Výpočtové učenie

5. apríla 2005

Kapitola 1

Efektívne učenie II

1.1 Účinnosť versus dôveryhodnosť a presnosť

Diskusia v predchádzajúcej kapitole sa sústredila na chovanie sa učiaceho algoritmu v čase behu, ktorý závisel (bol funkciou) len od veľkosti príkladov n. Zrejme sú aj **iné faktory, ktoré určujú čas** behu učiaceho algoritmu, a mohli by sme zaviesť pojem efektívnosti vzhľadom na **úroveň dôveryhodnosti a úroveň presnosti**. Potom prediskutujeme efektívnosť vzhľadom na veľkosť reprezentácie cieľového konceptu. Tieto úvahy sú relevantné pre ľubovoľný hypotézový priestor a môžu byť kombinované s ideami nasledujúcej kapitoly, aby bola zavedená celkom všeobecná definícia pojmu **efektívny PAC učiaci algoritmus**.

V predchádzajúcej kapitole sme zaviedli pevné, ale ľubovoľné parametre, a síce parameter dôveryhodnosti δ a parameter presnosti ϵ. Je zrejmé, že zníženie aspoň jednej hodnoty z týchto veličín urobí učenie ťažším, a preto čas behu efektívneho PAC učiaceho algoritmu by mohol byť ohraničovaný nejakým vhodným spôsobom pomocou rastúcich δ^{-1} a ϵ^{-1}. Mohli by sme sa jednoducho pýtať, či čas behu rastie polynomiálne vzhľadom na δ^{-1} a ϵ^{-1}, ale táto závislosť na δ^{-1} nie je celkom vhodná z nasledujúcich dôvodov. Ak dĺžka tréningovej vzorky, ktorá vstupuje do efektívneho učiaceho algoritmu je zdvojnásobená, môžeme očakávať pravdepodobnosť, že výstupná hypotéza je zlá približne kvadratickú. Inak povedané, **vzťah medzi zložitosťou vzorky a δ^{-1} je logaritmický**. Motivovaní týmto, mohli by sme povedať, že učiaci algoritmus L je **efektívny vzhľadom na dôveryhodnosť**, ak jeho čas behu je polynomiálny v m a zložitosť vzorky $m_L(H, \delta, \epsilon)$ závisí polynomiálne od veličiny $\ln(\delta^{-1})$, čo budeme označovať δ^*. V prípade parametra presnosti budeme hovoriť, že L **je efektívny vzhľadom na presnosť**, ak jeho čas behu je polynomiálny v m a zložitosť vzorky závisí polynomiálne od ϵ^{-1}. Ak obe tieto podmienky platia, potom čas behu potrebný na vytvorenie PAC výstupnej hypotézy je polynomiálny v δ^* a ϵ^{-1}.

Napríklad, ak H je ľubovoľný konečný hypotézový priestor a L je konzistentný učiaci algoritmus pre H, potom teória navrhnutá predtým implikuje, že dolná hranica pre zložitosť vzorky je $m_0(H, \delta, \epsilon) = \lceil \epsilon^{-1} \cdot \ln(|H|/\delta) \rceil$.

V tomto prípade m_0 je zrejme ohraničené polymomiálnou funkciou vzhľadom na δ^* a ϵ^{-1}. Ak čas behu L je polynóm v m, potom L je PAC učiaci algoritmus pre H, ktorý beží v polynomiálnom čase vzhľadom na δ^* a ϵ^{-1}. Ten istý argument platí v odstupňovanom prípade. Ak $H = \bigcup H_n$ je hypotézový priestor booleovských funkcií odstupňovaný veľkosťou príkladov, potom dolná hranica pre zložitosť vzorky je

$$m_0(H_n, \delta, \epsilon) = \left\lceil \frac{1}{\epsilon} \cdot \ln\left(\frac{|H_n|}{\delta}\right) \right\rceil.$$

V tomto prípade, ak čas behu $R_L(m, n)$ je polynóm v m a n a ak $\ln |H_n|$ je polynóm v n, potom L PAC učí H_n v polynomiálnom čase behu nielen v n, ale tiež v δ^* a ϵ^{-1}.

1.2 PAC učenie a problém konzistencie

Analýza urobená na konci predchádzajúcej sekcie je motivovaná doteraz známymi vzťahmi medzi konzistenciou a PAC učením. Obrátením našej pozornosti na neodstupňovaný prípad výsledok je jednoduchý, a síce ak existuje konzistentný učiaci algoritmus L pre konečný hypotézový priestor H, ktorý beží v polynomiálnom čase vzhľadom na dĺžku vzorky m, potom L PAC učí H v polynomiálnom čase vzhľadom na δ^* a ϵ^{-1}. Zhruba povedané, môžeme hovoriť, že efektívny "konzistentný-hľadač-hypotéz" je efektívny "PAC-learner". V tejto sekcii uvedieme, k čomu vedie opačná úvaha.

Toto by znamenalo, že efektívne PAC učenie implikuje efektívne hľadanie konzistentných hypotéz za predpokladu, že sme pripravení akceptovať **náhodný algoritmus**. Úplné vysvetlenie tohto pojmu možno nájsť v knihe Cormena, Leiersona a Rivesta (1990), ale pre naše účely postačí vysvetlenie uvedené v nasledujúcich odsekoch.

Predpokladajme, že máme daný nejaký **generátor náhodných čísel**, ktorý pre dané ľubovoľné celé číslo $I \geq 2$ produkuje náhodné čísla i v intervale $1 \leq i \leq I$, pričom každá hodnota je rovnako pravdepodobná.

Náhodný algoritmus A má povolené brať tieto čísla ako časť svojho vstupu. **Výpočet algoritmu A je riadený jeho vstupom** tak, že závisí od partikulárnej postupnosti produkovanej generátorom náhodných čísel. Z toho vyplýva, že môžeme hovoriť o pravdepodobnosti, že A má daný výsledok, čím je mienená relatívna frekvencia postupností, ktoré produkuje tento výsledok vzhľadom na celkový počet možných postupností.

Hovoríme, že náhodný algoritmus A "rieši" problém vyhľadávania Π, ak sa chová nasledujúcim spôsobom: Algoritmus vždy zastaví a produkuje výstup. Ak A padol pri hľadaní riešenia pre Π, dá jednoducho výstup *nie*. Ale s pravdepodobnosťou aspoň $\frac{1}{2}$ (v zmysle vyjadrenom vyššie), A je úspešný pri hľadaní riešenia pre π a výstupom je jeho riešenie.

Praktická použiteľnosť náhodného algoritmu vyplýva z faktu, že opakovaním algoritmu niekoľkokrát veľmi rýchlo rastie pravdepodobnosť úspechu. Ak algoritmus padne pri prvom pokuse, čo sa stane s pravdepodobnosťou najviac $\frac{1}{2}$, potom jednoducho skúšame ďalej. Pravdepodobnosť, že padne dvakrát, je najviac $\frac{1}{4}$, že padne k-krát je najviac $(\frac{1}{2})^k \to 0$. Teda v praxi náhodný algoritmus je takmer tak dobrý ako obyčajný - samozrejme za predpokladu, že má polynomiálny čas behu. Máme nasledujúcu vetu Pitta a Valianta (1988) (tiež Matarjan (1989) a Haussler et. al. (1988)).

Veta 1 *Nech H je hypotézový priestor a predpokladajme, že existuje PAC učiaci algoritmus pre H s časom behu polynomiálnym v ϵ^{-1}. Potom existuje náhodný algoritmus, ktorý rieši problém hľadania hypotézy v H konzistentnej s danou tréningovou vzorkou a ktorý má čas behu polynomiálny v m (dĺžka tréningovej vzorky).*

Dôkaz. Predpokladajme, že $\overline{s}$ je tréningová vzorka pre cieľovú hypotézu $t \in H$ a $\overline{s}$ obsahuje m rôzne označených príkladov. Ukážeme, že je možné nájsť hypotézu konzistentnú s $\overline{s}$ spustením daného PAC algoritmu L na odpovedajúcej tréningovej vzorke. Definujme pravdepodobnostné rozdelenie μ na príkladovom priestore X takto:

$$\mu(x) = \begin{cases} \frac{1}{m}, & \text{ak } x \text{ sa vyskytuje v } \overline{s} \\ 0, & \text{inak.} \end{cases}$$

Môžeme použiť generátor náhodných čísel s výstupnými hodnotami $\in \langle 1, m \rangle$ na výber príkladov z X podľa tohoto rozdelenia: pridelíme každé náhodné číslo ako návestie $1..m$ rovnako pravdepodobného príkladu. Teda výber tréningovej vzorky dĺžky m pre t na základe rozdelenia μ môže byť simulovaný generovaním postupnosti m náhodných čísel v danom intervale.

Nech L je PAC učiaci algoritmus, ako bolo uvedené vyššie. Potom ak máme dané 4 veličiny δ, ϵ, μ, t, tak môžeme nájsť celé číslo $m_0(\delta, \epsilon)$

$$\underset{\epsilon > 0}{\forall} \quad \underset{\delta > 0}{\forall} \quad \underset{m_0(\delta,\epsilon)}{\exists} \quad \underset{\mu}{\forall} \quad \underset{t}{\forall} \quad \mu^m\{s \in S(m,t) : \quad h(s,t) < \epsilon\} > 1 - \delta$$

Predpokladajme, že špecifikujeme dôveryhodnosť $\delta = \frac{1}{2}$ a presnosť $\epsilon = \frac{1}{m^*}$.

Ak spustíme učiaci algoritmus L na tréningovej vzorke dĺžky $m_0(\frac{1}{2}, \frac{1}{m^*})$, získanej náhodne podľa rozdelenia μ, vlastnosť PAC algoritmu zaručí, že pravdepodobnosť, že chyba výstupu je menšia než $\frac{1}{m^*}$, je včšia než $1 - \frac{1}{2} = \frac{1}{2}$. Pretože nie sú žiadne príklady s pravdepodobnosťou striktne medzi 0 a $\frac{1}{m^*}$, z toho vyplýva, že pravdepodobnosť, že výstup súhlasí presne s tréningovou vzorkou je väčší než $\frac{1}{2}$.
□

Procedúra uvedená vo vyššom odseku je základom pre náhodný algoritmus L^* pre hľadanie hypotézy, ktorá je konzistentá s danou tréningovou vzorkou $\overline{s^*}$.

Zhrnutie v krokoch pre L^*:

- Vyhodnotiť $m_0 = m_0(\frac{1}{2}, \frac{1}{m^*})$.

- Použitím gnč na skonštrukciu tréningovej vzorky $\overline{s}$ dĺžky m_0 podľa pravdepodobnostného rozdelenia μ.
- Spustiť daný PAC učiaci algoritmus L na $\overline{s}$.
- Skontrolovať výslednú hypotézu $L(\overline{s})$, či je konyistentná s $\overline{s^*}$.
- Ak hypotéza nie je konzistentná s $\overline{s^*}$, výstup "nie". Ak hypotéza je konzistentá s $\overline{s^*}$, výstupom je táto hypotéza.

Ako sme uviedli, PAC vlastnosť algoritmu L zaručí, že L^* je úspešný s pravdepodobnosťou viac ako $\frac{1}{2}$. Nakoniec je zrejmé, že ak čas behu algoritmu L je polynomiálny v ϵ^{-1}, potom čas behu L^* je polynomiálny v $m^* = \epsilon^{-1}$.
□

Veta 1 nám umožňuje rozšíriť zložitosť výsledkov pre problém konzistencie, ako bolo dokázané vyššie, na PAC učenie. Pripomeňme, že oba problémy - problém rozhodnutia o konzistencii a problém hľadania konzistentnej hypotézy sú NP-ťažké v niektorých prípadoch, takých ako hypotézový priestor $C^k = \bigcup C_n^k$. Veta hovorí, že ak by sme mohli PAC naučiť C_n^k v polynomiálnom čase vzhľadom na ϵ^{-1} a n, potom by sme mohli nájsť konzistentnú hypotézu použitím náhodného algoritmu s časom behu polynomiálnym v m a n. V jazyku teórie zložitosti by to mohlo znamenať, že uvedený problém je v triede RP, čo je trieda problémov, ktoré môžu byť riešené v "pravdepodobne polynomiálnom čase". Teraz sa predpokladá, že RP neobsahuje žiadny NP-ťažký problém - teda, že RP≠NP, čo niektorí považujú za podobne ťažké ako NP=P. Takže keď toto akceptujeme, z toho vyplýva, že neexistuje žiadny PAC polynomiálny algoritmus pre odstupňovaný priestor C^k, ak $k \geq 2$.

Vyššie uvedená diskusia ukazuje, že pre $k \geq 2$, C^k nie je efektívne PAC naučiteľný vzhľadom na veľkosť príkladu. Avšak pre ľub. n, C_n^k je obsiahnuté v $D_{n,k}$, priestore disjunkcií jednočlenov s najviac k literálmi. Valiantov učiaci algoritmus pre $D_k = \bigcup D_{n,k}$ popísaný vyššie je konzistentný algoritmus s časom behu $R_L(m,n) = O(m \cdot n^k)$, polynóm v n aj m. Preto pre ľub. tréningovú vzorku $\overline{s}$ pre hypotézu v C_n^k tento algoritmus bude produkovať v polynomiálnom čase hypotézu v $D_{n,k}$ konzistentnú s $\overline{s}$. V ďalších poznámkach predchádzajúcej kapitoly sme definovali, čo znamená učiaci algoritmus L pre odstupňovaný priestor $\bigcup C_n$ iným odstupňovaným priestorom $\bigcup H_n$; k danej tréningovej vzorke $\overline{s}$ pre hypotézu z C_n L vráti hypotézu $L(s) \in H_n$. Použitím tejto terminológie, štandardný učiaci algoritmus pre D_k je PAC učiaci algoritmus pre $(C^k, D_{n,k})$, efektívný vzhľadom na veľkosť príkladov. Teda v protiklade k negatívnemu výsledku vyššie, C^k je efektívne naučiteľný pomocou "**väčšieho priestoru**". Nie je tu žiadny spor. Zhruba povedané, je ťažké nájsť formulu pre konzistentnú hypotézu v C_n^k, pretože tento priestor je príliš "ohraničený". Daná je včšia flexibilita v práci v "bohatšom" priestore $D_{n,k}$, v ktorom algoritmus môže vyjadriť svoje hypotézy vo výrazoch $D_{n,k}$ formúl; môže byť dosiahnuté rýchlejšie učenie. Výsledok **o nenaučiteľnosti** je preto uvažovaný aj v zmysle závislosť od reprezentácie.

1.3 Veľkosť reprezentácie

Už sme sa zmienili o tom, že výstup realistického učiaceho algoritmu nie je abstraktná funkcia ale skôr reprezentácia tejto funkcie cez formulu alebo stav stroja. Pretože booleovská funkcia, ktorá môže byť reprezentovaná krátkou booleovskou formulou je zrejme "jednoduchšia" než taká, ktorá vyžaduje dlhšiu formulu, môžeme očakávať, že je ju ťažšie naučiť než krátku formulu.

Rámec potrebný pre starostlivú diskusiu o takých veciach je poskytnutý dojmom reprezentácie $\Omega \to H$ ako bolo uvedené vyššie. Množina Ω môže byť myslená ako množina formúl alebo množina stavov stroja tak, že pre každé $\omega \in \Omega$ existuje odpovedajúca hypotéza h_ω. V nasledujúcich niekoľkých sekciách uvedieme ako reprezentácia hypotéz ovplyvňuje čas behu učiacich algoritmov.

Aby sme to mohli urobiť, potrebujeme nejakú mieru pre "veľkosť" reprezentácie hypotézy. Samozrejme, že neexistuje žiadna absolútna miera a tak musíme skonštruovať jednu, ktorá sa zdá byť rozumná pre skúmané problémy. Booleovský prípad je najpriamočiarejší.

Štandardná metóda reprezentujúca bool. funkciu pomocou formúl bola popísaná v 2.3. Formálne používame abecedu $4 + 2n$ symbolov: ( ) $\wedge$ $\vee$ u_1 $\overline{u_1}$ $\ldots u_n$ $\overline{u_n}$, ktoré sú skombinované podľa určitých pravidiel. Táto abeceda môže byť zakódovaná použitím $3 + \lceil \log n \rceil$ bitov pre každý symbol ako je možné ukázať v nasledujúcej tabuľke:

Symbol	Kód
(	110 0000...0
)	101 000...0
$\vee$	101 000...0
$\wedge$	111 000...0
u_1	001 000...01
$\overline{u_1}$	000 000...01
u_2	$\underbrace{001}\ \underbrace{000\ldots10}$
	$\vdots$

Idea je v tom, že prvé 3 bity sú použité na kódovanie povahy symbolu a zvyšných $\lceil \log_2 n \rceil$ bitov je použitých na reprezentáciu symbolov u_i, $\overline{u_i}$. Nech ω je správna formula, ktorá je dosiahnutá z abecedy uvedenej vyššie. Ak ω má $\overline{s}$ symbolov, potom môže byť zakódovaná pomocou $s \cdot (3 + \lceil \log_2 n \rceil)$ bitov a tak môžeme definovať veľkosť ω

$$\|\omega\| = s \cdot (3 + \lceil \log_2 n \rceil).$$

Napríklad, ak $n = 3$, $\omega = (u_1 \wedge u_2) \vee u_3$ (7 symbolov)

$$\|\omega\| = 7 \cdot (3 + \lceil \log_2 3 \rceil) = 35.$$

Ako sme už uviedli, výstup učiaceho algoritmu nie je abstraktná funkcia alebo hypotéza, ale reprezentácia hypotézy pomocou formuly alebo stroja. Z tohto hľadiska je rozumné porovnať veľkosť takého výstupus veľkosťou vstupu do algoritmu, čo je vlastne tréningová vzorka označených príkladov.

Príklad 1 *Predpokladajme, že máme 20 príkladov 30 bitových do učiaceho algoritmu pre monočleny. Celkový počet bitov vstupu je* $20 \cdot (30 + 1) = 620$. *Výstupom je hypotéza-monočlen, ktorý môže byť akceptovaný pomocou 30 literálov a 29 konjunkcií. Použitím vyššie uvedenej kódovacej schémy dostávame počet bitov výstupu*

$$(30 + 29) \cdot (3 + \lceil \log_2 30 \rceil) = 59 \cdot 8 = 472$$

Pretože toto číslo je menšie než počet bitov na vstupe, je celkom rozumné hovoriť, že výstup je (v nejakom zmysle) kompresovaná forma vstupu. □

Tento príklad ilustruje, že môžeme očakávať, že učiaci algoritmus dáva na výstup reprezentáciu ω hypotézy h_ω takú, že h_ω nie je len rozšírením trénovacej vzorky, ale ω je kompresovaným tvarom vstupu. To je v zmysle, že ω obsahuje tak veľa informácie ako bolo vo vstupnej trén. vzorke, definuje rozšírenú funkciu a vyžaduje menej bitov než trén. vzorka. V ďalšom uvidíme, že ak učiaci algoritmus L dá na výstupe reprezentáciu hypotézy, ktorá nie je príliš dlhá a je signifikantne kompresovaná vzhľadom na vstup, potom L má určité pravdepodobnostné aproximatívne vlastnosti.

1.4 Hľadanie najmenšej konzistentnej hypotézy

Predpokladajme,že je daná tréningová vzorka pre monočlen. Štandardný učiaci algoritmus skonštruuje v polynomiálnom čase hypotézu konzistentnú s tréningovou vzorkou. Avšak môžeme očakávať viac a pýtať sa na najmenší monočlen konzistentný so vzorkou. V tomto kontexte môžeme ignorovať symboly konjunkcie v mnočlene a budeme uvažovať aproximáciu $k \log n$ pre dĺžku monočlena tvoreného k literálmi a definovaného na $\{0,1\}^n$. Teda v ľubovoľnej danej podmnožine M_n najmenší monočlen je taký, ktorý má najmenší počet literálov. Ukážeme, že problém nájdenia najmenšieho monočlena konzistentného s tréningovou vzorkou je NP-ťažký problém.

Náš cieľ bude dosiahnutý pomocou známeho NP-úplného problému. Predpokladajme, že U je konečná množina a $\mathbf{S}$ je konečný systém podmnožín množiny U, ktorého zjednotenie pokrýva celú množinu U. Hovoríme, že podsystém $\mathbf{S}'$ systému $\mathbf{S}$ je *podpokrytie*, ak zjednotenie množín v $\mathbf{S}'$ pokrýva celú množinu U. Nasledujúci problém je jeden z prvých problémov, o ktorom bolo dokázané, že je NP-úplný (Karp, 1972).

PODPOKRYTIE

Inštancia: Dvojica $(U, \mathbf{S})$ podľa vyššie uvedenej definície a kladné celé číslo $k \leq |\mathbf{S}|$

Otázka: Existuje podpokrytie pokrytia $\mathbf{S}$ obsahujúce najviac k množín?

Je treba poznamenať, že veľkosť inštancie závisí od $|U| = u$ aj od $|\mathbf{S}| = n$. V skutočnosti môžeme popísať $(U, \mathbf{S})$ pomocou matice veľkosti $u \times n$, v ktorej ku každému prvku (riadok) je vyjadrená píslušnosť k množine v systéme $\mathbf{S}$. Hodnota $u.n$ môže byť považovaná za veľkosť inštancie $(U, \mathbf{S})$ a toto je parameter, ktorého sa týka otázka polynomiálneho algoritmu.

Predhádzajúci problém je zapísaný v existenčnom tvare. Je možné sformulovať optimalizačý problém v nasledujúcom tvare:
PODPOKRYTIE
Inštancia: Dvojica $(U, \mathbf{S})$ podľa vyššie uvedenej definície a kladné celé číslo $k \leq |\mathbf{S}|$
Otázka: Aká je veľkosť minimálneho podpokrytia pre $(U, \mathbf{S})$?

Je zrejmé, keď máme odpoveď na druhý problém čas behu je polynomiálny v $u.n$, tak je zodpovedaný aj prvý problém v polynomiálnom čase.
Poznámka: urobené na cvičeniach.

1.5 OCCAM algoritmy

Nech $\Omega \to H$ je reprezentácia booleovských funkcií. Nech $\|\omega\|$ je miera veľkosti definovanej pre každé $\omega \in \Omega$. Pre každé $r \geq 1$ definujeme

$$\Omega_r = \{\omega \in \Omega | \quad \|\omega\| = r\}$$

a nech H_r označuje podmnožinu H obsahujúcu tie hypotézy h_ω, ktorých minimálna reprezentácia má veľkosť r.

Hovoríme, e také hypotézy majú veľkosť reprezentácie r.

Potom H môže byť odstupňované pomocou veľkosti reprezentácie $H = \bigcup H_r$. Učiaci algoritmus L pre H má vstup - tréningovú vzorku pre nejakú cieľovú funkciu $t \in H$. Predpokladajme, že $t \in H_r$; inak povedané najmenšia reprezentácia t má veľkosť r. Výstup z L bude špecifikovaný reprezentáciou $\omega \in \Omega_q$. Potrebujeme uvažovať vzťah medzi q a r. Na základe výsledkov predchádzajúceho odseku môže byť ťažké nájsť najmenšiu možnú hodnotu q, ale môžeme požadovať nájdenie nie najkratšej možnej reprezentáciu, ale dostatočne krátku. Táto idea je presnejšie vyjadrená v nasledujúcej defincii Blumera (1987).

Definícia 1.5.1 *Hovoríme, že učiaci algoritmus L pre H je* ***Occam*** *vzhľadom na reprezentáciu $\Omega \to H$, ak*

- *L je konzistentný*
- *k danej tréningovej vzorke $\overline{s}$ dĺžky m pre cieľovú hypoézu $t \in H_r$ výstupná hypotéza $L(s) = h_\omega$ je taká, že $\|\omega\| \leq m^\alpha \cdot r^\beta$, kde $0 < \alpha < 1$ a $\beta \geq 1$ sú konštanty.*

Hranica pre $\|\omega\|$ hovorí, že výstup je komprimovaný vzhľadom na dĺžku tréningovej vzorky a rastie len polynomilne s veľkosťou minimlnej reprezentcie dĺžky cieľovej hypotézy. Podmienka $\alpha < 1$ znamen, e vstup je vlastne komprimovan tvar vstupu; ak povolme $\alpha = 1$, potom vstup by bol porovnateľn s veľkosťou tréningovej vzorky, ktorej bitová dĺžka je lineárna v m a žiadna signifikantná komprimácia by nebola dosiahnutá. Nasledujúca veta ukazuje, že výstup krátkej reprezentácie v tomto zmysle, postačí pre PAC učenie.

Aby sme sformulovali túto vetu, vráťme sa k našej originálnej definícii učiaceho algoritmu s konceptovým a hypotézovým priestorom ako rôznymi. Tento rozdiel je tu použiteľný, pretože sme sa zaujímali o hypotézy v H_r použitím plného zdroja H.

Veta 2 *Nech H je priestor booleovských funkcií s reprezentáciou $\Omega \to H$, nech $H = \bigcap H_r$ je odstupňovaný veľkosťou reprezentácie. Ak L je Occam učiaci algoritmus vzhľadom na danú reprezentáciu, potom pre každé r, L existuje PAC učiaci algoritmus pre (H_r, H) so zložitosťou vzorky $m_L(H_r, \delta, \epsilon)$ polynomiálnou v r, δ^* a ϵ^{-1}.*

Dôkaz: Predpokladajme, že sú dané δ, ϵ, μ a $t \in H_r$. Pre každé dané m nech $L(m, t)$ označuje množinu hypotéz $h \in H$ takú, že h je výstup $L(\overline{s})$ algoritmu L pre nejakú tréningovú vzorku $\overline{s}$ dĺžky m a cieľový koncept t. Inak povedané, $L(m, t)$ je efektívny hypotézový priestor pre t. Podľa 2. podmienky Occam

algoritmu členmi $L(m,t)$ sú hypotézy h_ω, pre ktoré ω má najviac $M = \lfloor m^\alpha \cdot r^\beta \rfloor$ bitov, a celkový počet takých ω je najviac 2^{M+1}. Odtiaľ
$$|L(m,t)| \leq 2^{m^\alpha \cdot r^\beta + 1}.$$
Poznamenajme, že hranica závisí len od r nie od t samotnej; inak povedané, platí uniformne pre všetky $t \in H_r$. Teraz zopakujeme argument daný v predchádzajúcom odseku. Pravdepodobnosť, že ľubovoľná daná ϵ-zlá hypotéza z H súhlasí s t na tréningovej vzorke dĺžky m je $(1-\epsilon)^m$. Pretože L je konzistentný, jeho výstupné hypotézy súhlasia s tréningovouou vzorkou a teda pravdepodobnosť, že výstupná hypotéza je ϵ-zlá je najviac
$$|L(m,t)| \cdot (1-\epsilon)^m \leq 2^{m^\alpha \cdot r^\beta + 1}(1-\epsilon)^m.$$
Zostáva dokázať, že toto môže byť $< \delta$, ak vezmeme dostatočne veľké m a že m je polynóm v r, δ a ϵ^{-1}. Použitím nerovnosti $(1-\epsilon)^m < e^{-\epsilon \cdot m}$ a preusporiadaním dostávame, že $\epsilon \cdot m \geq A \cdot m^\alpha + B$, kde $A = r^\beta \cdot \ln 2$ a $B = \ln(\frac{2}{\delta})$.
Pretože $\alpha > 1$, podmienka platí ak
$$m^{1-\alpha} \geq (A+B)/\epsilon, t.j. \quad m \geq m_0 = \left\lceil \left(\frac{A+B}{\epsilon}\right)^{1/(1-\alpha)} \right\rceil.$$

Inak povedané, výraz pre m_0 je horná hranica pre zložitosť vzorky. Zrejme m_0 je polynóm v r, pretože A je $O(r^3)$ a tak m_0 je $O(r^{\beta/(1-\alpha)})$; alej m_0 je tiež polynóm v δ^* a ϵ^{-1}.
□

Zdôrazňujeme znovu podmienku $\alpha < 1$; je zrejmé, že podmienka $\epsilon \cdot m > A \cdot m^\alpha + B$ môže byť splnená, ak $\alpha = 1$.

Existuje bezprostredný dôsledok tejto vety, že ak čas behu Occam algoritmu je polynomiálny v m, potom čas behu PAC učiaceho algoritmu je polynomiálny v r, δ^* a ϵ^{-1}. Inak povedané, Occam algoritmus L pre H PAC učí každý H_r podľa H a urobí to efektívne vzhľadom na veľkosť reprezentácie a δ a ϵ. Poznamenajme, že nemusí nutne platiť, že H samotný je PAC naučiteľný aj keď by to mohlo tak byť, ak existuje horná hranica na veľkosť reprezentácie hypotézy v H.

1.6 Príklady Occam algoritmov

Predpokladajme, že je daný systém $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$ konečných množín $U = \bigcup_{i=1}^{n} S_i$. Chceme nájsť najmenšie podpokrytie $(U, \mathbf{S})$; t. j. najmenší podsystém $\overline{\mathbf{S}}$, ktorého zjednotením je U. Videli sme, že tento problém je NP-ťažký. Neznamená to, že neexistujú efektívne prostriedky na dosiahnutie aproximatívneho riešenia pre tento problém. Existuje jednoduchá intuitívna metóda na nájdenie aproximatívneho riešenia, založená na "greedy" metóde, ktorá sa zdá byť veľmi účinnou. Najprv vyberieme množinu S_{j_1}, ktorá obsahuje najväčší počet prvkov z U a odstránime ju z $\overline{s}$. Potom vyberieme S_{j_2}, ktorá obsahuje najväčší počet zvyšných prvkov, at. Pokračujeme týmto spôsobom, v každom kroku vyberieme množinu, ktorá obsahuje najväčší počet zvyšných prvkov.

Greedy algoritmus pre minimálne pokrytie.

```
 set X=U;
while X≠ ∅ do
begin choose S_j such that |S_j ∩ X| is maximal;
set X=X-S_j;
end;
```

Pretože $\overline{\mathbf{S}}$ pokrýva U, proces musí skončiť s podpokrytím $S' = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$. Samozrejme, veľkosť k výsledného podpokrytia nebude vo všeobecnosti najmenším možným podpokrytím, ale bolo ukázané (Nigmatullin (1969), Yohnsson (1974)), že platí nasledujúci vzťah: $k \leq l \cdot (\ln|U| + 1)$, kde l je veľkosť minimálneho podpokrytia.

Toto poskytuje dobrú hornú hranicu pre **pomer výkonnosti** $\frac{k}{l}$ a v tomto zmysle greedy algoritmus je dobrá aproximácia pre problém.

čas behu závisí od $u = |U|$ a $n = |S|$, $\quad u \leq n, \quad k \leq \min(u,n) \quad \Rightarrow k \leq n \cdot (\ln u + 1)$.

Každý výberový krok obsahuje nájdenie maxima najviac n celých čísel a vymazanie najviac u prvkov z každej z najviac n množín. Počet operácií $O(u \cdot n)$. Celkový čas behu je $O(u \cdot n \cdot \min(u, n))$.

Greedy metóda môže byť použitá na odvodenie algoritmov pre určité triedy booleovských formúl. Podľa Hausslera (1988) ukážeme technicky ako greedy algoritmus pre pokrytie môže byť použitý na priestor M_n-jednočlenov, ukážeme, že výsledný učiaci algoritmus je Occam.

Počiatočná hypotéza je jednočlen bez literálov, identicky 1-ková funkcia. V každom kroku je pridaný 1 literál do priebežnej konjunkcie literálov podľa pravidla založeného na greedy algoritme pre pokrytie.Budeme hovoriť, že literál λ *eliminuje* negatívny príklad x ak $\langle\lambda\rangle(x) = 0$. Vezmeme prvky, ktoré budú pokryté ako množinu yáporných príkladov v danej tréningovej vzorke a pokrývajúce množiny ako množiny záporných príkladov eliminovaných literálmi určitého druhu. V každom stave vyberieme literál, ktorý eliminuje najväčší počet záporných príkladov zo vzorky, pridáme tento literál do formuly a vymažeme príklady, ktoré eliminuje.

Prečo toto pracuje?

Nech $\overline{s}$ je tréningová vzorka pre monočlen a E nech je množina príkladov v $\overline{s}$ takých, že $E = E^+ \cup E^-$,
Pre ľubovoľný literál λ položíme $S_\lambda = \{x \in E^- | \quad \langle\lambda\rangle(x) = 0\}$
Nakoniec, nech

$$\Lambda = \{\lambda| \quad \langle\lambda\rangle(x) = 1 \quad \forall x \in E^+\}$$

Lema 1 *Systém množín* $\mathbf{S} = \{S_\lambda| \quad \lambda \in \Lambda\}$ *pokrýva* E^-.

Dôkaz: Pretože $\mathbf{s}$ je tréningová vzorka pre monočleny, vieme, že existuje monočlen $t = \langle\lambda_1 \wedge \ldots \wedge \lambda_l\rangle$ taký, že pre $x \in E, t(x)$ je 1 alebo 0 podľa toho či x je v E^+ alebo v E^-. Toto implikuje, že $\lambda_1, \ldots, \lambda_l$ všetky patria do Λ. A tiež, že pre ľubovoľné $x \in E^-$ aspoň jeden z literálov λ_j vyskytujúcich sa v t je taký, že $\langle\lambda_j\rangle(x) = 0$. Inak povedané $x \in S_{\lambda_j} \in \mathbf{S}$.
□

Lema 2 *Ak* $\mathbf{S}' = \{S_{\lambda_1}, \ldots, S_{\lambda_k}\}$ *je ľubovoľné podpokrytie* (E^-, S), *potom monočlen* $h = \langle\lambda_1 \wedge \ldots \wedge \lambda_k\rangle$ *je konzistentný s* $\overline{s}$.

Dôkaz: Predpokladajme, že $x \in E^+$. Pretože $\lambda_1, \ldots, \lambda_k$ sú členmi Λ, na x dávajú hodnotu 1, teda $h(x) = 1$. Ak $x \in E^-$, potom pretože S' je podpokrytie, existuje j, $1 \leq j \leq k$ také, že $x \in S_{\lambda_j}$. Teda $\langle\lambda_j\rangle(x) = 0$, a teda $h(x) = 0$.
□

Tieto lemy ukazujú, že greedy algoritmus pre problém pokrytia môže byť transformovaný na algoritmus pre hľadanie jednočlenov konzistentných s danou tréningovou vzorkou.

Aby sme zreteľne videli, že je to Occam algoritmus, uvažujme jeho chovanie sa na tréningovej vzorke pre monočlen t, ktorého najmenšia reprezentácia je pomocou formuly, ktorá obsahuje l literálov. Minimálna reprezentácia t je veľkosti $r = \lceil l \cdot \log n \rceil$. Výsledok práce greedy algoritmu pre problém pokrytia implikuje, že počet k literálov vo výstupnej formule je taký, že $k \leq l \cdot (\ln |E^-| + 1)$.Preto veľkosť výstupnej formuly splňuje

$$\|\omega\| = \lceil k \cdot \log n \rceil \leq \lceil l \cdot (\ln |E^-| + 1) \cdot \ln n \rceil \leq r \cdot (\ln |E|^- + 1)$$

Platí, že $|E|^- \leq m$.
$\|\omega\| \leq r \cdot (\ln m + 1)$, čo triviálne implikuje Occam kompresnú podmienku.
$\|\omega\| \leq m^\alpha \cdot r^\beta \quad \alpha = \frac{1}{2}, \quad \beta = 1.$
□

Greedy algoritmus sa líši evidentne od standardného algoritmu pre monočleny. Namiesto začiatku s identicky nulovou funkciou (konjunkcia $2n$ literálov) a nasledujúcim vymazaním literalov použitím kladných príkladov greedy algoritmus štartuje s funkciou identicky rovnou 1 (prázdna konjunkcia literálov) a potom pridáva literály použiím yáporných príkladov. Teda, pokým štandardný algoritmus je bezpamäťový on-line algoritmus, greedy algoritmus určite taký nie je. Avšak greedy algoritmus ako Occam algoritmus má dôležitú výhodu v tom, že jeho výstupom sú konzistentné hypotézy, ktoré sú relaívne jednoduché.

1.7 Epac učenie

Predpokladajme, že $H = \bigcup H_n$ je hypotézový veľkosťou príkladov odstupňovaný priestor a $\Omega \rightarrow H$ je reprezentácia pre H. Potom by sme mohli odstupňovať každé H_n pomocou veľkosti reprezentácie takto $H_n = \bigcup H_{n,r}$, kde $H_{n,r}$ pozostáva z tých hypotéz H_n, ktoré majú minimálnu veľkosť reprezentácie r. Teda

$$H = \bigcup_n \bigcup_r H_{n,r}$$

je *dvojite odstupňovaný.*

Nech L je učiaci algoritmus pre H v obvyklom zmysle, že $L(\overline{s})$ je v H_n vždy, keď $\overline{s}$ je tréningová vzorka pre hypotézy v H_n. Hovoríme, že L je *efektívny PAC* alebo *ePAC* ak (Valiant, 1991)

- čas behu $R_L(m, n)$ je polynomiálny v m aj v n;
- zložitosť vzorky $m_L(H_{n,r}, \delta, \epsilon)$ je polynomiálna v n, r, δ^* a ϵ^{-1}.

Teda $ePAC$ učiaci algoritmus zaručuje, že vydá pravdepodobnostne aproximovaný správny výstup s časom behu polynomiálnym v n, r, δ^* a ϵ^{-1}.

Jeden spôsob na zaručenie druhej podmienky je použitie nejakej verzie Occam podmienok. V tomto kontexte hovoríme, že L je Occam ak podmienky stanovené v definícii pre Occam algoritmus platia pre každé H_n s konštantami α a β nezávislými od n. Potom máme nasledujúci výsledok.

Veta 3 *Predpokladajme, že hypotézový priestor je $H = \bigcup H_{n,r}$ ako bolo uvedené vyššie a L je Occam algoritmus pre učenie $H_{n,r}$ pomocou H_n s polynomiálnym časom behu $R_L(m, n)$. Potom L je $ePAC$.*

Dôkaz: Z dôkazu vety vyššie uvedenej máme hornú hranicu

$$m_0(H_r, \delta, \epsilon) = \left[\left(\frac{A + B}{\epsilon} \right)^{1/(1-\alpha)} \right]$$

pre zložitosť vzorky algoritmu L na $H_{r,n}$, kde $A = r^\beta \ln 2$ a $B = \ln(2/\delta)$. Ako sme už poznamenali, toto je polynóm v r, δ^* a ϵ^{-1}. Pretože α a β sú nezávislé od n, aj $m_0(H_r, \delta, \epsilon)$ je nezávislé od n. Výsledok vyplýva z toho, že horná hranica na čas behu algoritmu L v PAC učení $H_{n,r}$ je

$$R_L(m_0(H_r, \delta, \epsilon), n)$$

čo je polynóm v n, r, δ^* a ϵ^{-1}.
$\square$

Príklad: Odstupňovaný hypotézový priestor $M = \bigcap M_n$ monočlenov môže byť odsupňovaný dvojite ako $M = M_{n,r}$, kde $M_{n,r}$ pozostáva z tých monočlenov n premenných, ktoré majú veľkosť reprezentácie r. V predchádzajúcom odseku bol popísaný algoritmus pre učenie $M_{n,r}$ pomocou M_n založený na greedy metóde a ukázali sme, ze má Occam vlastnosť pri $\alpha = 1/2$ a $\beta = 1$. Čas behu $R_L(m,n)$ je $O(m.n.min(m, n))$, čo je určite polynóm v m a n. Z toho nám vyplýva, že greedy algoritmus pre M je $ePAC$.
$\square$

1.8 Ďalšie poznámky

Ako sme už uviedli, fakt, že C^k nie je efektívne naučiteľný vzhľadom na veľkosť príkladov, je výsledok, ktorý závisí od reprezentácie. Keby výstupné hypotézy mohli byť reprezentované iným spôsobom než konjunkcie najviac k klauzúl, tak generovanie pravdepodobnostne aproximatívne korektných hypotéz by mohlo byť jednoduchšie. Kearns a Valiant (1989) v tomto smere dosiahli veľmi silný výsledok založený na kryptografických predpokladoch.

1.9 Úlohy

1. Ukážte, že $C_n^k \subseteq D_{n,k}$ pre všetky k a n a že inklúzia je striktná pre niektoré hodnoty n a k.

2. Sformulujte problém pokrytia množiny, ktorý zodpovedá nájdeniu najkratšieho monočlena konzistentného s nasledujúcimi príkladmi.

$$E^+ = \{1110011, 1111011, 1011001, 1011011, 1110001\};$$

$$E^- = \{1010100, 0111011, 0001111, 1001010, 0101111, 1100000\}.$$

Riešte problém pokrytia a napíšte najkratší monočlen.