل کرد. سپس این مثال‌ها را به الگوریتم یادگیری گزاره‌های عطفی که قبلا دیدیم، می‌دهیم. فرضیه‌ی خروجی از این الگوریتم را می‌توان به فرضیه‌ی مناسبی برای فرمول ناشناخته‌ی 3-CNF تبدیل کرد.

برای این‌که مثال‌ها را به نوع جدید تبدیل کنیم، فرمول 3-CNF را روی یک مجموعه‌ی بزرگتر از متغیرها به صورت یک گزاره‌ی عطفی می‌نویسیم. برای هر سه‌تایی از لیترال‌های u, v, w روی مجموعه متغیرهای اولیه، $x_۱, \ldots, x_n$، مجموعه متغیرهای جدید شامل متغیر $y_{u,v,w}$ است که مقدار آن به صورت $y_{u,v,w} = u \vee v \vee w$ به دست می‌آید. دقت کنید که اگر $u = v = w$ باشد آن‌گاه $y_{u,v,w} = u$ و بنابراین تمام متغیرهای اولیه در مجموعه‌ی جدید وجود دارند. هم‌چنین تعداد متغیرهای جدید از $(۲n)^۳ = O(n^۳)$ است.

برای هر گزاره‌ی $a \in \{۰,۱\}^n$ روی متغیرهای اولیه‌ی $x_۱, \ldots, x_n$، می‌توانیم در زمان $O(n^۳)$ گزاره‌ی معادل a' را روی متغیرهای جدید $\{y_{u,v,w}\}$ به دست آوریم. بعلاوه واضح است که هر فرمول 3-CNF، c روی متغیرهای $x_۱, \ldots, x_n$ معادل گزاره‌ی عطفی c' روی متغیرهای جدید است (تنها کافی است فصل $(u \vee v \vee w)$ را با متغیر معادل آن، $y_{u,v,w}$ در مجموعه‌ی جدید جایگزین کنیم). بنابراین می‌توانیم الگوریتم یادگیری گزاره‌های عطفی را روی مثال‌هایی که از تبدیل مثال‌های به دست

[۳۴] conjunctive normal form
[۳۵] clause

آمده از فرمول ناشناخته‌ی 3-CNF (که اوراکل در دسترس در اختیارمان قرار می‌دهد) اجرا کنیم. سپس می‌توانیم گزاره‌ی عطفی فرضیه، h' روی متغیرهای $y_{u,v,w}$ را با جایگزین کردن فصل $(u \vee v \vee w)$ با متغیر معادل $y_{u,v,w}$ به فرمول 3-CNF، h تبدیل کنیم.

در نهایت باید نشان دهیم اگر c فرمول 3-CNF هدف و $\mathcal{D}$ توزیع هدف روی $\{0,1\}^n$ باشد و c' و $\mathcal{D}'$ به ترتیب فرمول عطفی معادل روی متغیرهای $y_{u,v,w}$ و توزیع القا شده روی گزاره‌های a' نسبت به $y_{u,v,w}$ باشند در این صورت اگر h' نسبت به c' و $\mathcal{D}'$ خطای کمتر از ϵ داشته باشد، h نیز نسبت به c و $\mathcal{D}$ خطایی کمتر از ϵ خواهد داشت. به سادگی دیده می‌شود که نگاشتی که فرمول 3-CNF را به گزاره‌ی عطفی تبدیل می‌کرد، یک به یک است: اگر a_1 به a'_1 و a_2 به a'_2 نگاشته شود و $a_1 \neq a_2$ آن‌گاه $a'_1 \neq a'_2$. بنابراین هر بردار a' که h' و c' در آن تفاوت دارند، وارون یکتای a را دارد که h و c روی آن متفاوتند و a تحت $\mathcal{D}$ دقیقا وزن a' تحت توزیع $\mathcal{D}'$ را دارد. قابل ذکر است که الگوریتم برای هر توزیع به درستی عمل می‌کرد، بنابراین برای توزیع القایی از $\mathcal{D}$ نیز خروجی مناسبی خواهد داشت. بنابراین قضیه‌ی زیر ثابت شد:

قضیه ۷. *کلاس نمایش فرمول‌های 3-CNF قابل یادگیری کارای PAC است.*

از طرفی نشان دادیم که هر فرمول DNF با سه مولفه را می‌توان به صورت یک فرمول 3-CNF نمایش داد. بنابراین اگر اجازه داشته باشیم که فرضیه را به صورت یک فرمول 3-CNF نمایش دهیم می‌توان فرمول‌های DNF سه مولفه‌ای را به صورت کارای PAC یاد گرفت. به همین روش می‌توان نشان داد برای $k \geq 2$ که k در طول الگوریتم ثابت در نظر گرفته می‌شود، فرمول‌های DNF با k مولفه را می‌توان با فرمول‌های k-CNF یاد گرفت. این مطلب یکی از اصول مهم در نظریه‌ی یادگیری را نشان می‌دهد: حتی برای یک کلاس مفهوم در نظر گرفته شده که مفاهیم هدف از آن انتخاب شده‌اند، انتخاب نمایش برای فرضیه اغلب می‌تواند در الگوریتم‌های کارا و غیرکارا متفاوت باشد. بنابراین به گونه‌ای باید کلاس نمایش فرضیه را در تعریف بگنجانیم. پس تعریف را به صورت زیر اصلاح می‌کنیم:

تعریف ۸. مدل PAC، تعریف نهایی. *اگر $\mathcal{C}$ کلاس مفاهیم روی X و $\mathcal{H}$ یک کلاس نمایش روی X باشد، می‌گوییم $\mathcal{C}$ قابل یادگیری (کارا) PAC با استفاده از $\mathcal{H}$ است اگر شرایط تعریف اولیه یادگیری PAC برقرار باشد و به الگوریتم اجازه دهیم که فرضیه‌ی خروجی را از کلاس $\mathcal{H}$ ارائه دهد. در این‌جا فرض می‌کنیم $\mathcal{H}$ حداقل شامل $\mathcal{C}$ هست و بنابراین برای هر تابع در $\mathcal{C}$ نمایشی در $\mathcal{H}$ برای آن وجود دارد. به $\mathcal{H}$ کلاس فرضیه الگوریتم یادگیری PAC می‌گوییم.*

با توجه به بحث‌هایی که صورت گرفت، نمی‌خواهیم که $\mathcal{H}$ را با محدودیت‌های غیرضروری محدود کنیم هم‌چنین نمی‌خواهیم آن را بدون هیچ محدودیتی رها کنیم. به طور خاص، یکی از محدودیت‌های موجود زمان اجرای چندجمله‌ای برای الگوریتم یادگیری است و فرضیه‌ی خروجی توسط الگوریتم باید در زمان چندجمله‌ای قابل ارزیابی و نمایش باشد. این مطلب ما را به تعریف زیر رهنمون می‌کند:

تعریف ۹. *کلاس نمایش $\mathcal{H}$ در زمان چندجمله‌ای قابل ارزیابی[۳۶] است اگر الگوریتمی وجود داشته باشد که هر $x \in X_n$ و $h \in \mathcal{H}$ را به عنوان ورودی بگیرد و $h(x)$ را در زمان چندجمله‌ای برحسب n و $\text{size}(h)$ خروجی بدهد.*

در ادامه همواره فرض ما بر این است که الگوریتم یادگیری PAC از کلاس نمایشی برای فرضیه‌های استفاده می‌کند که قابل ارزیابی چندجمله‌ای است. بنابراین منظور از یادگیری کارای PAC برای کلاس مفهوم $\mathcal{C}$ یادگیری کارای PAC با استفاده از کلاس قابل ارزیابی چندجمله‌ای $\mathcal{H}$ است. حال می‌توانیم نتایجی که تاکنون به دست آمد را به صورت قضیه‌ی زیر بیان کنیم:

قضیه ۱۰. *کلاس نمایش فرمول‌های DNF یک مولفه‌ای (گزاره‌های عطفی) قابل یادگیری کارای PAC با استفاده از فرمول‌های DNF یک مولفه‌ای است. برای ثابت $k \geq 2$، کلاس نمایش فرمول‌های DNF با k مولفه به وسیله‌ی فرمول‌های DNF با k*

[۳۶] polynomially evaluatable

فرمول قابل یادگیری کارای PAC نیست (با فرض $\mathrm{NP} \neq \mathrm{RP}$)، اما به وسیله‌ی فرمول‌های $k - \mathrm{CNF}$ قابل یادگیری کارای PAC است.

۶ ضمیمه: مقدماتی از پیچیدگی محاسبه

همان‌طور که دیدیم یکی از ابزارهایی مورد نیاز در این نوشتار، پیچیدگی محاسبه است. در پیچیدگی محاسبه، به دنبال نظریه‌ای هستیم که بتوان توسط آن به مسائل یک "میزان سختی" نسبت داد و با این معیار آن‌ها را با هم مقایسه کرد. در این‌جا سعی می‌کنیم تعاریف و قضایای اصلی این مبحث را براساس [۱] و [۴] ارائه کنیم.

تعداد متناهی شیء را در نظر بگیرید که توسط دنباله‌ای دودویی متناهی کدگذاری شده‌اند. به این کدها، رشته[۳۷] می‌گوییم. برای عدد طبیعی n، مجموعه‌ی تمام دنباله‌های دودویی از طول n را با $\{۰,۱\}^n$ نمایش می‌دهیم و قرار می‌دهیم $\{۰,۱\}^* = \cup_{n\in\mathbb{N}}\{۰,۱\}^n$. برای $x \in \{۰,۱\}^*$ طول x را با $|x|$ نمایش می‌دهیم.

ابتدا باید منظورمان را از مساله مشخص کنیم. به چندتایی $P = (C, I, Q)$ یک مساله می‌گوییم که در آن I مجموعه رشته‌های ورودی مجاز برای مساله است، Q خانواده‌ی تمام رشته‌های ممکن از جواب‌هاست و C تعدادی کاراکتر یا رشته‌ی ثابت که در مقداری ثابت از حافظه قرار داده شده و در حل مساله با داشتن پارامترهای ورودی برنامه، می‌توان بارها از آن‌ها کمک گرفت. چون کار متداول ما نسبت دادن الگوریتم و برنامه به مسائل است گاهی Q، I و C را به خود الگوریتم‌ها یا مسائل نسبت می‌دهیم. یک عضو مجموعه‌ی I یا مصداق و یک عضو مجموعه‌ی Q را خروجی می‌نامیم. دو نوع مساله وجود دارد: تصمیم‌گیری و ساختی.

مثال ۱۱. *مسئله‌ی کوتاه‌ترین مسیر را برای گراف $G = (V, E)$ در نظر بگیرید. در این مساله I مجموعه‌ی تمام رشته‌هایی است که به نحوی که از قبل توافق کرده‌ایم، نمایش یک گراف و رئوس مبدا و مقصد می‌باشند و Q نیز شامل تمام دنباله‌هایی به شکل مسیرهای بین رئوس گراف‌هاست. حال به دو شکل می‌توان مساله را مطرح ساخت؛ نوع ساختی، "مطلوب است کوتاه‌ترین مسیری بین رئوس u و v از گراف G" و نوع تصمیم‌گیری، "آیا مسیری بین رئوس u و v از گراف G وجود دارد که طول آن حداکثر l باشد؟".*

یک عضو Q از نوع ساختی مساله را یک راه‌حل آن مساله می‌نامیم و مسائلی که جواب آن‌ها بله ـ خیر ($Q = \{\mathrm{Yes}, \mathrm{No}\}$) است را مسائل تصمیم‌گیری می‌گوییم. مسائل کلاس‌های معروف پیچیدگی از نوع بله‌ـ‌خیر هستند بنابراین در ادامه منظور ما از مساله، مساله‌ای از نوع تصمیم‌گیری است مگر خلاف آن ذکر شود.

تعریف ۱۲. (کلاس P) *رده‌ای از مسائل تصمیم‌گیری هستند که در زمان چندجمله‌ای حل‌پذیرند.*

مساله‌ی تصمیم‌گیری $S \subset \{۰,۱\}^*$ را حل‌پذیر در زمان چندجمله‌ای می‌گوییم اگر ماشین تورینگ[۳۸] M از زمان چندجمله‌ای وجود داشته باشد که $M(x) = ۱$ اگر و تنها اگر $x \in S$. (در واقع می‌توانی به جای ماشین تورینگ همان الگوریتم‌ها را در نظر گرفت.)

با وجود این‌که تعیین وجود مسیری بین رئوس u و v از گراف داده شده‌ی G که طول آن حداقل k باشد مساله‌ی ساده‌ای نیست، اگر با داشتن گراف G و عدد k، مسیری بین رئوس u و v به ما بدهند، به سرعت می‌توانیم بگوییم که این مسیر از طول دست‌کم k هست یا نه. برنامه‌ای که این وظیفه را انجام می‌دهد، یعنی تعیین می‌کند مصداق داده شده از نوع ساختی مسئله شرایط نوع تصمیم‌گیری را ارضا می‌کند یا خیر، تصدیق‌کننده می‌گوییم.

[۳۷] string
[۳۸] Turing machine

تعریف ۱۳. (کلاس NP) *رده‌ای از مسائل تصمیم‌گیری که طول رشته راه‌حل‌های آن‌ها نسبت به طول رشته‌ی ورودی از مرتبه‌ی چندجمله‌ای است و تصدیق‌کننده‌ای برای راه‌حل‌های آن‌ها وجود دارد که در زمان چندجمله‌ای نسبت به طول رشته راه‌حل کار می‌کند.*

با توجه به تعاریف واضح است که $\mathrm{P} \subset \mathrm{NP}$ اما برقراری عکس این موضوع یکی از بزرگترین مسائل حل نشده‌ی علوم کامپیوتر محسوب می‌شود. رده‌ی دیگری از مسائل که مکمل مسائل NP محسوب می‌شود نیز قابل تعریف است. این رده را با coNP نمایش می‌دهیم. در واقع داریم $\mathrm{coNP} := \{\{0,1\}^* \backslash S : S \in \mathrm{NP}\}$. با توجه به تعریف آیا $\mathrm{NP} = \mathrm{coNP}$؟ این نیز یکی دیگر از مسائل حل نشده در پیچیدگی محاسبه است.

تاکنون معیاری برای سختی یک مساله ارائه نکرده‌ایم. در ادامه ابزاری معرفی می‌کنیم که به کمک آن می‌توانیم نشان دهیم که یک مساله تصمیم‌گیری از مساله‌ی تصمیم‌گیری دیگر سخت‌تر است. فرض کنید که مساله‌ی A یک مساله تصمیم‌گیری است که می‌خواهیم آن را در زمان چندجمله‌ای حل کنیم. هم‌چنین فرض کنید B یک مساله‌ی تصمیم‌گیری است که الگوریتم با زمان چندجمله‌ای دارد. در نهایت فرض کنید یک رویه داریم که هر مصداق α از مساله‌ی A را به یک مصداق β از مساله‌ی B تبدیل می‌کند که خواص زیر را دارد:

- این عملیات تبدیل در زمان چندجمله‌ای از طول مصداق α انجام می‌گیرد.
- جواب هر دو مصداق دقیقا یکی باشد. یعنی جواب α بلی است اگر و فقط اگر جواب β بلی باشد.

به چنین رویه‌ای، یک الگوریتم کاهش چندجمله‌ای می‌گوییم و این رویه به صورت زیر یک راه‌حل از زمان چندجمله‌ای برای مساله‌ی A ارائه می‌کند:

- مصداق α از مساله‌ی A را به وسیله‌ی رویه کاهش در زمان چندجمله‌ای به مصداق β از مساله‌ی B تبدیل می‌کنیم.
- الگوریتم چندجمله‌ای که برای مساله‌ی B داریم را بر روی مصداق β اجرا می‌کنیم.
- جواب β را به عنوان جواب α برمی‌گردانیم.

نمادی که برای کاهش مساله A به مساله B به کار می‌رود $A \leq B$ است.

با تعریفی که از کاهش چندجمله‌ای ارائه شد، می‌توانیم مفهومی را معرفی کنیم که می‌توان با آن مفهوم سخت‌تر بودن یک مساله از مساله‌ی دیگر را تعریف کرد. یعنی به عنوان مثال وقتی می‌نویسیم $L_1 \leq L_2$ به این مفهوم است که مساله‌ی L_1 با یک کاهش چندجمله‌ای به مساله‌ی L_2 کاهیده می‌شود و هر مصداق مساله‌ی L_1 با یک الگوریتم چندجمله‌ای به یک مصداق مساله‌ی L_2 تبدیل می‌شود. به طور کلی به تفاوت زمان‌های چندجمله‌ی و کمتر از چندجمله‌ای بین الگوریتم‌ها اهمیتی داده نمی‌شود. بنابراین وقتی $L_1 \leq L_2$، چون برای حل L_2 کافی است پیش از اجرای الگوریتم L_1 یک رویه‌ی چندجمله‌ای اجرا کنیم، نتیجه می‌گیریم "L_2 حداکثر به سختی L_1 است"، یعنی اضافه شدن یک رویه‌ی چندجمله‌ای برای ما به معنی "اکیدا سخت‌تر" شدن مساله نیست.

حال می‌توانیم رده‌ی مسائل NP-Complete را تعریف کنیم. مساله‌ی L را NP-Complete می‌گویند اگر

- L یک مساله NP باشد.
- برای هر مساله‌ی L' در NP یک کاهش از L' به L وجود داشته باشد.

اگر مساله‌ی L شرط اول را نداشته و شرط دوم را بپذیرد، به آن یک مساله‌ی NP-Hard می‌گویند.

قضیه ۱۴. *اگر یک مساله‌ی NPC در رده‌ی P قرار بگیرد آنگاه $\mathrm{P} = \mathrm{NP}$. معادلا اگر یک مساله NP وجود داشته باشد که جز رده‌ی P نباشد، آنگاه هیچ مساله‌ی NPC در P قرار ندارد.*

مثال ۱۵. (مساله SAT) *ورودی مساله یک فرمول به شکل فرم نرمال عطفی، CNF و خواسته‌ی آن این است که آیا این فرمول ارضا شدنی است یا خیر. در مساله‌ی 3-SAT ورودی فرمول‌های 3-CNF است. این مساله جزو رده‌ی NPC محسوب می‌شود.*

تعریف ۱۶. (زمان نمایی) *رده‌ای از مسائل است که برای حل آن‌ها الگوریتمی وجود دارد که زمان اجرای آن از $O(۲^{n^k})$ است که در آن n طول ورودی و k یک عدد طبیعی ثابت است. این کلاس را با EXP-Time نشان می‌دهیم. به عنوان مثال SAT یکی از این مسائل است.*

قضیه ۱۷. *EXP-Time⊂NP.*

رده‌های دیگری از کلاس‌های پیچیدگی وجود دارند که به دسته‌بندی الگوریتم‌های تصادفی می‌پردازد.

تعریف ۱۸. (کلاس RP) *زبان L در کلاس RP قرار دارد اگر الگوریتم تصادفی A برای آن وجود داشته باشد به طوری که:*

- *اگر $x \notin L$ در این صورت A با احتمال ۱ این موضوع را تشخیص دهد.*
- *اگر $x \in L$ در این صورت A با احتمال حداقل $\frac{۱}{۲}$ این موضوع را به درستی تشخیص دهد.*

با توجه به تعریف، اگر الگوریتم فوق را روی یک ورودی به تعداد کافی انجام دهیم با هر دقتی می‌توان به درستی در مورد ورودی تصمیم‌گیری کرد. به عبارتی اگر الگوریتم را t بار انجام دهیم احتمال خطا حداکثر $\frac{۱}{۲^t}$ خواهد بود. هم‌چنین واضح است که داریم NP⊂RP⊂P.

۷ ضمیمه: چند نامساوی احتمالاتی

قضیه ۱۹. *برای هر $x \in \mathbb{R}$ نامساوی زیر برقرار است:*

$$۱ + x < e^x$$

قضیه ۲۰. (نامساوی مارکوف) *فرض کنید x یک متغیر تصادفی مثبت باشد در این صورت داریم:*

$$\Pr[x \geq \alpha] \leq \frac{E[x]}{\alpha}$$

قضیه ۲۱. (کران چرنوف‌ـ هوفینگ) *فرض کنید $x_۱, \ldots, x_n$ متغیرهای تصادفی کران‌دار و مستقل باشند به طوری که $x_i \in [۰, ۱]$ و $E[x_i] = p$. اگر $\hat{p} = \frac{۱}{n}\sum_{i=۱}^{n} x_i$ آن‌گاه داریم:*

$$\Pr[|\hat{p} - p| > \epsilon] \leq ۲e^{-۲n\epsilon^۲}$$

۸ ضمیمه: یادگیری و رمزنگاری

همان‌طور که دیدیم فرمول‌های DNF سه مولفه‌ای را نمی‌توان به وسیله‌ی فرمول‌های DNF سه مولفه‌ای به طور کارا یاد گرفت با این حال توانستیم با کمک 3-CNF آن‌ها را در مدل PAC یاد بگیریم. بنابراین سختی مساله‌ی فوق به نمایش وابسته بود. برای نشان دادن سختی یک مساله‌ی یادگیری، مستقل از نمایش[۳۹] آن باید از فرضیات رمزنگاری استفاده کنیم. در این‌جا به طور مختصر به این موضوع می‌پردازیم. به طور دقیق‌تر، نشان می‌دهیم که یادگیری PAC باعث شکستن سیستم رمزنگاری RSA می‌شود.

[۳۹] representation-independent

در واقع دقت داشته باشید که در یک الگوریتم رمزنگاری ما به الگوریتم رمز کردن دسترسی داریم بنابراین می‌توانیم هر تعداد مثال بخواهیم تولید کنیم. در واقع اگر f تابع رمزنگاری باشد می‌توان مثال‌های $< f(x), x >$ را برای الگوریتم رمزگشایی در نظر گرفت، بنابراین اگر بتوان الگوریتم یادگیری ارائه داد که از روی مثال‌های ساخته شده بتواند مدار رمزگشایی را یاد بگیرد می‌توان سیستم رمزنگاری را شکست. بنابراین اگر f تابعی باشد که وارون آن به راحتی قابل یادگیری باشد، این کار ممکن است. این مطلب تعریف زیر را مطرح می‌کند:

تعریف ۲۲. (تابع یک‌طرفه[۴۰]) *به تابعی گفته می‌شود که برای هر ورودی، خروجی تابع به راحتی قابل محاسبه است، اما به دست آوردن تصویر وارون خروجی متناظر با یک ورودی تصادفی دشوار است. (در زمان چندجمله‌ای این کار ممکن نیست).*

تعریف فوق را می‌توان به این شکل نیز بیان کرد: تابع $f : \{0,1\}^* \to \{0,1\}^*$ یک‌طرفه است هرگاه f توسط یک الگوریتم چندجمله‌ای قابل محاسبه باشد. برای هر الگوریتم تصادفی که در زمان چندجمله‌ای از طول ورودی اجرا می‌شود و برای هر چندجمله‌ای $p(n)$ و برای مقادیر بزرگ n داشته باشیم:

$$\Pr[f(A(f(x))) = f(x)] < \frac{1}{p(x)}$$

که در آن x به طور یکنواخت روی $\{0,1\}^n$ انتخاب شده است. توجه داشته باشید که طبق تعریف، به دست آوردن تصویر وارون یک عنصر باید در حالت میانگین سخت باشد و لزومی ندارد که در بدترین حالت هم سخت باشد.[۴۱]

در رمزنگاری، نمونه‌هایی از چنین توابعی معرفی می‌شود (با فرض $\mathrm{NP} \neq \mathrm{P}$) که RSA یکی از آن‌ها است. اما قضیه‌ی زیر است که این توابع را چنین مهم جلوه می‌دهد:

قضیه ۲۳. *تابع یک‌طرفه وجود دارد اگر و تنها اگر مولد شبه‌تصادفی[۴۲] وجود داشته باشد.*

مولدهای شبه‌تصادفی به گونه‌ای عمل می‌کنند که نمی‌توان بین خروجی آن و یک تابع تصادفی تفاوتی قائل شد. در واقع مولدهای شبه‌تصادفی توابعی هستند که روی یک رشته اعمال شده و یک جایگشت از آن رشته را خروجی می‌دهند به گونه‌ای که نمی‌توان بین خروجی داده شده و توزیع یکنواخت تفاوتی قائل شد. این خاصیت باعث می‌شود که این توابع در رمزنگاری مورد استفاده قرار گیرد و به همین خاطر به آن‌ها مولد امن اعداد شبه تصادفی[۴۳] نیز می‌گویند.

با توجه به مطالب گفته شده، فرض کنید یک تابع مولد شبه‌تصادفی به عنوان مفهوم c داده شده است. بنابراین الگوریتم یادگیری باید فرضیه‌ای ارائه دهد که بتواند خروجی بعدی c را به خوبی تشخیص دهد. اما با توجه به تعریف مولد شبه‌تصادفی، این کار با محاسبات ساده ممکن نیست.

اگر به جای استفاده از توابع یک‌طرفه، توابعی را مورد استفاده قرار دهیم که در حالت عادی محاسبه‌ی وارون آن سخت باشد اما با داشتن اطلاعات اضافی بتوان در زمان مناسبی برای ورودی y مقدار وارون آن را به دست آورد چه اتفاقی رخ خواهد داد؟ به چنین توابعی، توابع دریچه[۴۴] می‌گویند. به طور مثال فرض کنید حاصل‌ضرب دو عدد اول داده شده است و شما باید دو عدد اول را بیابید. به دست آوردن این دو عدد اول به سادگی امکان‌پذیر نیست اما اگر به عنوان اطلاعات اضافی، یکی از دو عدد اول به نیز داده شود، محاسبه‌ی عدد دوم به سادگی صورت می‌گیرد. بنابراین با داشتن اطلاعات اضافی، یادگیری چنین توابعی چندان سخت به نظر نمی‌رسد.

به طور معمول برای این‌که نشان دهند یک مساله قابل یادگیری کارای PAC نیست از فرضیاتی که در رمزنگاری وجود دارد، همان‌طور که دیدیم، استفاده می‌کنند.

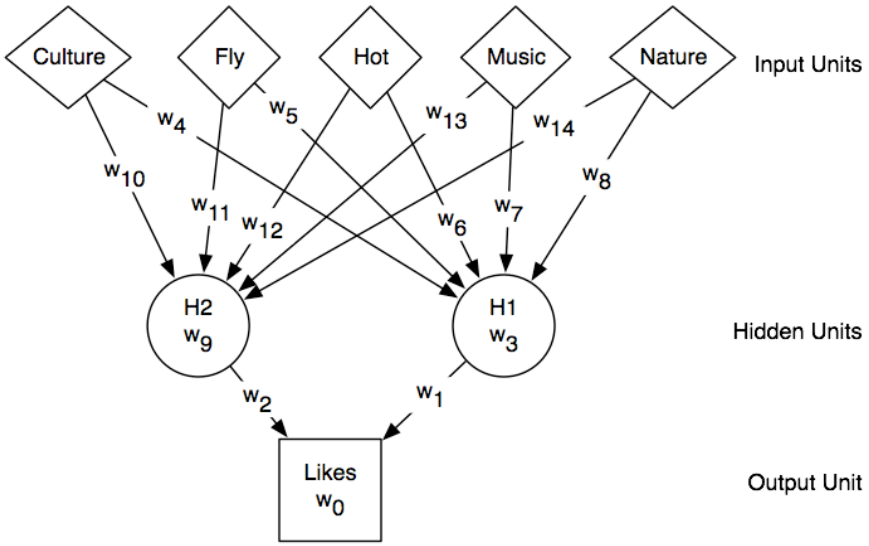
---

[۴۰] One-way function
[۴۱] وی‌کی‌پدیا
[۴۲] Pseudorandom generator
[۴۳] Cryptographically secure pseudorandom number generator
[۴۴] Trapdoor function

۹ ضمیمه: چند مساله‌ی ناکارآمد

در این بخش تعدادی از مسائل که قابل یادگیری کارای PAC نیستند را فهرست‌وار می‌آوریم.

مساله ۱. شبکه‌ی عصبی ۳ـ گره‌ای یکی از مسائل سخت در یادگیری PAC است. شکل ۷ یک شبکه‌ی عصبی ۳ ـ گره‌ای با ۵ ورودی بولی را نمایش می‌دهد.



شکل ۷: یک شبکه‌ی عصبی ۳ـ گره‌ای. یادگیری این شبکه‌ها از مسائل سخت است.

مساله ۲. یادگیری کوچکترین اتوماتای قطعی برای یک مجموعه‌ی مشخص و متناهی از دیگر مسائل سخت یادگیری محسوب می‌شود. حتی اگر مساله را تغییر داده و اتوماتای خروجی نزدیک به کوچکترین را هم بخواهیم یاد بگیریم، جزء مسائل سخت محسوب می‌شود.

مساله ۳. اگر در بازی یادگیری مستطیل از شکل‌های پیچیده‌تری استفاده شود، در این صورت مساله قابل یادگیری کارای PAC نخواهد بود.

نکته‌ی آخر این‌که یکی از مهم‌ترین مسائل حل نشده در مدل یادگیری PAC این است که آیا فرمول‌های DNF قابل یادگیری کارا است یا خیر؟

مراجع

[۱] Goldreich, Oded and Wigderson, Avi. *Computational Complexity*. October 3, 2004.

[۲] Valiant, L. G. *A Theory of the Learnable*. Communications of the ACM, Volume 27, 1984.

[۳] Kearns, J. Michael and Vazirani, V. Umesh *An Introduction to Computational Learning Theory*. The MIT Press, 1994.

[۴] رونق، پویا و ملکی، سعید. مقدمه‌ای بر نظریه‌ی پیچیدگی محاسبه. مهر ۱۳۸۷