تمرین ۱. فرض کنید $f(.)$ تابعی با خروجی اعداد صحیح باشد هم‌چنین فرض کنید هیچ الگوریتم تصادفی وجود ندارد که گراف G و پارامتر $0 < \delta \leq 1$ را به عنوان ورودی بگیرد، در زمان چندجمله‌ای بر حسب اندازه‌ی G و $\frac{1}{\delta}$ اجرا شود و با احتمال $1 - \delta$ خروجی آن به این شکل باشد: اگر G قابل رنگ‌آمیزی با k رنگ نباشد بگوید "نه" و در غیر این صورت یک $f(k)$ رنگ‌آمیزی از G ارائه دهد. با این فرضیات نشان دهید که برای $k \geq 3$، فرمول‌های *DNF* با k مولفه قابل یادگیری کارای *PAC* با فرمول‌های *DNF* با $f(k)$ مولفه نیستند.

پاسخ ۱. درست همانند اثباتی که برای فرمول‌های *DNF* سه مولفه‌ای انجام شد، عمل می‌کنیم. بنابراین باید نشان دهیم که S_G معرفی شده سازگار است اگر و تنها اگر گراف G قابل رنگ‌آمیزی با k رنگ باشد و بتوان یک $f(k)$-رنگ‌آمیزی برای آن ارائه داد. ابتدا فرض کنید G یک گراف k-رنگ‌پذیر باشد و یک $f(k)$-رنگ‌آمیزی برای آن ارائه شده است. بنابراین فرض می‌کنیم راس‌ها را با $C_1, \ldots, C_{f(k)}$ رنگ کرده‌ایم و یک رنگ‌آمیزی مجاز داریم. حال یک فرمول با $f(k)$ مولفه و سازگار با S_G براساس این رنگ‌آمیزی ارائه می‌دهیم. مولفه‌ی iام، T_i را عطف حاصل از تمام راس‌هایی که رنگ مخالف C_i دارند در نظر بگیرید و فرمول موردنظر را قرار دهید $T = T_1 \vee \cdots \vee T_{f(k)}$. این فرمول با S_G سازگار است زیرا اول راس v را در نظر بگیرید، این راس با یکی از رنگ‌ها مثلا C_j رنگ شده است بنابراین گزاره‌ی متناظر با آن در مولفه‌ی T_j صدق می‌کند پس خروجی T برای این مثال درست خواهد بود. هم‌چنین فرض کنید e یک یال از گراف باشد که دو راس v_i و v_j، $i \neq j$ را به یکدیگر وصل می‌کند. چون $k \geq 3$ بنابراین این دو راس باید رنگ‌های متفاوتی داشته باشند. بنابراین هر کدام از مولفه‌های فرمول را در نظر بگیریم، حداقل یکی از متغیرهای x_i یا x_j را دارد، بنابراین گزاره‌ی متناظر با e را صفر می‌کند، بنابراین یک مثال منفی برای T خواهد بود. پس T با S_G سازگار است.

برعکس فرض کنید S_G با فرمول *DNF*، $f(k)$-مولفه‌ای T سازگار باشد. راس دلخواه v از G را در نظر بگیرید. طبق تعریف S_G، گزاره‌ی معادل با v یک نمونه‌ی مثبت است پس باید در T صدق کند. بنابراین حداقل یکی از مولفه‌های T وجود دارد که v در آن صدق می‌کند. رنگ v را برابر رنگ متناظر با یکی از این مولفه‌ها قرار می‌دهیم. باید نشان دهیم که این رنگ‌آمیزی، یک رنگ‌آمیزی مجاز است. این نیز واضح است (همانند حالت ۳-رنگ‌آمیزی).

فرض کنید که یک الگوریتم کارای *PAC* وجود دارد که به کمک آن می‌توان فرمول‌های *DNF* با k مولفه را به کمک فرمول‌های *DNF* با $f(k)$ مولفه یاد گرفت. گراف دلخواه G را در نظر بگیرید. مجموعه‌ی S_G را در زمان چندجمله‌ای می‌توان ساخت. اگر الگوریتم فوق را با توزیع یکنواخت روی S_G انجام دهیم در زمان چندجمله‌ای با احتمال بالایی می‌توان فرمول *DNF* سازگار با S_G (در صورت وجود) یافت، پس می‌توان یک رنگ‌آمیزی برای گراف G ارائه کرد که این با فرض این‌که هیچ الگوریتم تقریبی از زمان چندجمله‌ای برای این کار وجود ندارد، تناقض دارد.

تمرین ۲. نوع دیگری از مدل *PAC* با دو اوراکل به این شکل است: برای مفهوم $c \in \mathcal{C}$ توزیع‌های دلخواه و جدا از هم $\mathcal{D}_c^+$ روی نمونه‌های مثبت و $\mathcal{D}_c^-$ روی نمونه‌های منفی وجود دارد. الگوریتم یادگیری به اوراکل‌های دسترسی $\mathrm{EX}(c, \mathcal{D}_c^+)$ و $\mathrm{EX}(c, \mathcal{D}_c^-)$ دسترسی دارد که در واحد زمان به ترتیب یک نمونه‌ی تصادفی مثبت و منفی را خروجی می‌دهند. برای پارامتر خطای ϵ، الگوریتم یادگیری باید فرضیه‌ای ارائه دهد که $\Pr_{x \in \mathcal{D}_c^+}[h(x) = 0] \leq \epsilon$ و $\Pr_{x \in \mathcal{D}_c^-}[h(x) = 1] \leq \epsilon$.

فرض کنید $\mathcal{C}$ یک کلاس مفهوم دلخواه و $\mathcal{H}$ کلاس نمایش باشد و h_0 و h_1 به ترتیب نمایش تابع‌های ثابت ۰ و ۱ باشند. ثابت کنید $\mathcal{C}$ با استفاده از کلاس نمایش $\mathcal{H}$ قابل یادگیری کارای *PAC* است اگر و تنها اگر در مدل جدید با استفاده از کلاس نمایش $\mathcal{H} \cup \{h_0, h_1\}$ قابل یادگیری *PAC* باشد.

پاسخ ۲. ابتدا فرض کنید مساله قابل یادگیری کارای *PAC* با تعریف اول (یک اوراکل) باشد. باید الگوریتمی با تعریف دوم ارائه کنیم. ابتدا در نظر داشته باشید که با یک درخواست به هر کدام از اوراکل‌های $\mathrm{EX}(c, \mathcal{D}_c^+)$ یا $\mathrm{EX}(c, \mathcal{D}_c^-)$ می‌توان فهمید که آیا یکی از دو مجموعه‌ی نمونه‌های مثبت و یا منفی تهی است یا خیر که در صورت تهی بودن می‌توان از نمایش‌های h_0 و h_1 استفاده کرد. بنابراین فرض می‌کنیم که هیچ‌کدام از دو مجموعه تهی نباشد. توزیع $\mathcal{D}$ را به صورت زیر تعریف می‌کنیم:

$$\mathcal{D} = \frac{1}{2}(\mathcal{D}_c^+ + \mathcal{D}_c^-)$$

با توجه به این‌که توزیع‌های $\mathcal{D}_c^+$ و $\mathcal{D}_c^-$ توزیع‌های احتمال هستند، توزیع $\mathcal{D}$ نیز یک توزیع احتمال ارائه می‌دهد. اگر الگوریتم را روی این توزیع و با پارامترهای ϵ و δ اجرا کنیم با احتمال $1 - \delta$ فرضیه‌ی $h \in \mathcal{H}$ را خروجی می‌دهد به طوری که داریم

$$\mathrm{Pr}_{x \sim \mathcal{D}}[c(x) \neq h(x)] \leq \epsilon$$

که نتیجه می‌دهد

$$\mathrm{Pr}_{x \in \mathcal{D}_c^+}[h(x) = 0] \leq \mathrm{Pr}_{x \sim \mathcal{D}}[c(x) \neq h(x)] \leq \epsilon$$

$$\mathrm{Pr}_{x \in \mathcal{D}_c^-}[h(x) = 1] \leq \mathrm{Pr}_{x \sim \mathcal{D}}[c(x) \neq h(x)] \leq \epsilon$$

بنابراین h در تعریف دوم فرضیه‌ی مطلوب است. بنابراین تنها کافی است توزیع $\mathcal{D}$ را شبیه سازی کنیم که با توجه به تعریف آن، از یک توزیع یکنواخت روی $\{-1, 1\}$ و سپس استفاده از توزیع متناظر $\mathcal{D}_c^+$ یا $\mathcal{D}_c^-$ حاصل می‌شود.

برعکس فرض کنید یک الگوریتم با تعریف دوم داریم و می‌خواهیم الگوریتمی سازگار با تعریف اول برای توزیع $\mathcal{D}$ و پارامترهای ϵ و δ ارائه دهیم. توزیع $\mathcal{D}$ را می‌توان به صورت ترکیب دو توزیع $\mathcal{D} = \mathcal{D}_c^+ + \mathcal{D}_c^-$ نوشت که به ترتیب نمونه‌های مثبت و منفی را در بر می‌گیرند. فرض کنید یک نمونه‌ی مثبت از اوراکل $\mathrm{EX}(c, \mathcal{D})$ می‌خواهیم به دست آوریم. احتمال این‌که بعد از m بار صدا زدن اوراکل هنوز نمونه‌ی مثبتی به دست نیامده باشد را حساب می‌کنیم. فرض کنید وزن مجموعه‌ی نقاط مثبت تحت توزیع $\mathcal{D}$ برابر α باشد. بنابراین احتمال این‌که با هر بار صدا زدن اوراکل نمونه‌ی منفی دریافت کنیم برابر خواهد بود با $(1 - \alpha)$ پس بعد از m بار فراخوانی با احتمال $(1 - \alpha)^m$ هنوز هیچ نمونه‌ی مثبتی نخواهیم داشت. اگر $\alpha > \epsilon$ باشد می‌خواهیم که خطا کمتر از δ شود. بنابراین داریم:

$$(1 - \alpha)^m \leq e^{-\alpha m} \leq e^{-\epsilon m} \leq \delta \Rightarrow m \geq \frac{1}{\epsilon} \log \frac{1}{\delta}$$

بنابراین اگر با $\frac{1}{\epsilon} \log \frac{1}{\delta}$ فراخوانی اوراکل هیچ نتیجه‌ی مثبتی حاصل نشود می‌توان فرض کرد که مجموعه‌ی مقادیر مثبت تهی است در این صورت با احتمال $1 - \delta$ حداکثر خطای ϵ خواهیم داشت. به همین ترتیب برای مقادیر منفی می‌توان چنین نتیجه‌ای گرفت.

الگوریتم اولیه که با دو اوراکل کار می‌کند را L می‌نامیم. الگوریتم جدید را به این شکل ارائه می‌دهیم: در هر مرحله که L به یکی از دو اوراکل درخواست می‌دهد که یک نمونه جدید بگیرد، مثلا به اوراکل $\mathrm{EX}(c, \mathcal{D}_c^+)$ در خواست می‌دهد، از اوراکل $\mathrm{EX}(c, \mathcal{D})$ یک نمونه می‌گیریم اگر نمونه مثبت بود آن را برمی‌گردانیم، و اگر منفی بود دوباره به اوراکل درخواست می‌دهیم. این کار را حداکثر $\frac{1}{\epsilon} \log \frac{1}{\delta}$ بار انجام می‌دهیم. اگر بعد از این مرحله، هیچ نمونه‌ی مثبتی دریافت نشده بود و در مرحله‌ی اول بود (یعنی اولین بار بود که L درخواست نمونه‌ی مثبت داده بود) می‌گوییم که نمونه‌های مثبت تهی است و اگر مرحله‌ی اول نبود و قبلا اوراکل درخواست نمونه مثبت داده بود و دریافت کرده بود، آخرین نمونه‌ی مثبت را برمی‌گردانیم. برای نمونه‌های منفی نیز به همین منوال عمل می‌کنیم. به این ترتیب الگوریتم L با احتمال $1 - \delta$ یک فرضیه‌ی h را خروجی می‌دهد که خطای آن روی نمونه‌های مثبت و منفی ϵ است. بنابراین خطای کل حداکثر 2ϵ خواهد بود. چون الگوریتم برای هر ϵ دلخواه به درستی عمل می‌کند اگر از ابتدا $\epsilon/2$ را در نظر می‌گرفتیم، خطای نهایی برابر ϵ می‌شد.

در نهایت الگوریتم فوق در زمان چندجمله‌ای اجرا خواهد شد. چون L در زمان چندجمله‌ای اجرا خواهد شد و الگوریتم جدید در هر مرحله از L حداکثر $\frac{1}{\epsilon} \log \frac{1}{\delta}$ کار انجام می‌دهد که چندجمله‌ای است بنابراین در نهایت حاصل‌ضرب دو چندجمله‌ای زمان اجرای الگوریتم خواهد بود که خود یک چندجمله‌ای است.

تمرین ۳. در تعریف دومی که برای مدل *PAC* ارائه شده، به الگوریتم یادگیری اجازه داده می‌شد که در زمان چندجمله‌ای برحسب n و $\mathrm{size}(c)$ نیز باشد و $\mathrm{size}(c)$ را هم به عنوان ورودی بگیرد. ثابت کنید این ورودی در واقع مورد نیاز نیست: اگر الگوریتم یادگیری کارای *PAC* وجود داشته باشد که $\mathrm{size}(c)$ را به عنوان ورودی می‌گیرد آن‌گاه الگوریتم یادگیری کارای *PAC* برای $\mathcal{C}$ بدون ورودی $\mathrm{size}(c)$ نیز وجود دارد.

پاسخ این تمرین براساس مراجع [۲] و [۱] به دست آمده است.

پاسخ ۳. با توجه به این‌که اطلاعی از اندازه‌ی مفهوم نداریم، باید الگوریتم موجود را برای اندازه‌های مختلفی از مفهوم اجرا کنیم تا نتیجه‌ی مطلوب حاصل شود. بنابراین باید معیاری برای سنجیدن فرضیه‌های خروجی الگوریتم داشته باشیم. بنابراین $\epsilon > 0$، $\delta > 0$ و $i \geq 1$ را دلخواه بگیرید و تعریف کنید:

$$m = [\frac{32}{\epsilon}(i \log 2 + \log \frac{2}{\delta})]$$

فرض کنید S مجموعه‌ای شامل m عضو است که از اوراکل $\mathrm{EX}(c, \mathcal{D})$ به دست آمده است. فرضیه‌ی h را سازگار می‌گوییم اگر خطای آن روی مجموعه‌ی S حداکثر $\frac{3}{4}\epsilon$ باشد (یعنی حداکثر در تعداد $\frac{3}{4}\epsilon$ نمونه از مجموعه‌ی S، h با c مخالف باشد) در غیر این صورت می‌گوییم h ناسازگار است.

ابتدا واقعیت‌هایی را در مورد سازگاری و ناسازگاری h با تعریف فوق به دست می‌آوریم. فرض کنید خطای h روی کل فضای مورد بحث (خطای واقعی)، بزرگتر یا مساوی با ϵ باشد، در این صورت در مورد سازگاری h نسبت به توزیع $\mathcal{D}$ داریم:

$$\Pr[h \text{ سازگار باشد. }] = \Pr[S \text{ روی } h \text{ خطای } \leq \frac{3}{4}\epsilon]$$

از طرفی طبق فرض می‌دانیم خطای واقعی h بزرگتر از ϵ است، بنابراین:

$$\le \Pr[S \text{ روی } h \text{ خطای } \le \frac{۳}{۴}(h \text{ خطای واقعی })]$$

از نامساوی چرنوف داریم:

$$\le \exp\big(-\frac{m}{۲}(h \text{ خطای واقعی})(\frac{۱}{۴})^{۲}\big)$$

با جایگذاری m داریم:

$$= \exp\big(-\frac{h \text{ خطای واقعی}}{\epsilon}\log\frac{۲^{i+۱}}{\delta}\big) \le \exp\big(-\log\frac{۲^{i+۱}}{\delta}\big) = \frac{\delta}{۲^{i+۱}}$$

به همین ترتیب می‌توان نشان داد با فرض این‌که خطای واقعی h کمتر یا مساوی با $\frac{\epsilon}{۲}$ باشد آن‌گاه احتمال این‌که h نسبت به توزیع $\mathcal{D}$ ناسازگار باشد کمتر مساوی با $\frac{\delta}{۲^{i+۱}}$ است.

اکنون فرض کنید الگوریتم L وجود دارد که ϵ، δ و $\mathrm{size}(c)$ را به عنوان ورودی می‌گیرد و در زمان چندجمله‌ای با احتمال $۱-\delta$ فرضیه‌ی h را خروجی می‌دهد که حداکثر خطای ϵ دارد. الگوریتم جدید L' را به این شکل تعریف می‌کنیم: با $i = ۱$ شروع می‌کنیم و ورودی‌های $\frac{\epsilon}{۲}$، $\delta = \frac{۱}{۲}$ و $\mathrm{size}(c) = \tilde{s} = [۲^{\frac{(i-۱)}{\log\frac{۲}{\delta}}}]$ را به عنوان ورودی به الگوریتم L می‌دهیم. الگوریتم فرضیه‌ی h_i را به عنوان خروجی می‌دهد که با احتمال $\frac{۱}{۲}$ حداکثر خطای $\frac{\epsilon}{۲}$ دارد. مجموعه‌ی S_i شامل $m_i = [\frac{۳۲}{\epsilon}(i\log ۲ + \log\frac{۲}{\delta})]$ نمونه را در نظر بگیرید. اگر h_i با S_i سازگار بود، آن را به عنوان خروجی الگوریتم برمی‌گردانیم و در غیر این صورت مقدار i را یک واحد افزایش می‌دهیم و دوباره همین کار را انجام می‌دهیم. باید نشان دهیم الگوریتم L' یک الگوریتم یادگیری کارای *PAC* برای مساله‌ای است که L آن را یاد می‌گیرد.

فرض کنید $\tilde{s}$ بزرگتر از $\mathrm{size}(c)$ باشد. اگر فرضیه‌ی h_i دارای خطای حداقل $\frac{\epsilon}{۲}$ باشد در این صورت با احتمال حداکثر $\frac{۱}{۴} \le \frac{\delta}{۴} \le \frac{\delta}{۲^{i+۱}}$ ناسازگار خواهد بود و بنابراین با احتمال حداقل $\frac{۳}{۴}$ سازگار است. بنابراین احتمال این‌که h سازگار باشد بزرگتر مساوی خواهد بود با $\frac{۳}{۸} = \frac{۳}{۴}\times\frac{۱}{۲}$. بنابراین احتمال این‌که الگوریتم برای $\mathrm{size(c)} \le \tilde{s}$ به پایان نرسد برابر خواهد بود با $\frac{۵}{۸}$. قرار دهید $j = [\frac{\log\frac{۲}{\delta}}{\log\frac{۵}{۸}}]$. احتمال این‌که بعد از j بار اجرای حلقه در الگوریتم L' برنامه به پایان نرسیده باشد حداکثر برابر خواهد بود با

$$(\frac{۵}{۸})^{j} \le (\frac{۵}{۸})^{\frac{\log\frac{۲}{\delta}}{\log\frac{۸}{۵}}} = \exp\big(\frac{\log\frac{۲}{\delta}}{\log\frac{۸}{۵}\log\frac{۵}{۸}}\big) = \frac{\delta}{۲}$$

اما چه زمانی $\tilde{s} \ge \mathrm{size}(c)$ اتفاق می‌افتد:

$$\begin{aligned}\tilde{s} \ge \mathrm{size}(c) &\iff ۲^{\frac{(i-۱)}{\log\frac{۲}{\delta}}} \ge \mathrm{size}(c)\\ &\iff \frac{i-۱}{\log\frac{۲}{\delta}} \ge \log\mathrm{size}(c)\\ &\iff i \ge ۱ + (\log\mathrm{size}(c))\log\frac{۲}{\delta}\end{aligned}$$

بنابراین با احتمال حداقل $۱-\frac{\delta}{۲}$ الگوریتم L' بعد از حداکثر $j' = [۱ + (\log\mathrm{size}(c))\log\frac{۲}{\delta}] + j$ بار اجرای حلقه متوقف خواهد شد. که فرضیه‌ی خروجی با احتمال حداکثر $\frac{\delta}{۲} \le \frac{\delta}{۲^{j'+۱}}$ خطایی بیشتر از ϵ خواهد داشت. بنابراین الگوریتم L' با احتمال $۱-\delta$ حداکثر خطای ϵ خواهد داشت.

زمان اجرای الگوریتم فوق چندجمله‌ای از اندازه‌ی مفهوم، ϵ و δ خواهد بود. زیرا تعداد اجرای حلقه بعد از j' بار به پایان می‌رسد که j' یک چندجمله‌ای از δ و اندازه‌ی مفهوم است. از طرفی در هر بار اجرای حلقه، یک بار الگوریتم L فراخوانی می‌شود که طبق فرض در زمان چندجمله‌ای اجرای می‌شود. آزمودن فرضیه‌ی خروجی توسط الگوریتم L به وسیله‌ی S نیز در زمان چندجمله‌ای صورت می‌پذیرد. بنابراین در نهایت زمان اجرا حاصل‌ضرب دو چندجمله‌ای است که خود یک چندجمله‌ای خواهد بود.

مراجع

[۱] Haussler, David and Kearns, Michael and Littlestone, Nick and Warmuth, K. Manfred *Equivalence of Models for Polynomial Learnability*. Information and Computation 95, 129-161 (1991)

[۲] Mohri, Mehyar *Course: Foundations of Machine Learning*. Fall 2014.