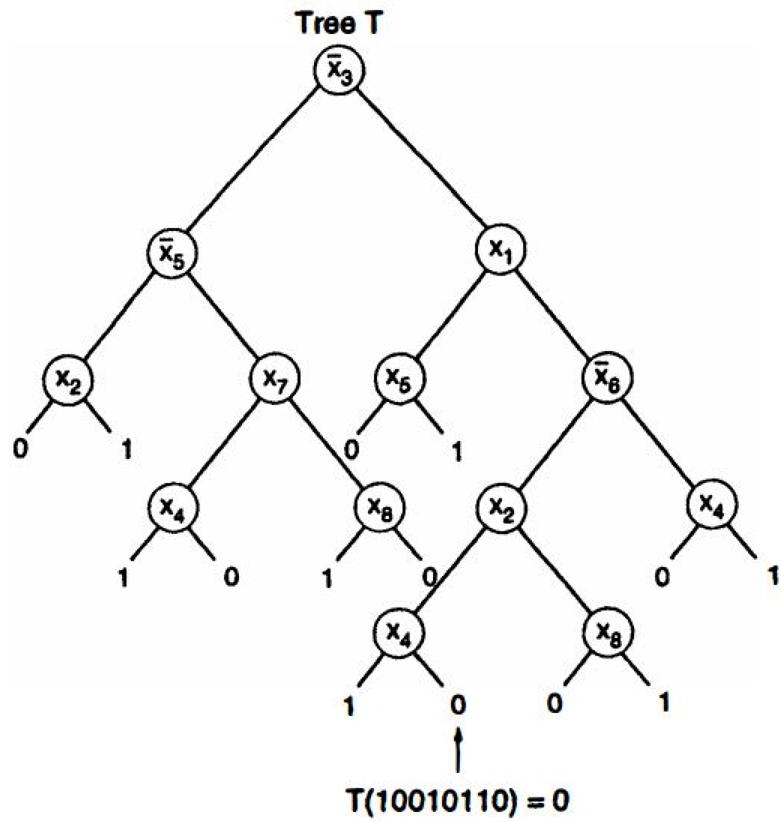
تمرین ۱. یک درخت تصمیم[۱] مشابه یک ۱-لیست تصمیم[۲] است با این تفاوت که در شرایط از درخت‌های دودویی استفاده می‌کنیم و بیت‌های تصمیم‌گیری تنها در برگ‌ها رخ می‌دهد. برای محاسبه‌ی یک درخت مانند T روی ورودی $a \in \{۰,۱\}^n$ مسیری در T را دنبال می‌کنیم که از ریشه شروع می‌شود و لیترال هر راس را روی ورودی a به دست می‌آوریم. اگر مقدار به دست آمده برابر صفر باشد به سمت چپ و اگر یک باشد به راست حرکت می‌کنیم. مقدار $T(a)$ مقداری است که در برگ روی این مسیر ذخیره شده است. شکل ۱ مثالی از یک درخت تصمیم و یک ورودی روی آن را نشان می‌دهد.



شکل ۱: یک درخت تصمیم و مسیری که توسط یک ورودی دنبال می‌شود.

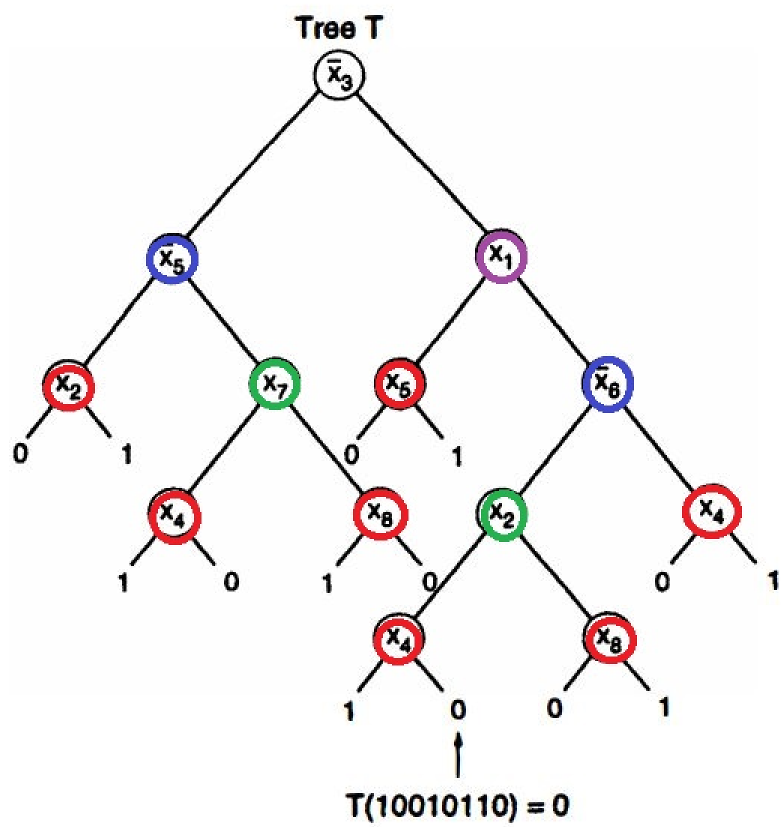
رتبه[۳] یک درخت تصمیم T را به این شکل تعریف می‌کنیم: رتبه‌ی یک درخت با یک راس را برابر ۰ قرار می‌دهیم. اگر رتبه‌ی زیردرخت چپ و راست را به ترتیب با r_L و r_R نشان دهیم در این صورت اگر $r_L = r_R$ باشد، رتبه‌ی T را برابر $r_L + ۱$ و در غیر این صورت برابر $\max(r_L, r_R)$ تعریف می‌کنیم. رتبه میزان نامتوازن بودن یک درخت را مشخص می‌کند. طبق تعریف یک درخت دودویی کامل بیشترین رتبه را خواهد داشت.

رتبه‌ی درخت تصمیم که در شکل ۱ داده شده است را به دست آورید و نشان دهید کلاس توابعی که با درخت‌های تصمیم از رتبه‌ی r

[۱] decision tree
[۲] 1-decision list
[۳] rank

محاسبه می‌شوند زیرمجموعه‌ای از توابعی هستند که توسط r-لیست‌های تصمیم محاسبه به دست می‌آیند. بنابراین برای هر r می‌توان درخت‌های تصمیم با رتبه‌ی r را به صورت کارای PAC یاد گرفت.

پاسخ ۱. از کوچکترین زیردرخت‌ها شروع می‌کنیم و به صورت بازگشتی رتبه‌ی درخت T را به دست می‌آوریم. همان‌طور که در شکل ۲ می‌بینید کوچکترین زیردرخت‌ها را با رنگ قرمز مشخص کرده‌ایم. رتبه‌ی این زیردرخت‌ها همگی برابر ۱ است. در مرحله‌ی بعد زیردرخت‌هایی که با رنگ سبز مشخص شده‌اند را در نظر بگیرید. در این زیردرخت‌های، رتبه‌ی زیردرخت‌های چپ و راست برابر ۱ است پس طبق تعریف رتبه‌ی آن برابر $۲ = ۱ + ۱$ خواهد بود. زیردرخت‌هایی که با رنگ آبی مشخص شده‌اند از دو زیردرخت با رتبه‌های متفاوت تشکیل شده است پس رتبه‌ی آن برابر رتبه‌ی بیشینه‌ی هر دو زیردرخت است که طبق محاسباتی که تا این‌جا انجام شد برابر با ۲ خواهد بود. زیردرخت با رنگ بنفش نیز از دو زیردرخت با رتبه‌های ۲ و ۱ ساخته شده است پس رتبه‌ی آن برابر ۲ خواهد بود. در نهایت برای محاسبه‌ی رتبه‌ی T، رتبه‌ی دو زیردرخت چپ و راست آن برابر ۲ است پس رتبه‌ی آن برابر $۳ = ۱ + ۲$ خواهد بود.



شکل ۲: یک درخت تصمیم و مسیری که توسط یک ورودی دنبال می‌شود.

با استقرا نشان می‌دهیم که درخت‌های تصمیم با رتبه‌ی r، زیرمجموعه‌ای از r-لیست‌های تصمیم است. فرض استقرا، $r = ۱$، را با یک استقرای دیگر روی n تعداد گره‌های درخت، ثابت می‌کنیم قابل تبدیل به یک ۱-لیست است. کوچکترین درخت از مرتبه‌ی یک، شامل یک

ریشه و دو برگ چپ و راست است که به ترتیب مقادیر l یا r را دارند. اگر لیترال ریشه برابر x باشد در این صورت می‌توان آن را به یک ۱-لیست که شامل (x, r) و $b = l$ است، تبدیل کرد. حال فرض کنید برای هر درخت که تعداد راس‌های آن کمتر از n باشد حکم برقرار باشد و بتوان ۱-لیست معادل برای آن ساخت. اگر T یک ۱-درخت تصمیم با n راس باشد. ادعا می‌کنیم یکی از دو زیردرخت T یک برگ است. زیرا در غیر این صورت دو زیردرخت آن هر کدام حداقل رتبه‌ی ۱ دارند، بنابراین طبق تعریف، رتبه‌ی T بیشتر از ۱ خواهد بود که خلاف فرض است (فرض بر این است که در یک درخت تصمیم می‌توان راسی داشت یک به عنوان فرزند، یک برگ و یک راس غیربرگ داشته باشد). بنابراین راس ریشه‌ی T یک برگ دارد و یک زیر درخت. طبق فرض، زیردرخت T با یک ۱-لیست تصمیم معادل است. حال اگر لیترال موجود در ریشه برابر x باشد و برگ آن مقدار $b_۱$ را داشته باشد، اگر ریشه فرزند راست (چپ) باشد قرار می‌دهیم $(x, b_۱)$ ($(\bar{x}, b_۱)$) و سپس ۱-لیست تصمیم زیردرخت را بعد از آن قرار می‌دهیم که حاصل یک ۱-لیست تصمیم خواهد بود و نتیجه‌ی آن همان خروجی درخت تصمیم است.

بنابراین پایه‌ی استقرا برقرار است. فرض کنید هر درخت تصمیم با رتبه‌ی کمتر از $r > ۱$ قابل تبدیل به یک لیست معادل باشد. فرض کنید T یک درخت از رتبه‌ی r و به ترتیب T_r و T_l زیردرخت‌های راست و چپ آن باشند. بعلاوه فرض کنید لیترال x در ریشه قرار دارد. برای این‌که رتبه‌ی T برابر r باشد، دو حالت داریم:

الف) رتبه‌ی هر دو زیردرخت برابر با $r - ۱$ باشد. در این صورت طبق فرض، هر دو زیردرخت را می‌توان به $(r - ۱)$-لیست‌های تصمیم معادل تبدیل کرد. لیترال x را با تمام درایه‌های لیست درخت راست عطف می‌کنیم و لیترال $\bar{x}$ را با تمام درایه‌های لیست زیردرخت چپ. دو r-لیست به دست می‌آید. اگر این دو لیست را کنار هم قرار دهیم، r-لیست حاصل معادل درخت تصمیم T خواهد بود. برای خروجی آخر لیست (اگر هیچکدام از شروط برقرار نبود) اگر فرض کنیم b_l مربوط به لیست حاصل از درخت چپ و b_r درخت راست باشد، در انتها یک درایه‌ی دیگر به لیست اضافه می‌کنیم، لیترال (x, b_r) را در آن قرار می‌دهیم و خروجی آخر لیست را برابر b_l قرار می‌دهیم.

ب) رتبه‌ی دو زیردرخت برابر نباشد. بنابراین یکی از دو زیر درخت T_l و T_r دارای رتبه‌ی r مثلا T_r، و دیگری رتبه‌ی کمتر از r دارد. چون T_l رتبه‌ی کمتر از r دارد پس در فرض استقرا صدق می‌کند. فرض کنید لیترال $y_۱$ در ریشه قرار دارد. نقیض این لیترال (چون در زیردرخت چپ هستیم) را با تمام درایه‌های لیست عطف می‌کنیم و لیست جدیدی به دست می‌آوریم بعلاوه فرض کنید خروجی انتهایی این لیست $b_۱$ باشد. حال زیردرخت سمت راست را در نظر بگیرید. دوباره برای آن می‌توان به همین منوال عمل کرد. اگر $y_۲$ در ریشه‌ی این زیردرخت قرار داشته باشد دوباره دو حالت پیش می‌آید. دوباره همین کار را ادامه می‌دهیم. چون درخت متناهی است، زمانی پیش می‌آید که حالت الف رخ دهد. تمام لیست‌های ساخته شده را در کنار یکدیگر قرار می‌دهیم و در انتهای لیست به ترتیب (y_i, b_i) را اضافه می‌کنیم. بنابراین خروجی یک r-لیست تصمیم خواهد بود.

برای این‌که روش فوق روشن شود، آن را روی درخت تصمیم شکل ۱ که نشان دادیم دارای رتبه‌ی ۳ است اجرا می‌کنیم. برای زیر درخت راست داریم:

$$(x_۴, ۰) \rightarrow ۱$$

$$(x_8, 1) \to 0$$

$$(\bar{x}_2 \wedge x_4, 0) \to (x_2 \wedge x_8, 1) \to (x_2, 0) \to 1$$

$$(x_4, 1) \to 0$$

$$(\bar{x}_6 \wedge x_4, 0) \to 1$$

$$(\bar{x}_6 \wedge x_4, 0) \to (\bar{x}_2 \wedge x_4, 0) \to (x_2 \wedge x_8, 1) \to (x_2, 0) \to (\bar{x}_6, 1) \to 0$$

$$(x_5, 0) \to 1$$

$$(\bar{x}_1 \wedge x_5, 0) \to 1$$

$$(\bar{x}_1 \wedge x_5, 0) \to (\bar{x}_6 \wedge x_4, 0) \to (\bar{x}_2 \wedge x_4, 0) \to (x_2 \wedge x_8, 1) \to (x_2, 0) \to (\bar{x}_6, 1) \to (\bar{x}_1, 1) \to 0$$

به همین ترتیب برای زیردرخت چپ داریم:

$$(x_4, 1) \to 0$$

$$(x_8, 1) \to 0$$

$$(\bar{x}_7 \wedge x_4, 1) \to 0$$

$$(x_7 \wedge x_8, 1) \to 0$$

$$(\bar{x}_7 \wedge x_4, 1) \to (x_7 \wedge x_8, 1) \to (x_7, 0) \to 0$$

$$(x_2, 0) \to 1$$

$$(x_5 \wedge x_2, 0) \to 1$$

$$(x_5 \wedge x_2, 0) \to (\bar{x}_7 \wedge x_4, 1) \to (x_7 \wedge x_8, 1) \to (x_7, 0) \to (x_5, 1) \to 0$$

بنابراین نتیجه نهایی برابر خواهد بود با:

$$(x_3 \wedge x_5 \wedge x_2, 0) \to (x_3 \wedge \bar{x}_7 \wedge x_4, 1) \to (x_3 \wedge x_7 \wedge x_8, 1) \to (x_3 \wedge x_7, 0) \to (x_3 \wedge x_5, 1)$$

$$\to (\bar{x}_3 \wedge \bar{x}_1 \wedge x_5, 0) \to (\bar{x}_3 \wedge \bar{x}_6 \wedge x_4, 0) \to (\bar{x}_3 \wedge \bar{x}_2 \wedge x_4, 0) \to (\bar{x}_3 \wedge x_2 \wedge x_8, 1)$$

$$\to (\bar{x}_3 \wedge x_2, 0) \to (\bar{x}_3 \wedge \bar{x}_6, 1) \to (\bar{x}_3 \wedge \bar{x}_1, 1) \to (\bar{x}_3 \wedge, 0) \to 0$$

این‌که خروجی r-لیست تصمیم ساخته شده به این شکل با خروجی درخت تصمیم یکسان است نیز در بطن اثبات استقرایی می‌گنجد. بنابراین تنها کافی است نشان دهیم که این تبدیل در زمان چندجمله‌ای امکان‌پذیر است بنابراین با توجه به این‌که r-لیست‌ها قابل یادگیری کارای *PAC*

است، بنابراین می‌توان درخت‌های تصمیم از رتبه‌ی مشخص r را نیز به صورت کارا مورد یادگیری قرار داد. این مساله نیز با توجه به الگوریتم ارائه شده واضح است. در هر مرحله لیترال موجود در یکی از راس‌های درخت با درایه‌های یک لیست (که حداکثر برابر تعداد راس‌هاست) عطف می‌شود و برای هر راس این کار یک‌بار صورت می‌گیرد. ترکیب دو لیست نیز حداکثر به تعداد راس‌ها زمان می‌گیرد بنابراین در کل زمان اجرای آن از $O(n^۲)$ خواهد بود و این کار را تمام می‌کند.

تمرین ۲. فرض کنید $\mathcal{C}$ کلاس مفهوم دلخواهی باشد. نشان دهید اگر $\mathcal{C}$ قابل یادگیری کارای *PAC* باشد در این صورت ثابت‌های $\alpha \geq ۱$ و $\beta < ۱$ وجود خواهد داشت به طوری که می‌توان یک الگوریتم (α, β)-اوکام برای $\mathcal{C}$ معرفی کرد.

پاسخ ۲. طبق تعریف یادگیری اوکام، الگوریتم ارائه شده باید با مجموعه S داده شده از نمونه‌ها سازگار باشد. در الگوریتم یادگیری *PAC* این سازگاری با یک خطا و عدم اطمینان همراه است. در این مساله دسترسی ما تنها به یک الگوریتم کارای *PAC* است که به هیچ عنوان نمی‌توان در مورد آن به این نتیجه رسید که خطا روی نمونه‌ها صفر است بنابراین شرط سازگاری قطعی را نخواهیم داشت با این حال اگر اجازه دهیم تا زمان اجرای الگوریتم اوکام تصادفی باشد به شرط این‌که متوسط زمان اجرای آن چندجمله‌ای باشد، آن‌گاه این کار امکان‌پذیر است. بنابراین حکم را برای چنین الگوریتم اوکامی نشان می‌دهیم.

با توجه به این‌که کلاس مفهوم فوق قابل یادگیری کارای *PAC* است پس الگوریتم یادگیری L برای آن وجود دارد و زمان اجرای آن یک چندجمله‌ای برحسب $\frac{۱}{\epsilon}$، $\frac{۱}{\delta}$، $\text{size}(c)$ و n است. اگر این چندجمله‌ای با P نمایش دهیم، در این صورت برای اندازه‌ی فرضیه‌ی h شرط زیر را داریم:

$$\text{size}(h) \leq P$$

از طرفی برای تعداد نمونه‌های لازم اگر کران پایینی داشته باشیم با توجه به این‌که الگوریتم کاراست آن‌هم یک چندجمله‌ای برحسب موارد گفته شده خواهد بود که آن را با Q نمایش می‌دهیم. این کران باید از P کوچکتر باشد. بزرگترین ضرایب ممکن برای $\frac{۱}{\epsilon}$، $\frac{۱}{\delta}$، $\text{size}(c)$ و n در P را در نظر بگیرید، در این صورت با فاکتورگیری از آن‌ها یک مقدار ثابت باقی می‌ماند. بنابراین می‌توان نوشت:

$$P \leq a(n\text{size}(c))^{\alpha}(\frac{۱}{\epsilon\delta})^{k} - Q$$

اگر قرار دهیم $\epsilon = \delta = \frac{۱}{m^{۱/۲K}}$ به دست می‌آوریم:

$$\text{size}(h) \leq a(n\text{size}(c))^{\alpha} m^{\frac{k}{K}}$$

بنابراین اگر $K > k$ انتخاب شود $\beta = \frac{k}{K} < ۱$ خواهد بود و با احتمال δ خطای ϵ خواهیم داشت. در الگوریتم اوکام، فرضیه‌ی خروجی باید با مجموعه‌ی S سازگار باشد بنابراین خطاهای موجود را باید به نحوی از بین ببریم. برای این‌کار، فرضیه را مورد آزمون قرار می‌دهیم و در هر مورد خطا داشتیم، آن را حفظ (در حافظه ذخیره) می‌کنیم. برای ذخیره‌ی هر کدام از نمونه‌ها، به تعداد ثابتی بیت نیاز داریم که اگر Σ زبان مورد استفاده در نمایش باشد $\log(|\Sigma|^{n})$ بیت مورد نیاز است. و با توجه به این‌که خطا حداکثر ϵ است، در نهایت داریم:

$$\text{size}(h) \leq a(n\text{size}(c))^{\alpha} m^{\frac{k}{K}} + \epsilon m(\log(|\Sigma|^{n})) = a(n\text{size}(c))^{\alpha} m^{\frac{k}{K}} + m^{(۱-\frac{۱}{۲K})}(\log(|\Sigma|^{n}))$$

بنابراین اگر قرار دهیم $\beta = \max(\frac{k}{K}, ۱ - \frac{۱}{۲K}) < ۱$ و $b = \max(a, \log(|\Sigma|^n))$ خواهیم داشت:

$$\mathrm{size}(h) \leq b(n\mathrm{size}(c))^{\alpha} m^{\beta}$$

با توجه به توضیحات فوق، الگوریتم (α, β)-اوکام را به شکل زیر ارائه می‌کنیم:

فرض کنید مجموعه‌ی S شامل m نمونه از نمونه‌های برچسب‌دار حاصل از مفهوم $c \in \mathcal{C}$ داده شده است. اعضای S را با شماره‌های ۰ تا $m - ۱$ مرتب می‌کنیم. الگوریتم L را اجرا می‌کنیم. اگر الگوریتم برای i-امین بار از اوراکل درخواست یک نمونه‌ی جدید دارد نمونه‌ی iام (به همنهشتی m) از مجموعه‌ی S را برمی‌گردانیم (می‌توان از توزیع یکنواخت روی S نیز استفاده کرد) . بنابراین خطای فرضیه روی S با احتمال $۱ - \delta$ حداکثر برابر با ϵ خواهد بود. که با توجه به ϵ و δ انتخاب شده، اگر m به مقدار کافی بزرگ باشد، خطای حاصل به مقدار کافی کوچک خواهد بود.

الگوریتم فوق را آن‌قدر تکرار می‌کنیم تا فرضیه‌ی خروجی خطای کمتر از ϵ داشته باشد. بنابراین تنها کافی است نشان دهیم متوسط زمان اجرای این‌کار چندجمله‌ای است. احتمال این‌که تا مرحله‌ی $i - ۱$ نتیجه‌ی مناسب نگیریم و در مرحله‌ی iام فرضیه‌ی با خطای کم حاصل شود برابر خواهد بود با $(۱ - \delta)\delta^{i-۱}$. اگر فرض کنیم زمان اجرای هر مرتبه از الگوریتم برابر چندجمله‌ای H باشد در این صورت زمان متوسط اجرای الگوریتم برابر خواهد بود با:

$$\sum_{i=۱}^{\infty} (i\delta^{i-۱}(۱ - \delta)H) = (۱ - \delta)H \sum_{i=۱}^{\infty} i\delta^{i-۱}$$

که با توجه به این‌که $\delta < ۱$ است سری فوق همگراست و داریم $\sum\limits_{i=۱}^{\infty} i\delta^{i-۱} = \frac{۱}{(۱-\delta)^۲}$ بنابراین متوسط زمان اجرا برابر خواهد بود با:

$$H(۱ - \delta)\frac{۱}{(۱ - \delta)^۲}$$

که چندجمله‌ای است برحسب m، n و $\mathrm{size}(c)$ خواهد بود (چون δ و ϵ برحسب m در نظر گرفته شده‌اند).