به نام آنکه وجودم ز وجودش گشته موجود

دانشگاه صنعتی شریف

دانشکده علوم ریاضی

گزارش کار ارائه ای برای درس پیچیدگی محاسبات

با عنوان :

درخت تصمیم

گزارش کار :

راضیه شفیعی

استاد درس :

دکتر محمد هادی فروغمند اعرابی

بهار 1394

چکیده :

جمله ی حکیمانه ی زیر، انگیزه ای است برای خواندن مطالب این فصل از کتاب :

هیچ کس نمی گوید انجام کارسخت است، بلکه سخت ترین کار در جهان تصمیم گرفتن است.

در حال حاضر بررسی یه سری مسائل پایه ای در رابطه با توان ماشین تورینگ خارج از دسترس ما است، لذا به بررسی این مسائل روی مدل های محاسباتی محدودتر و ساده تر می پردازیم.

اتفاقا یکی از ساده ترین این مدل ها، درخت تصمیم است.

در این فصل پیچیدگی یک تابع بولی**f**، بر حسب تعداد بیت هایی از ورودی که بررسی میشوند تا مقدار تابع روی آن ورودی محاسبه شود، اندازه گیری می شود.

در این فصل به تعاریفی مثل: درخت تصمیم، پیچیدگی درخت تصمیم و انواع درخت تصمیم؛ مثل غیرقطعی و احتمالاتی؛ مشابه ماشین تورینگ پرداخته میشود.

همچنین یه سری تکنیک برای اثبات کران پایین روی درخت تصمیم ذکر می شود و نیز لم مفید **Yao's Min Max** یکی از این تکنیک ها است برای اثبات کران پایین پیچیدگی درخت تصمیم تصادفی است و عموما برای اثبات کران پیجیدگی تصادفی مدل های محاسباتی دیگر کاربرد دارد.

فهرست مطالب بررسی شده :

فهرست شکل ها :

بخش 12.1 : درخت تصمیم و پیچیدگی درخت تصمیم

تابع $f:\{0,1\}^n \rightarrow \{0,1\}$ را در نظر بگیرید.

تعریف درخت تصمیم :

تعریف اول :

درختی است که گره های داخلی آن با متغیربرچسب گزاری شده است واز هر گره دو یال خارج می شودکه با 0 یا 1 برچسب داده شده است و به ترتیب 0-یال یا 1-یال نامیده می شوند و نیز هر برگ درخت با 0 یا1 برچسب گزاری شده که بیانگر مقدار تابع برای آن شاخه است.

تعریف دوم :

2)درختی است که در هرسطح از متغیرها سوال پرسیده می شود و متغیرها بصورت احتمالی و باتوجه به سوال قبلی پاسخ می دهند .به شکل های زیر توجه شود:

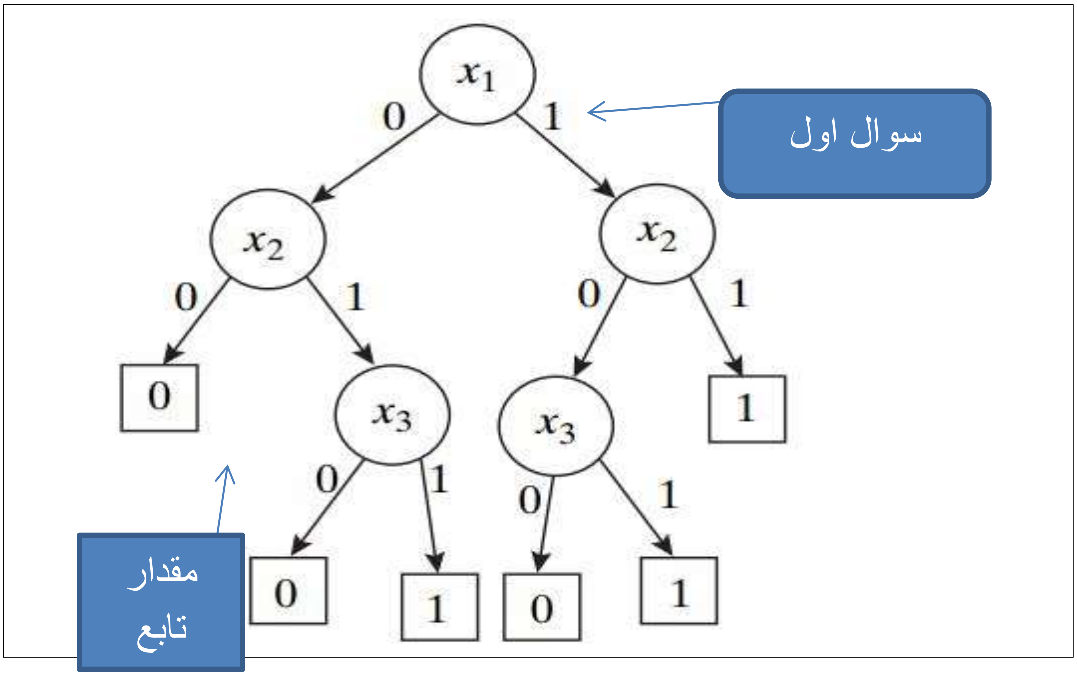

figure 1- Decision tree for computing $Maj(x_1,x_2,x_3)$

به کمک مثال زیر به توضیح ***روش خصمانه*** می پردازیم :

مثال ٭٭: تابع $f(x_1, \dots, x_n) = \vee_{i=1}^{n} x_i$ را در نظربگیرید. می خواهیم نشان دهیم که درخت تصمیم با عمق کمتر از n نمی تواند این تابع را محاسبه کند، از روش خصمانه کمک می گیریم، به این صورت که اگر t یک درخت تصمیم باشد که تابع را محاسبه می کند، آنگاه اجرای t به این صورت است : تا آنجایی که می تواند جواب غلط می دهد و درخت تصمیم را آویزان نگه می دارد، یعنی برای (n-1) مرحله به هر سوال پاسخ 0 (یعنی غلط) می دهد، بنابراین تنها n-امین بیت مشخص کننده ی مقدار تابع است که بلاخره 0 است خروجی یا 1، لذا D(f)=n

درخت تصمیم به عمق n درختی است که حداکثرتعداد سوالات هرشاخه ی آن n تا است .

تعریف cost(t,x) :

ارزش درخت t روی ورودی x که با نماد cost(t,x) نشان داده می شود عبارتست از تعداد بیت های ورودی x که توسط درخت t بررسی میشود. برای شکل قبل داریم :

Cost(t,00)=2

Cost(t,011)=3

تعریف پیچیدگی درخت تصمیمی که تابع f را محاسبه میکند :

D(f)=$\mathbf{min}_{t\in\mathcal{T}_f} \mathbf{max}_{x\in\{0,1\}^n} \boldsymbol{cost(t,x)}$

بصورت شهودی تعداد بیت هایی است که توسط کارامدترین درخت تصمیم روی بدترین ورودی ممکن بررسی می شود.

در تعریف فوق $\mathcal{T}_f$ برابر است با مجموعه درخت های تصمیم قطعی که تابع f را محاسبه می کنند.

توجه داریم که چون هرتابع f روی {0و1} توسط یک درخت باینری پر با عمق n محاسبه میشود لذا D(f)≤n (حداکثر n سوال میتوان پرسید)

بخش 12.2 : پیچیدگی سند (که می تواند به عنوان یک ورژن غیرقطعی از پیچیدگی درخت تصمیم در نظر گرفته شود.) :

تعریف پیچیدگی سند:

فرض کنید $f:\{0,1\}^n \rightarrow \{0,1\}$ و $x\in \{0,1\}^n$. زیر مجموعه ی $S\subseteq \{0,1\}^n$ را یک **0–سند** برای x نامیم هرگاه برای هر x' که $x'/S = x / S$ داشته باشیم $f(x')=0$.
مشابها $S\subseteq \{0,1\}^n$ را **1–سند** نامیم هرگاه برای هر x' که $x'/S = x / S$ داشته باشیم: $f(x')=1$.

پیچیدگی سند تابع f را کوچکترین k نامیم که برای هر رشته x یک f(x) –سند به اندازه حداکثر k داشته باشیم.

توجه1 : 0سند و 1 سند همزمان برای یک ورودی وجود ندارند.

توجه 2 : درصورتی که تابع f دارای درخت تصمیمی مثل t باعمق k باشد، آنگاه $C(f)\leq k$ زیرا بوضوح مجموعه مکان هایی که توسط t برای ورودی x بررسی میشود، یک f(x) –سند است برای x. بنابراین $C(f)\leq D(f)$ اما گاهی اوقات داریم : $C(f)< D(f)$.

توجه 3 :رابطه مطالبی که گفتیم با کلاس های P, NP , coNP به شرح زیر است:

low decision tree complexity ↔ P
low 1- certificate complexity ↔ NP
low 0- certificate complexity ↔ coNP

قضیه : فرض کنید $f:\{0,1\}^n \rightarrow \{0,1\}$. برای این تابع داریم :

$$D(f) \leq C(f)^2$$

طرح اثبات : یک سند دلخواه **C(f)=k** را برای تابع در نظر میگیرد و یک الگوریتم قطعی درخت تصمیم ارائه می دهد که حداکثر k^2 سوال می پرسد (توجه داریم که 1–سند و 0- سند همزمان برای یک ورودی وجود ندارند.)

سپس نتیجه می شود که :

$$C(f)\leq D(f) \leq C(f)^2$$

با توجه به قضیه ی فوق به طرز شگفت انگیزی دیده می شود که :

$$P = NP \cap coNP$$

بخش 12.3 : تعریف درخت تصمیم تصادفی

درخت تصمیم تصادفی عبارتست از یک توزیع احتمالی روی درخت های تصمیم قطعی .

ما درخت های تصمیم تصادفی را در نظر خواهیم گرفت که خروجی آن ها همیشه جواب صحیح است اما با استفاده از احتمال تصادفی، امید ریاضی cost را افزایش می دهد.

(مشابه کلاس ZPP(بخش7.3 کتاب))

نکته : در درخت تصمیم تصادفی اینکه کدام محل از ورودی مورد پرسش قرار بگیرد، بصورت احتمالی مشخص میگردد.

تعریف پیچیدگی درخت تصمیم تصادفی :

برای هر تابع f فرض کنید که $\mathcal{P}_f$ مجموعه ی تمام توزیع های احتمالی روی درخت های تصمیمی باشد که تابع را محاسبه می کنند.

پیچیدگی درخت تصادفی روی f بصورت زیر تعریف می شود:

$$R(f) = \min_{P \in \mathcal{P}_f} \max_{x \in \{0,1\}^n} \underset{t \in_R P}{E} [cost(t, x)]$$

که دیده می شود : پیچیدگی درخت تصمیم تصادفی بهترین توزیع احتمالی ممکن روی درخت را برای بدترین ورودی ممکن ارائه می دهد.

به شکل زیر که درخت تصمیم تصادفی برای تابع اکثریت است، توجه شود :

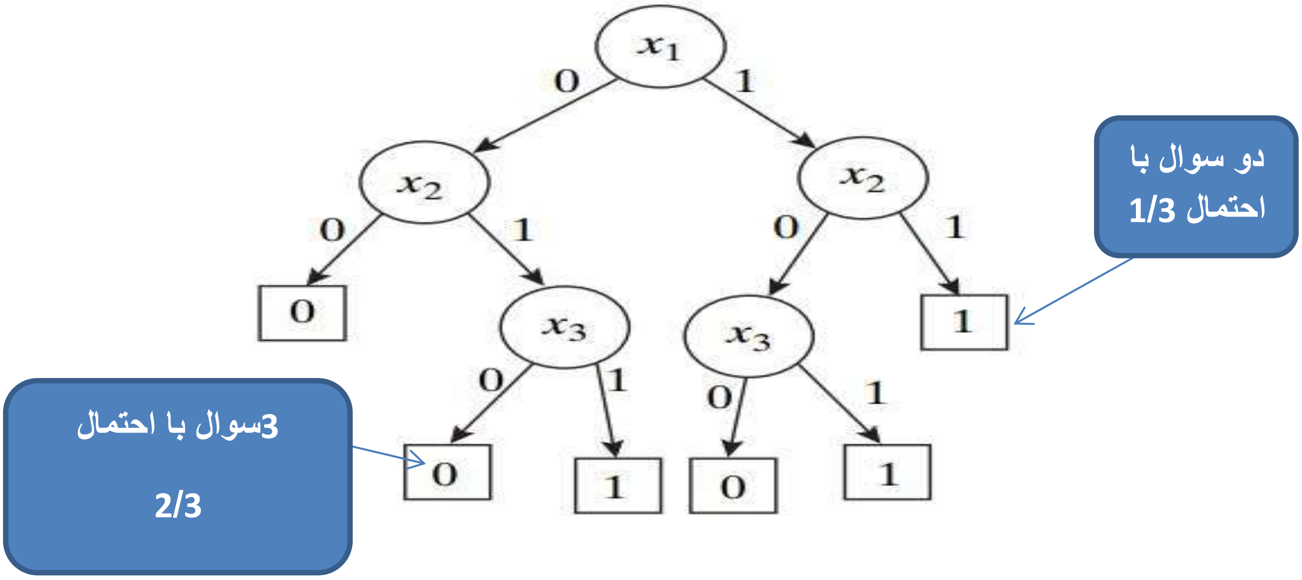

Figure 2 : Randomized decision tree

نکته 1 : **R(f)**$\leq D(f)$ زیرا درخت تصمیم قطعی، نوع خاصی از درخت توزیع احتمالی است.

نکته 2 : **R(f)**$\geq C(f)$ زیرا برای هر ورودی $x\in \{0,1\}^n$ و هر درخت تصمیم که تابع را محاسبه میکند، یک *f(x)*-سند برای *x* از اندازه *cost(t,x)* بدست می آوریم؛ بنابراین امید ریاضی با اندازه کوچکترین سند ممکن برای *x* کراندار شده است.

مطلب فوق مشابه است با اینکه ***ZPP***$\subseteq NP \cap coNP$. لذا داریم :

C(f)$\leq R(f) \leq D(f)$

بخش 12.4 : چند تکنیک برای اثبات کران پایین درخت تصمیم :

روش خصمانه را در مثال ** از بخش 12.1 دیدیم. این روش برای اثبات کران پایین روی پیچیدگی درخت تصمیم قطعی بکار می رود، اما همیشه مفید نیست، بویژه هنگامی که پیچیدگی سند(درخت تصمیم غیرقطعی) و پیچیدگی درخت تصمیم تصادفی مد نظر باشد.

روش دیگر **لم Yao's Min Max** می باشد که در زیر آمده است :

لم Yao's Min-Max میگوید که در زیر رابطه 1 با 2 برابراست :

فرض کنید که X یک مجموعه ی متناهی از ورودی ها باشد و فرض کنید که A مجموعه ی همه ی الگوریتم های قطعی باشد که بعضی از مسائل محاسباتی f را روی این ورودی ها حل می کند. برای

$A \in \mathcal{A}$, $x \in X$ ارزش A روی x راکه با cost(A,x) نمایش دا ده می شود که عبارتست از ارزش متحمل شده از جانب A روی ورودی x .(که این ارزش میتواند زمان اجرا باشد یا پیچیدگی درخت و...). یک الگوریتم تصادفی A می تواند به عنوان یک توزیع احتمالی R روی A در نظر گرفته شود .ارزش R روی ورودی x که با cost(R,x) نشان داده می شود برابر است با

$$\underbrace{E}_{A \in_R R} [cost(A,x)]$$

پیچیدگی تصادفی مساله برابر است با

$$\min_R \max_{x \in X} cost(R,x) \qquad (1)$$

فرض کنید D یک توزیع روی ورودی ها باشد .برای هر الگوریتم قطعی A ارزش A روی D را با cost(A,D) نشان میدهیم که هست :

$$\underset{x \in_R D}{E} [cost(A,x)]$$

پیچیدگی توزیع مساله برابر است با :

$$\max_D \min_{A \in A} cost(A,D) \qquad (2)$$