Obsah

Kapitola 1

Úvod do teórie strojového učenia

V oblasti teórie strojového učenia sa zaoberáme učením; formalizujeme ho, navrhujeme rôzne algoritmy schopné tohto učenia, analyzujeme ako rýchlo sa tieto algoritmy učia, ... Čo to ale vlastne je to "učenie", a ako sa nad týmto konceptom dá zamýšľať?

Začnime tým, že si vybavíme, čo všetko sa nám spája s učením. Napríklad učenie sa na skúšky, ktoré pre niektorých vyzerá tak, že sa snažia si zapamätať všetkých 80 strán skrípt naspamäť. Alebo keď sa snažíme naučiť novú skladbu na klavíri: hráme ju znova a znova, až kým v tom nie sme dobrí. Alebo trénovanie na dôležitý zápas vo futbale, ...

Vidíme teda, že učenie je zložitý koncept. Konkrétne detaily toho, ako presne sa učíme (resp. prečo to funguje), sú známe málo ľuďom. Avšak vidíme, že jednotlivé "učenia" mali niečo spoločné: nejaká aktivita sa opakovala veľakrát, a čím viackrát sa zopakovala, tým lepší bol výkon. Zároveň ale nechceme, aby sa zakaždým odohralo to isté; nová skúsenosť je dôležitou súčasťou učenia.

Ak by sme len tak vychrlili nejakú definíciu, ktorá nám príde rozumná, riskujeme, že nebude "sedieť s našou intuíciou", alebo nebude dostatočne všeobecná. Kvôli tejto zložitosti teda namiesto toho, aby sme zachytili "učenie" v jednej definícii, študujeme veľa rôznych modelov učenia, ktoré sú aplikovateľné v rôznych kontextoch. Pod "modelom" rozumieme akúsi hračku, zjednodušený mentálny obraz, s ktorým sa jednoducho pracuje a zároveň "sedí s našou intuíciou". Takto síce nijak neručíme, že pokrývame "všetko učenie"; čím viac viet a teorém v rôznych modeloch ale dokážeme, tým lepší budeme mať obraz o tom, čo to učenie je.

My sa v týchto skriptách budeme zaoberať hlavne oblasťou *učenia s učiteľom*. Uvedieme hlavný model (tzv. *statistical learning framework*), s ktorým budeme pracovať, a v prípade potreby rozširovať/dolaďovať.

1.1 Model

Predstavme si, že chceme vedieť na základe nejakých vstupných dát (ktoré budeme spravidla označovať x) predpovedať výstupné dáta (označované y). Napríklad chceme na základe rozlohy bytu vedieť predpovedať jeho cenu. Alebo vedieť z obrázku (vo formáte 32×32 čiernobielych pixelov) povedať, či sa v ňom nachádza mačka alebo nie.

Snažíme sa teda zachytiť nejaký vzťah medzi dátami x a y. To, akým spôsobom to budeme robiť, je nasledovné: nejakým procesom P získame t *trénovacích príkladov* $(x_1, y_1), \ldots, (x_t, y_t)$. Na základe týchto príkladov budeme chcieť navrhnúť nejakú funkciu h, ktorá bude vedieť podľa vstupu x predpovedať výstup y s dostatočnou presnosťou.

Prirodzený spôsob, akým môžeme merať úspešnosť funkcie h, je podľa toho, ako sa jej darí na našich t príkladoch. Prístupu, kde hľadáme funkciu, ktorej sa čo najlepšie darí na trénovacích príkladoch, hovoríme *empirical risk minimization* (skrátene ERM). Dobrá funkcia h ale bude schopná aj *generalizovať*: bude sa jej dariť na ďalších dátach, ktoré vieme získať tým istým procesom P.

Schopnosť generalizácie je jednou z najdôležitejších vlastností, ktoré od trénovania pomocou techník strojového učenia požadujeme. Predstavme si, že by sme funkciu h zostrojili tak, že si "zapamätáme" všetky trénovacie príklady: ak je na vstupe x, ktoré bolo medzi trénovacími príkladmi, vrátime príslušné y. Takáto funkcia h má zrejme nulovú chybu na trénovacích príkladoch (pokiaľ pre jedno x existuje vždy len jedno správne y), ale mimo nich sa jej vôbec nebude dariť.

Vidíme teda, že ERM vo svojej holej podstate nefunguje. Problém je v tom, že neobmedzujeme to, aké funkcie považujeme sa "kandidátov": nič nezaručuje, že informácia z trénovacích príkladov sa prenesie aj na ostatné príklady. Obmedzíme teda množinu funkcii, ktoré uvažujeme, na nejakú užšiu skupinu, ktorú nazveme *množina hypotéz*.

Napríklad si predstavme, že hľadáme funkciu $f : \mathbb{R} \to \mathbb{R}$ a dostali sme trénovacie príklady $(0, 0)$ a $(1, 1)$. Ak o probléme nič ďalšie nevieme, tak nevieme povedať nič o tom, ako sa bude f správať pre $x \notin \{0, 1\}$. Ak ale vieme, že hľadáme neklesajúcu funkciu s oborom hodnôt $\{0, 1\}$, tak vieme, že $f(x) = 0$ pre $x \leq 0$, $f(x) = 1$ pre $x \geq 1$, a niekde v intervale $(0, 1)$ sa to "láme".

Tým, že sme obmedzili množinu hypotéz, sme v podstate zaviedli *inductive bias*: predpoklady, ktoré využívame na to, aby sme odvodili výsledky aj na tých vstupoch, ktoré sme ešte nevideli.

1.2 Definície a označenia

Množinu všetkých vstupov označíme X, niekedy jej budeme hovoriť aj *množina inštancii*. Množinu všetkých výstupov označíme Y.

Množinu všetkých trénovacích príkladov budeme volať *trénovacia množina*, označíme ju T. (Z matematického hľadiska to ale nie je množina, skôr multimnožina, kedže jedna dvojica (x, y) sa medzi trénovacími príkladmi môže vyskytovať viackrát. Pre jednoduchosť to ale budeme nazývať množinou.) Počet trénovacích príkladov označíme t.

Proces, ktorým získavame dáta (x, y), vieme formalizovať ako pravdepodobnostné rozdelenie P nad priestorom $X \times Y$, ktoré každému možnému pozorovaniu priradí nejakú pravdepodobnosť (resp. hustotu pravdepodobnosti v prípade spojitého rozdelenia). Trénovacie príklady získame ako t nezávislých vzoriek z rozdelenia P.

Postup, ktorým zostrojíme funkciu h, vieme formalizovať ako algoritmus, ktorý na vstupe dostane niekoľko trénovacích príkladov $(x_1, y_1), \ldots, (x_t, y_t)$, a na výstupe vráti funkciu $h : X \to Y$, ktorú budeme volať *hypotéza*. Množinu hypotéz označíme H.

V tejto kapitole sa nebudeme zaoberať výpočtovou stránkou strojového učenia, takže od detailov ako časová zložitosť, spôsob hľadania optimálneho riešenia a pod. abstrahujeme.

1.2.1 Meranie chyby hypotézy

Ako vyjadriť mieru toho, že sa hypotéze "dobre darí"? Spravíme tak pomocou *chybovej funkcie* $\mathrm{err} : Y \times Y \to \mathbb{R}_0^+$, ktorej význam je nasledovný: $\mathrm{err}(y, y')$ vyjadruje, ako veľmi sa od seba líšia výstupy y a y'.

Chyba na jednom príklade. Keď dostaneme nejaký príklad (x, y), vieme zmerať, ako veľmi sa naša hypotéza h na tomto vstupe pomýlila, ako $\mathrm{err}(h(x), y)$.

Očakávaná testovacia chyba hypotézy (OTeChHyp). Celkovú chybu našej hypotézy potom vieme zmerať ako očakávanú chybu na náhodne vybranom príklade z rozdelenia P. Túto chybu budeme označovať Err, a vypočítame ju nasledovne:

$$\mathrm{Err}(h, P) = \mathop{\mathrm{E}}_{(x,y)\sim P} \left[\mathrm{err}(h(x), y)\right].$$

Častokrát ale bude rozdelenie P jasné z kontextu, v takom prípade budeme skrátene písať $\mathrm{Err}(h)$.

Trénovacia chyba. Niekedy nás bude zaujímať ale aj chyba, ktorú hypotéza nadobúda na

trénovacej množine. Budeme ju tiež označovať Err, a vypočítame ju nasledovne:

$$\mathrm{Err}(h,T) = \mathop{\mathrm{E}}_{(x,y)\in T}[\mathrm{err}(h(x),y)] = \frac{1}{t}\cdot\sum_{i=1}^{t}\mathrm{err}(h(x_i),y_i).$$

Tento zápis sa dá chápať aj tak, že trénovaciu množinu interpretujeme ako pravdepodobnostné rozdelenie, pričom pravdepodobnosť každého príkladu je úmerná tomu, koľkokrát sa v množine vyskytuje. Niekedy túto chybu budeme zapisovať aj $\mathrm{Err}_T(h)$.

1.2.2 Meranie chyby algoritmu

Hypotéza je iba výstupom trénovacieho algoritmu, a závisí od toho, akú konkrétnu trénovaciu množinu dostaneme a od prípadnej náhodnosti samotného algoritmu. Ak by sme chceli porovnať dve rôzne algoritmy, na úrovni výstupných hypotéz to nevieme spraviť: pre jednu trénovaciu množinu môže byť lepší prvý algoritmus, pre inú množinu zas druhý. Môžeme ale hovoriť o tom, ako dobré budú tie algoritmy v priemere, ak vezmeme do úvahy všetky možné náhodné faktory.

Trénovací algoritmus označíme L (z anglického *learner*). Jeho výstup na trénovacej množine T označíme $L(T)$; prípadne $\hat{h}_T$, alebo len $\hat{h}$ ak bude z kontextu jasné, aká je trénovacia množina.

Očakávaná testovacia chyba algoritmu (OTeChAlg). Celkovú chybu algoritmu na t trénovacích príkladoch zmeriame ako OTeChHyp, ktorú získame natrénovaním na náhodne zvolenej trénovacej množine veľkosti t. Túto chybu budeme opäť značiť Err, vypočítame ju ako

$$\mathrm{Err}(L,P)[t] = \mathop{\mathrm{E}}_{T\sim P^t}[\mathrm{Err}(L(T),P)] = \mathop{\mathrm{E}}_{T\sim P^t}\left[\mathrm{Err}(\hat{h}_T)\right].$$

Očakávaná trénovacia chyba algoritmu (OTrChAlg). Niekedy nás bude zaujímať aj to, akú môžeme očakávať trénovaciu chybu výstupnej hypotézy. Uvedomme si, že táto hodnota sa bude líšiť od očakávanej testovacej chyby: totiž výstupná hypotéza priamo závisí od trénovacej množiny T, ale iba nepriamo závisí od pravdepodobnostného rozdelenia P. (Ako príklad uvedieme algoritmus "zapamätaj si trénovacie dáta".) Túto chybu budeme značiť Err_T, a vypočítame ju nasledovne:

$$\mathrm{Err}_T(L,P)[t] = \mathop{\mathrm{E}}_{T\sim P^t}[\mathrm{Err}(L(T),T)] = \mathop{\mathrm{E}}_{T\sim P^t}\left[\mathrm{Err}_T(\hat{h}_T)\right].$$

Pri oboch týchto chybách, pokiaľ bude z kontextu jasné rozdelenie P alebo počet príkladov t, zo zápisu ich vynecháme. Teda v najkratšej forme budeme OTeChAlg označovať $\mathrm{Err}(L)$ a OTrChAlg budeme značiť $\mathrm{Err}_T(L)$. Rozlišovať medzi chybami hypotéz/algoritmov a trénovacími/testovacími chybami budeme podľa kontextu (t.j. či je v indexe $_T$ a či je v zátvorke hypotéza h alebo algoritmus L).

1.2.3 Príklady chybových funkcii

Uvedieme ešte príklady používaných chybových funkcii. Pri klasifikácii sa naša hypotéza vždy buď trafí, alebo netrafí do správnej odpovede. Neexistuje ale nejaká miera toho, ako veľmi sa netrafila, resp. ako blízko bola ku správnej odpovedi. Dáva teda zmysel každej správnej odpovedi priradiť chybu 0, a každej nesprávnej odpovedi chybu 1:

$$\mathrm{err}(y,y') = \begin{cases} 0, & \text{ak } y = y' \\ 1, & \text{inak} \end{cases}.$$

Potom sa nám viacero vzorcov pre chyby zjednoduší: testovacia chyba hypotézy je

$$\mathrm{Err}(h) = \mathop{\mathrm{E}}_{(x,y)\sim P}[\mathrm{err}(h(x),y)] = \Pr_{(x,y)\sim P}(h(x)\neq y),$$

a podobne sa dá zjednodušiť aj trénovacia chyba.

Pri regresii naopak takáto miera existuje, dokonca by sa dalo povedať, že každá takáto miera zodpovedá jednej chybovej funkcii. Bežnými voľbami sú kvadratická chyba $(y-y')^2$ a absolútna chyba $|y-y'|$.

1.2.4 Ďalšie označenia

Pri zápise stredných hodnôt budeme vynechávať to, odkiaľ premenné pochádzajú, pokiaľ to bude z kontextu jasné. Konkrétne zavedieme nasledovné skrátené zápisy:

- ak sa stredná hodnota berie cez príklady z rozdelenia P:

$$\mathop{\mathrm{E}}_{(x,y)\sim P} \equiv \mathop{\mathrm{E}}_{x,y}$$

- ak sa stredná hodnota berie cez všetky trénovacie príklady z množiny T:

$$\mathop{\mathrm{E}}_{(x,y)\in T} \equiv \mathop{\mathrm{E}}_{x_i,y_i}$$

- ak sa stredná hodnota berie cez všetky možné trénovacie množiny veľkosti t:

$$\mathop{\mathrm{E}}_{T\in P^t} \equiv \mathop{\mathrm{E}}_{T}$$

Podobné skrátené zápisy budeme používať aj pri pravdepodobnostiach a pod.

1.3 Analýza veľkostí chýb

V tejto časti sa podrobnejšie pozrieme na to, ako závisia vyššie uvedené metriky (t.j. OTeChAlg a OTrChAlg) od veľkosti trénovacej množiny t a od veľkosti množiny hypotéz H. V celej časti budeme predpokladať, že úloha je regresného charakteru a chyba sa meria ako kvadratická odchýlka.

1.3.1 Teoretické limity

Uvedomme si, že naša hypotéza musí byť funkciou, teda pre jedno x musí vždy vracať jednu a tú istú hodnotu $y = h(x)$. Rozdelenie P ale môže pre jedno x obsahovať viaceré hodnoty y, pre ktoré je pravdepodobnosť nenulová; napríklad P môže reprezentovať zašumené dáta.

Teda ani najlepšia možná hypotéza-funkcia (nie nutne z H) nemusí mať nulovú chybu. Označme túto hypotézu $h^\square$. Ak by sa nám podarilo vyjadriť jej testovaciu chybu, získali by sme akýsi ireducibilný komponent, ktorý má každá hypotéza; môžeme sa snažiť znížiť iba tú časť chyby, ktorá je "navyše".

Z definície

$$h^\square = \arg\min_h \left(\mathrm{Err}(h)\right) = \arg\min_h \left(\mathop{\mathrm{E}}_{x,y} \left[(h(x) - y)^2\right] \right).$$

Chybu ľubovoľnej hypotézy h vieme upraviť nasledovne:

$$\mathrm{Err}(h) = \mathop{\mathrm{E}}_{x,y} \left[(h(x) - y)^2\right] \tag{1.1}$$

$$= \mathop{\mathrm{E}}_{x} \left[\mathop{\mathrm{E}}_{y|x=x} \left[(h(x) - y)^2\right] \right]. \tag{1.2}$$

Pozrime sa na vnútornú strednú hodnotu ($\mathrm{E}_{y|x}$). V nej je x konštanta, a teda aj $h(x) = c$ je konštanta. Aká konštanta minimalizuje danú strednú hodnotu? Nie je ťažké vidieť (napríklad zderivovaním), že minimum sa nadobúda pre $c = \mathrm{E}_{y|x=x}[y]$. Takže

$$h^\square(x) = \mathop{\mathrm{E}}_{y|x=x}[y],$$

a testovacia chyba najlepšej hypotézy-funkcie teda je

$$\mathrm{Err}(h^\square) = \mathop{\mathrm{E}}_{x} \left[\mathop{\mathrm{E}}_{y|x=x} \left[(y - \mathrm{E}[y])^2\right] \right] = \mathop{\mathrm{E}}_{x} \left[\mathop{\mathrm{Var}}_{y|x=x} (y) \right].$$

1.3.2 Kompromis medzi výchylkou a rozptylom (bias-complexity tradeoff)

V tomto odseku si ukážeme, ako sa dá rozložiť celková OTeChAlg na niekoľko komponentov. To nám dá lepší obraz o tom, čo treba spraviť, ak chceme znížiť OTeChAlg nášho trénovacieho algoritmu L (a dosiahnuť tak lepšie výsledky).

Budeme predpokladať, že výstupom algoritmu je vždy tá hypotéza z množiny H, ktorá je z ERM pohľadu najlepšia, teda dosahuje najmenšiu trénovaciu chybu.

$$\hat{h}_T = \arg\min_{h \in H} \left(\mathrm{Err}_T(h)\right)$$

Označme $h^\star$ tú hypotézu z množiny H, ktorá má najmenšiu testovaciu chybu. Teda

$$h^\star = \arg\min_{h \in H} \left(\mathrm{Err}(h)\right).$$

Uvedomme si, že tieto dve hypotézy sa líšia iba tým, na akom pravdepodobnostnom rozdelení sú optimálne. Hypotéza $\hat{h}_T$ je optimálna na trénovacej množine, ktorá je iba konečnou aproximáciou skutočného rozdelenia P, zatiaľ čo $h^\star$ je optimálna na tomto skutočnom rozdelení. Platia teda nerovnosti

$$\mathrm{Err}_T(\hat{h}_T) \leq \mathrm{Err}_T(h^\star)$$
$$\mathrm{Err}(h^\star) \leq \mathrm{Err}(\hat{h}_T)$$

Toto platí pre ľubovoľnú trénovaciu množinu T. Keď to teda vezmeme dokopy cez všetky možné T, dostaneme "očakávanú" formu týchto nerovností:

$$\mathrm{Err}_T(L) \leq \mathop{\mathrm{E}}_{T}\left[\mathrm{Err}_T(h^\star)\right]$$
$$\mathrm{Err}(h^\star) \leq \mathrm{Err}(L)$$

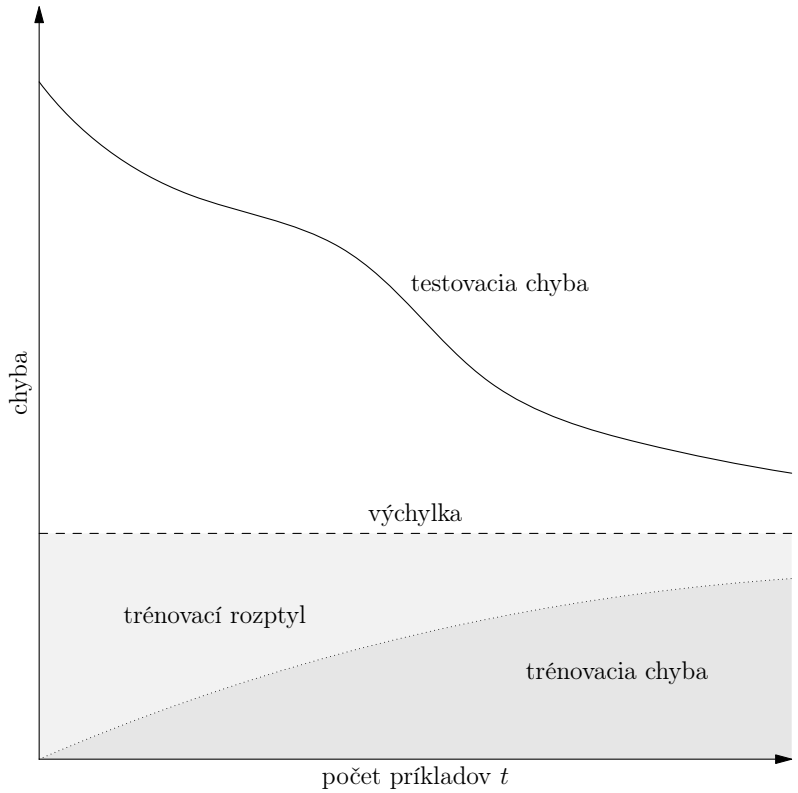
Pritom očakávaná trénovacia chyba $h^\star$ je rovnaká, ako jej očakávaná testovacia chyba, nakoľko $h^\star$ je nezávislé od trénovacej množiny. (Akokoľvek je trénovacia množina vybraná, z pohľadu $h^\star$ to vyzerá tak, akoby sme vybrali niekoľko náhodných príkladov z P.) Dostávame tak

$$\mathrm{Err}_T(L) \leq \mathrm{Err}(h^\star) \leq \mathrm{Err}(L).$$

OTeCh nášho algoritmu sa od najlepšej možnej chyby líši o hodnotu $\epsilon_{\text{est.}} = \mathrm{Err}(L) - \epsilon_{\text{approx.}}$, túto hodnotu budeme volať *rozptyl* (po anglicky *estimation error*). Je spôsobený tým, že sme pri minimalizácii chyby neuvažovali skutočné rozdelenie P (ktoré nepoznáme), ale iba niekoľko trénovacích príkladov. To, čo je najlepšie v T, nie je nutne najlepšie v P. Čím väčšia je ale trénovacia množina, tým viac sa (pravdepodobne) podobá na skutočné rozdelenie P a tým nižší rozptyl.

Prostredný člen budeme volať *výchylka* a označovať $\epsilon_{\text{approx.}}$ (po anglicky *approximation error*). Vyjadruje chybu, ktorá je spôsobená tým, že sa náš algoritmus obmedzil na nejakú konkrétnu množinu hypotéz H. Čím väčšia množina hypotéz, tým menšia výchylka: kedže $h^\star$ je najlepšia hypotéza v množine H, jej zväčšením si môžeme iba prilepšiť. Zložitejšia množina hypotéz sa ale ľahšie "napasuje" na ľubovoľné trénovacie dáta. To zvyšuje riziko toho, že výsledná hypotéza bude špecifická pre trénovacie dáta a nebude schopná generalizácie. Teda čím väčšia množina hypotéz, tým väčší rozptyl. (Neplatí to úplne v každom prípade, dajú sa skonštruovať situácie, kedy zväčšením množiny hypotéz sa rozptyl zachová alebo dokonca zmenší. V prirodzených situáciách to ale do istej miery platí.)

OTrCh nášho algoritmu sa od očakávanej chyby $h^\star$ na trénovacích dátach líši o $\epsilon^T_{\text{est.}} = \mathrm{Err}(h^\star) - \mathrm{Err}_T(L)$, túto hodnotu budeme volať *trénovací rozptyl*. Na skutočnom rozdelení P sa nám nemôže dariť lepšie, ako hypotéze $h^\star$; fakt, že na trénovacích dátach sa nám darí lepšie, je



Obr. 1.1: Závislosť chyby algoritmu od počtu trénovacích príkladov t.

spôsobený tým, že to, čo je najlepšie v P, nemusí byť najlepšie v T. Opäť ale platí, že čím väčšia trénovacia množina, tým viac sa (pravdepodobne) podobá na P a tým nižší trénovací rozptyl.

Na základe dosiaľ uvedeného vieme graficky znázorniť, ako sa zhruba správajú OTeChAlg, rozptyl, výchylka, trénovací rozptyl a OTrChAlg, v závislosti od veľkosti trénovacej množiny (obrázok 1.1) a od zložitosti množiny hypotéz (obrázok 1.2).

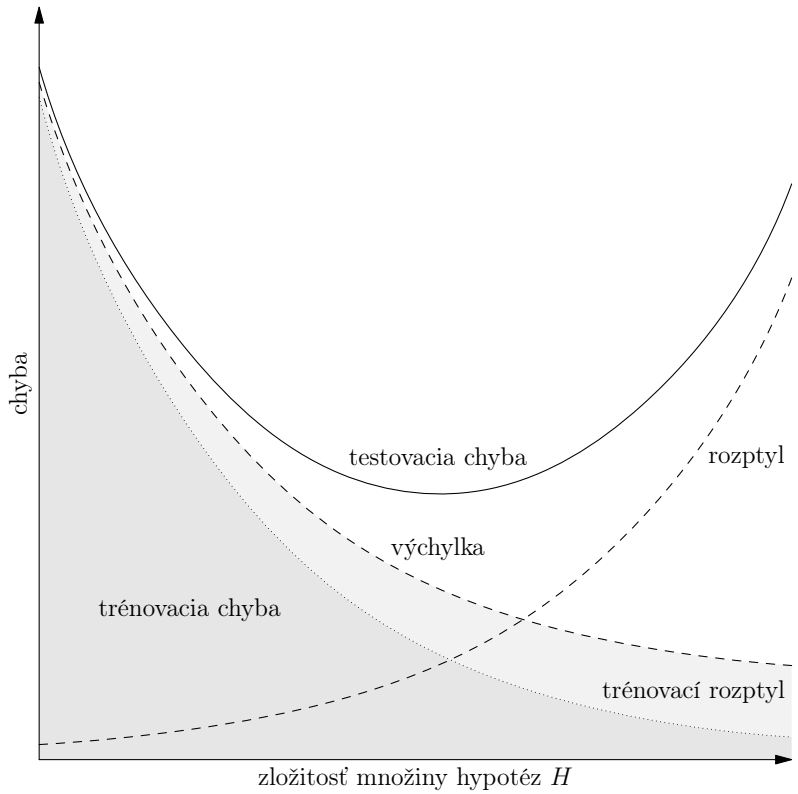
1.4 Podučenie/preučenie

Ak máme fixné trénovacie dáta T, pri voľbe množiny hypotéz H sa snažíme nájsť kompromis medzi malým rozptylom a malou výchylkou. Zložité H bude mať malú výchylku ale veľký rozptyl, čo vedie k tzv. *preučeniu* (angl. *overfitting*). Jednoduché H bude mať malý rozptyl, ale veľkú výchylku, tzv. *podučenie* (angl. *underfitting*).

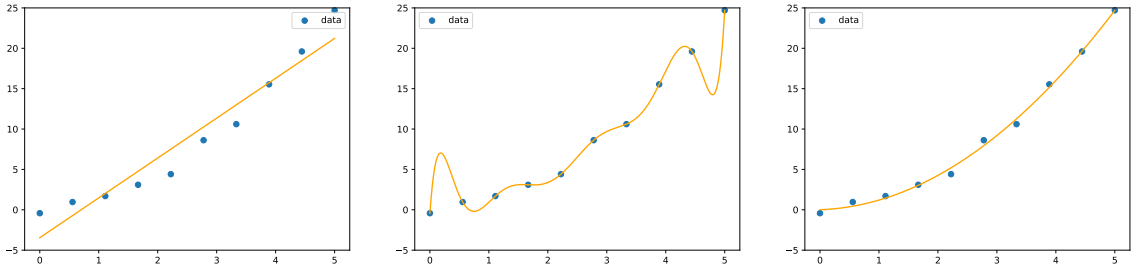
Na obrázku 1.3 ilustrujeme oba koncepty. Úlohou je modelovať kvadratickú funkciu, ku ktorej sme pridali malé množstvo šumu. Rozdelenie P teda vracia nejaké x (napríklad z intervalu $\langle 0, 10\rangle$) a $y = x^2 + \varepsilon$, kde ε je zvolené náhodne z intervalu $\langle -1, 1\rangle$. Ak za množinu hypotéz zvolíme lineárne funkcie, vďaka ich jednoduchosti už pri malých trénovacích množinách bude trénovacia chyba blízko očakávanej testovacej chyby (nízky rozptyl). Za to ale budú všetky chyby vysoké (vysoká výchylka). Ak za množinu hypotéz zvolíme polynómy nejakého vysokého stupňa, ľahko nájdeme polynóm prechádzajúci cez trénovacie dáta (nízka trénovacia chyba), avšak mimo nich bude dávať výsledky úplne mimo (vysoká testovacia chyba, a teda vysoký rozptyl).

1.4.1 Regularizácia

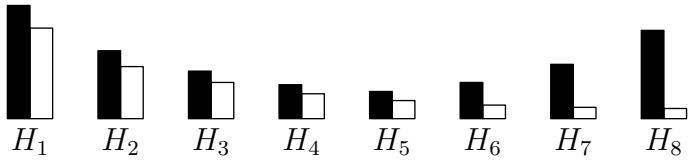
Predstavme si, že máme na výber z viacerých množín hypotéz $H_1, H_2, \ldots$, čím ďalej tým zložitejších. Teda $H_1 \subseteq H_2 \subseteq H_3 \subseteq \ldots$. Ak by sme si graficky znázornili testovacie a trénovacie chyby najlepších hypotéz z jednotlivých množín, vyzeralo by to zhruba ako na obrázku 1.4.



Obr. 1.2: Závislosť chyby algoritmu od zložitosti množiny hypotéz H.



Obr. 1.3: Podučenie, preučenie, akurát.



Obr. 1.4: Testovacie (čierne) a trénovacie (biele) chyby v čím ďalej, tým zložitejších množinách hypotéz H.

Z ktorej množiny hypotéz chceme vybrať? Pri trénovaní sa snažíme nájsť hypotézu h, ktorá minimalizuje chybu na trénovacích dátach. Táto chyba sa ale od očakávanej testovacej chyby líši zhruba o $\epsilon_{\text{est.}} + \epsilon^T_{\text{est.}}$ (v očakávanom prípade).

V prístupe zvanom *regularizácia* do minimalizovaného výrazu umelo pridáme *pokutu*, ktorá aproximuje $\epsilon_{\text{est.}} + \epsilon^T_{\text{est.}}$. Tento člen označíme $\Lambda(h)$. Z čím zložitejšej množiny hypotéza h je, tým väčšia bude pokuta. Výstupom algoritmu potom je

$$\hat{h}_T = \underset{h \in H_1 \cup H_2 \cup \ldots}{\arg\min} \left(\text{Err}_T(h) + \Lambda(h) \right).$$

Uvedomme si, že vrámci jednej množiny H_i ostáva ako najlepšia hypotéza stále tá istá, ako pred zavedením pokuty. V jednej množine sú totiž všetky hypotézy penalizované rovnako, nerobí to teda rozdiel. Penalizácia nám ale umožňuje "férovejšie" porovnávať hypotézy z rôznych množín, nakoľko bez pokuty by na tom boli (neprávom) lepšie zložitejšie hypotézy.

Množiny H_i nemusia byť explicitné, môžu byť implicitne skryté v tom, aký tvar má výraz $\Lambda(h)$. Do jednej množiny patria tie hypotézy, ktoré majú rovnakú penalizáciu.

Vo všetkých prípadoch je pokuta parametrizovaná reálnym parametrom λ hovoriacim, ako veľké pokuty chceme udeľovať. Veľké λ (v porovnaní s trénovacou chybou) nám hovorí, že sa snažíme hlavne dosiahnuť jednoduché hypotézy; s malým λ zas kladieme dôraz na hypotézy s menšou trénovacou chybou.

Uvedieme si niekoľko príkladov výrazov, ktoré sú bežne používané ako pokuta. Budeme predpokladať, že celá množina hypotéz, z ktorej vyberáme (tj. $H_1 \cup H_2 \cup \ldots$) je množina lineárnych funkcii $\mathbb{R}^n \to \mathbb{R}$. Hypotézy majú teda tvar

$$h(x) = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n.$$

- L_2 regularizácia (známa aj ako *ridge regression*).

  $$\Lambda(h) = \lambda \cdot \|(a_1, a_2, \ldots, a_n)\|^2 = \lambda \cdot (a_1^2 + a_2^2 + \ldots + a_n^2)$$

  Táto pokuta "tlačí" váhy nepotrebných atribútov do nuly, pričom väčšie váhy tlačí viac. Takže čím dôležitejší atribút, tým väčšiu váhu si môže dovoliť mať.

- L_1 regularizácia (známa aj ako *lasso*).

  $$\Lambda(h) = \lambda \cdot (|a_1| + |a_2| + \ldots + |a_n|)$$

  Opäť "tlačíme" nepotrebné atribúty do nuly, pričom ale všetky váhy tlačíme rovnako. To nám vie vynulovať nepotrebné atribúty, čo nám vie znížiť výpočtové nároky: nemusíme pri výpočte uvažovať tie členy, ktoré majú nulový koeficient.

1.4.2 Holdout testing

V tomto prístupe si rozdelíme dostupné dáta na dve časti: trénovaciu množinu a *validačnú množinu*. Pomocou validačnej množiny budeme odhadovať testovacie chyby pre jednotlivé množiny hypotéz, na základe ktorých zistíme, ktorá množina hypotéz je pre náš problém najvhodnejšia. Konkrétnejšie:

1. Trénovaciu množinu použijeme na natrénovanie hypotéz z jednotlivých množín.

2. Ako odhad testovacej chyby jednotlivých hypotéz použijeme ich chybu na validačnej množine. Podľa týchto odhadov zistíme, ktorá množina hypotéz je pre náš problém najvhodnejšia.

3. Použijeme všetky dáta, ktoré máme k dispozícii (tj. z trénovacej aj validačnej množiny), na natrénovanie najlepšej možnej hypotézy. Berieme samozrejme do úvahy iba hypotézy z tej množiny hypotéz, ktorú sme identifikovali ako najvhodnejšiu. Výsledná hypotéza je výstupom algoritmu.

V kroku 2 je dôležité, aby bola validačná množina nezávislá od trénovacej. Prečo je to dôležité? Môžeme uvažovať extrémny prípad, keď je validačná množina totožná s trénovacou. Potom ale ako náš "odhad" dostaneme trénovaciu chybu, ktorá rozhodne nie je dobrým odhadom testovacej chyby. Nezávislosť nám teda zaručuje, že odhad získaný na validačnej množine je dobrý.

k-fold evaluation. Pri holdout testovaní je dôležité mať dobrý odhad testovacej chyby pre jednotlivé množiny hypotéz. Dát ale môže byť málo, a v takom prípade môže byť odhad nestabilný/nepresný. Môžeme ale experiment zopakovať niekoľkokrát: v každej iterácii teda zvolíme inú trénovaciu a inú validačnú množinu, a dostaneme iný odhad testovacej chyby. Keď tieto odhady spriemerujeme, dostaneme oveľa presnejší odhad, ako keby sme vykonali iba jednu iteráciu. V tomto konkrétnom prístupe je k iterácii, a množiny sa volia nasledovne: všetky dáta sa rozdelia na k zhruba rovnako veľkých a navzájom nezávislých množín $K_1, K_2, \ldots, K_k$. Následne, v iterácii i sa ako validačné dáta použije množina K_i. Všetko ostatné budú trénovacie dáta.

Leave-one-out. Ak chceme zmerať testovaciu chybu výstupnej hypotézy, musíme si na to rezervovať ďalšiu časť dát: *testovaciu množinu.* Tú nepoužívame ani pri trénovaní, ani pri validácii. Iba úplne na konci celého procesu na nej vypočítame chybu výslednej hypotézy.

1.5 Cvičenia

V nasledujúcich dvoch cvičeniach môžete predpokladať, že trénovací algoritmus vždy vráti nejakú funkciu (nemusí byť len jedna) s minimálnou chybou na trénovacích dátach.

1.1. Je rozumné predpokladať, že s väčším množstvom trénovacích dát sa nám bude testovacia chyba zmenšovať. Sú ale zostrojiteľné situácie, kedy tomu tak nie je. Nájdite jednu takú situáciu.

Konkrétne, nájdite takú množinu hypotéz H funkcii $\mathbb{R}^n \to \mathbb{R}$ a rozdelenie P, pre ktoré sa bude $\mathrm{Err}(\hat{h}_T)$ so zvyšujúcim sa počtom trénovacích chýb *zvyšovať.* Na množinu hypotéz je kladená jedna podmienka: pre každú možnú trénovaciu množinu T musí existovať hypotéza v H, ktorá minimalizuje trénovaciu chybu. (Teda vždy musí existovať minimum, môže ich byť ale viac. Pre všeobecné H vieme povedať iba to, že existuje infimum.)

1.2. Za určitých podmienok ale skutočne platí, že viac trénovacích dát nám vo veľkom merítku neuškodí. Nech množina hypotéz H je konečná a všetky jej funkcie ($\mathbb{R}^n \to \mathbb{R}$) sú ohraničené. Dokážte, že pre $t \to \infty$ sa bude OTeCh hypotézy $\hat{h}_T$ blížiť k OTeCh najlepšej možnej hypotézy $h^\star$. Inak zapísané, dokážte

$$\lim_{t\to\infty} \mathop{\mathrm{E}}_{T} \Big[\mathrm{Err}(\hat{h}_T) - \mathrm{Err}(h^\star)\Big] = 0.$$

1.3. Jednou výhodou L_2 regularizácie oproti L_1 regularizácie je, že sa ľahšie minimalizuje výsledný výraz. Ako príklad uvedieme lineárnu regresiu. V nej je hypotéza parametrizovaná stĺpcovým vektorom $\theta = (\theta_1, \ldots, \theta_n)^T$. Výstupom pre vstup $x = (x_1, \ldots, x_n)$ je $x \cdot \theta$.

Označme X maticu, ktorej riadkami sú vstupy jednotlivých trénovacích príkladov. Ďalej nech y je stĺpcový vektor cieľových výstupov na jednotlivých príkladoch. Je známe, že optimálnymi parametrami lineárnej hypotézy je taký stĺpcový vektor θ, ktorý je riešením rovnice

$$X^T X \cdot \theta = X^T y.$$

Dokáže, že keď k minimalizovanej hodnote pridáme pokutu vo forme $\lambda \cdot \|\theta\|^2$, tak sa optimálnymi parametrami stane θ riešiaca rovnicu

$$(X^T X + \lambda I) \cdot \theta = X^T y.$$

Rozmyslite si tiež, že takéto explicitné vyjadrenie nie je možné priamočiaro získať pre L_1 regularizáciu.

1.6 Appendix: iný kompromis medzi inou výchylkou a iným rozptylom

V literatúre pod názvom *bias-variance tradeoff* vystupuje odlišný výsledok, ako vyššie spomenutý bias-complexity tradeoff. Uvádzame ho, pretože sa často zamieňajú a je v tom zmätok. Pokúsime sa vyjasniť, kde sú rozdiely medzi týmito dvomi výsledkami.

Veta 1.1. *Zamerajme sa na jeden konkrétny vstup x, a merajme chybu výslednej hypotézy $\hat{h}_T$ iba na tomto vstupe. (Stále ale môžeme dostať rôzne y.) Na meranie chyby použijeme kvadratickú odchýlku. Označme očakávanú hodnotu tejto chyby (cez všetky možné trénovacie množiny T a výstupy y) ako* $\mathrm{Err}_{|x=x}(L)$. *Tvrdíme, že sa dá vyjadriť nasledovne:*

$$\mathrm{Err}_{|x=x}(L) = \underbrace{\operatorname*{Var}_T\left(\hat{h}_T(x)\right)}_{\text{rozptyl}} + \underbrace{\left(\operatorname*{E}_T\left[\hat{h}_T(x)\right] - h^{\square}(x)\right)^2}_{\text{výchylka}^2} + \underbrace{\operatorname*{E}_y\left[\varepsilon^2\right]}_{\text{šum}},$$

kde $\varepsilon := y - h^{\square}(x)$.

Zastavme sa najprv nad tým, ako toto tvrdenie interpretovať. Pre každú možnú vzorku trénovacích dát T dostaneme nejakú inú hypotézu $\hat{h}_T$. Rozptyl je nízky práve vtedy, keď budú všetky tieto hypotézy dávať podobné výsledky. Ak je vysoký, znamená to, že výsledná hypotéza je veľmi citlivá na trénovacie dáta; oplatí sa preto zväčšiť ich množstvo. (Ak chceme byť veľmi skeptický, nič nezaručuje, že zväčšením trénovacej množiny sa rozptyl zníži. Možno existuje iný dôvod, kvôli ktorému je vysoký; takmer určite sa dajú skonštruovať takéto umelé protipríklady.)

Keď už uvažujeme všetky možné výsledné hypotézy $\hat{h}_T$, môžme sa pozrieť na to, aký je ich "priemerný odhad": ak by sme spriemerovali všetky ich výsledky, čo by sme dostali? Presne $\mathrm{E}_T[\hat{h}_T(x)]$; výchylka-na-druhú potom meria, o koľko sa takáto priemerná hypotéza líši od najlepšej možnej funkcie $h^{\square}$. Ak je výchylka vysoká ale variancia nízka, znamená to, že takmer všetky $\hat{h}_T$ majú problém na vstupe x, treba teda zvážiť voľbu zložitejšej množiny hypotéz.

Nakoniec, šum zodpovedá ireducibilnej chybe, ktorú bude mať každá hypotéza. Je to v dôsledku toho, že jednému x môže pripadať viacero rôznych y. Presne túto chybu nadobúda najlepšia hypotéza-funkcia $h^{\square}$ (pre ktorú sú rozptyl aj výchylka-na-druhú nulové).

Dôkaz. Upravujme pôvodný výraz.

$$\begin{aligned}
\mathrm{Err}_{|x=x}(L) &= \operatorname*{E}_{T,y}\left[(\hat{h}_T(x) - y)^2\right] && (1.3)\\
&= \operatorname*{E}_{T,y}\left[((\hat{h}_T(x) - h^{\square}(x)) - \varepsilon)^2\right] && (1.4)\\
&= \operatorname*{E}_{T,y}\left[(\hat{h}_T(x) - h^{\square}(x))^2 + \varepsilon^2 - 2\cdot\varepsilon\cdot(\hat{h}_T(x) - h^{\square}(x))\right] && (1.5)\\
&= \operatorname*{E}_{T}\left[(\hat{h}_T(x) - h^{\square}(x))^2\right] + \operatorname*{E}_{y}\left[\varepsilon^2\right] - \operatorname*{E}_{T,y}\left[2\cdot\varepsilon\cdot(\hat{h}_T(x) - h^{\square}(x))\right] && (1.6)\\
&= \operatorname*{E}_{T}\left[(\hat{h}_T(x) - h^{\square}(x))^2\right] + \operatorname*{E}_{y}\left[\varepsilon^2\right] - 2\cdot\operatorname*{E}_{y}\left[\varepsilon\right]\cdot\operatorname*{E}_{T}\left[(\hat{h}_T(x) - h^{\square}(x))\right] && (1.7)\\
&= \operatorname*{E}_{T}\left[(\hat{h}_T(x) - h^{\square}(x))^2\right] + \operatorname*{E}_{y}\left[\varepsilon^2\right] && (1.8)
\end{aligned}$$

Výraz sme upravili, potom v kroku 1.5 roznásobili a v kroku 1.6 využili linearitu strednej hodnoty. Ďalej v kroku 1.7 sme využili, že stredná hodnota súčinu nezávislých premenných je súčinom stredných hodnôt tých premenných. Nakoniec v kroku 1.8 využívame $\mathrm{E}[\varepsilon] = 0$. Zamerajme

sa ďalej na prvý sčítanec.

$$= \underset{T}{\mathrm{E}}\left[(\hat{h}_T(x) - h^{\square}(x))^2\right] \tag{1.9}$$

$$= \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)^2 + h^{\square}(x)^2 - 2 \cdot \hat{h}_T(x) \cdot h^{\square}(x)\right] \tag{1.10}$$

$$= \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)^2\right] + \underset{T}{\mathrm{E}}\left[h^{\square}(x)^2\right] - \underset{T}{\mathrm{E}}\left[2 \cdot \hat{h}_T(x) \cdot h^{\square}(x)\right] \tag{1.11}$$

$$= \left(\underset{T}{\mathrm{Var}}\left(\hat{h}_T(x)\right) + \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right]^2\right) + \left(\underset{T}{\mathrm{Var}}\left(h^{\square}(x)\right) + \underset{T}{\mathrm{E}}\left[h^{\square}(x)\right]^2\right) - \underset{T}{\mathrm{E}}\left[2 \cdot \hat{h}_T(x) \cdot h^{\square}(x)\right] \tag{1.12}$$

$$\tag{1.13}$$

V kroku 1.10 sme výraz roznásobili a potom v kroku 1.11 využili linearitu strednej hodnoty. V poslednom kroku (1.12) sme využili vzťah $\mathrm{Var}(a) = \mathrm{E}\left[a^2\right] - \mathrm{E}\left[a\right]^2$. Pokračujme ďalej v úpravách.

$$= \left(\underset{T}{\mathrm{Var}}\left(\hat{h}_T(x)\right) + \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right]^2\right) + h^{\square}(x)^2 - 2 \cdot \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right] \cdot h^{\square}(x) \tag{1.14}$$

$$= \underset{T}{\mathrm{Var}}\left(\hat{h}_T(x)\right) + \left(\underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right]^2 + h^{\square}(x)^2 - 2 \cdot \underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right] \cdot h^{\square}(x)\right) \tag{1.15}$$

$$= \underset{T}{\mathrm{Var}}\left(\hat{h}_T(x)\right) + \left(\underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right] - h^{\square}(x)\right)^2 \tag{1.16}$$

V kroku 1.14 sme využili, že x je vopred dané, a teda $h^{\square}(x)$ je konštanta. Má teda nulový rozptyl a jeho stredná hodnota je identická s jeho hodnotou. Ďalej sme už len upravovali. Keď to celé dáme do jednej rovnice, dostaneme

$$\mathrm{Err}_{|x=x}(L) = \underbrace{\underset{T}{\mathrm{Var}}\left(\hat{h}_T(x)\right)}_{\text{rozptyl}} + \underbrace{\left(\underset{T}{\mathrm{E}}\left[\hat{h}_T(x)\right] - h^{\square}(x)\right)^2}_{\text{výchylka}^2} + \underbrace{\underset{y}{\mathrm{E}}\left[\varepsilon^2\right]}_{\text{šum}}.$$

□

Kapitola 2

PAC učenie

V tejto kapitole sa budeme zaoberať otázkou toho, ako závisí chyba algoritmu od veľkosti trénovacej množiny. Konkrétne sa budeme zaoberať otázkami ako:

- "Pri danej veľkosti trénovacej množiny t, akú chybu algoritmu môžeme očakávať?"
- "Pri danom t, s akou pravdepodobnosťou nám algoritmus vráti hypotézu, ktorej chyba je menšia ako ε?"

Na základe odpovedí na tieto dve otázky potom budeme schopní zodpovedať nasledovné, príbuzné otázky:

- "Akú veľkú trénovaciu množinu máme zvoliť, aby sme dosiahli dostatočne malú ($\leq \epsilon$) chybu algoritmu?"
- "Aké t máme zvoliť, aby sme s vysokou pravdepodobnosťou ($\geq \delta$) dostali dostatočne dobrú ($\leq \varepsilon$) hypotézu?"

Odtiaľ sa odvíja názov *PAC učenie* (z anglického *probably approximately correct learning*).

Obe typy "chýb" sú potrebné, keď sa chceme rozprávať o tom, aký vplyv má veľkosť trénovacej množiny na trénovací algoritmus. Po prvé, ε je potrebné ako miera toho, čo je dostatočne dobrá hypotéza. Po druhé, δ je potrebné, nakoľko vo všeobecnosti nevieme garantovať, že dostaneme dobrú hypotézu: mohli sme si (s malou pravdepodobnosťou) vytiahnuť zlé trénovacie dáta.

Začneme definíciou PAC učenia, ktorá ešte nebude brať do úvahy výpočtovú stránku učenia.

Zameriame sa na binárne klasifikačné úlohy, v ktorých je cieľom rozlíšiť medzi reprezentantmi nejakého cieľového konceptu od nereprezentantov. Napríklad cieľovým konceptom môže byť "písmeno A". Hypotéza dostane na vstupe obrázok 32×32 a má povedať, či tento obrázok vyobrazuje písmeno A alebo nie.

Máme teda množinu konceptov C, z ktorej pochádza cieľový koncept c. Náš trénovací algoritmus ale nevie, ktorý z nich to je. Jeho úlohou je nájsť dobrú aproximáciu, ktorú bude hľadať v množine hypotéz H. Budeme predpokladať, že $H \supseteq C$, aby bolo zaručená existencia dobrej hypotézy.

Algoritmus na vstupe dostane niekoľko trénovacích príkladov v tvare dvojíc $(x, c(x))$, pričom x pochádza z pravdepodobnostného rozdelenia P. Uvažovať y v rozdelení P nie je potrebné, nakoľko je jednoznačne určené cez x a c.

Definícia 1. Nech C je množina konceptov nad množinou vstupov X. Hovoríme, že C je *PAC naučiteľná* ak existuje algoritmus L s nasledujúcou vlastnosťou: pre každý (cieľový) koncept $c \in C$, pre každé $\varepsilon > 0$, $\delta > 0$ a pre každé možné pravdepodobnostné rozdelenie P, ak algoritmu L dáme na vstupe t trénovacích príkladov $(x_i, c(x_i))$ náhodne vybraných podľa P a t je dostatočne veľké, tak nám algoritmus s pravdepodobnosťou nanajvýš δ vráti hypotézu $\hat{h} \in C$ spĺňajúcu

$\mathrm{err}(\hat{h}) \leq \varepsilon$. Táto pravdepodobnosť sa berie cez náhodnosť v ťahaní trénovacích príkladov a prípadnú náhodnosť algoritmu L.

Poznámka 2.1. Neskôr túto definíciu rozšírime tak, aby brala do úvahy aj výpočtovú stránku algoritmu. Vtedy sa budeme zaoberať aj tým, v akom čase náš algoritmus beží a koľko trénovacích príkladov potrebuje. Budeme hovoriť o *efektívnej PAC naučiteľnosti*.

Poznámka 2.2. Pripomíname, že chyba hypotézy h sa pre klasifikačné úlohy počíta ako

$$\mathrm{err}(h) = \mathop{\mathrm{E}}_{x \sim P}[h(x) \neq c(x)] = \mathop{\Pr}_{x \sim P}(h(x) \neq c(x)).$$

Budeme hovoriť, že trénovací príklad (x, y) je *pozitívny*, ak $y = 1$. V opačnom prípade budeme hovoriť, že príklad je *negatívny*.

2.1 Konečné množiny hypotéz

V tejto časti sa zameriame na konečné množiny hypotéz.

Budeme predpokladať, že algoritmus vždy vráti *konzistentný klasifikátor*: takú hypotézu, ktorá je konzistentná s trénovacími príkladmi, teda že pre ľubovoľnú trénovaciu množinu T platí $\mathrm{err}_T(\hat{h}) = 0$. Za predpokladu $H \supseteq C$ je to splniteľný predpoklad: jedným konzistentným klasifikátorom je samotný cieľový koncept c.

2.1.1 Základné výsledky

Veta 2.1. *Nech je dané $\varepsilon > 0$. Hypotézu nazveme* zlú, *ak jej chybovosť je väčšia ako ε. Potom vieme pomocou počtu trénovacích príkladov t odhadnúť pravdepodobnosť, že nám algoritmus vráti zlú hypotézu, nasledovne:*

$$\Pr(\mathrm{err}(\hat{h}) > \varepsilon) < |H| \cdot e^{-\varepsilon t}$$

Dôkaz. Ak žiadna zlá hypotéza nie je konzistentná s príkladmi, tak výstupom algoritmu nemôže byť zlá hypotéza. Budeme sa teda snažiť zhora odhadnúť pravdepodobnosť, že aspoň jedna zlá hypotéza "prežila".

Nech h je ľubovoľná zlá hypotéza. Pravdepodobnosť, že je konzistentná s trénovacími príkladmi, je rovná $(1 - \mathrm{err}(h))^t$, čo vieme odhadnúť nasledovne:

$$(1 - \mathrm{err}(h))^t < (1 - \varepsilon)^t \leq e^{-\varepsilon t}$$

Zlých hypotéz je nanajvýš toľko, koľko je všetkých hypotéz, teda $|H|$. Pravdepodobnosť, že aspoň jedna z nich bude konzistentná s príkladmi, sa dá odhadnúť zhora ako súčet ich pravdepodobností:

$$\Pr(\text{aspoň jedna zlá}) < |H| \cdot e^{-\varepsilon t}$$

Odkiaľ dostávame požadovanú nerovnosť. □

Poznámka 2.3. Vo vyššie uvedenom dôkaze sme nepredpokladali nič o pravdepodobnostnom rozdelení P ani o cieľovom koncepte c. Uvedená veta teda platí pre ľubovoľné P a c.

Čo ak nás zaujíma druhá otázka: "Ako závisí chyba algoritmu od počtu trénovacích príkladov?" Pri klasifikačných úlohách je táto otázka úzko spätá s predošlou otázkou, kde sme sa zaujímali o ε a δ.

Veta 2.2. *Platí*

$$\text{očakávaná testovacia chyba} \leq \frac{1}{t} \cdot (\ln |H| + \ln t + 1)\,.$$

Dôkaz. Ak nám algoritmus vráti dobrú hypotézu (s chybou nanajvýš ε), vieme jej chybu odhadnúť zhora ako ε. Ak nám algoritmus vráti zlú hypotézu, jej chyba je nanajvýš 1. Z toho dostávame nasledovný horný odhad na celkovú chybu algoritmu:

$$\text{očakávaná testovacia chyba} = \mathrm{E}\left[\mathrm{err}(\hat{h})\right] \leq \Pr(\mathrm{err}(\hat{h}) \leq \varepsilon) \cdot \varepsilon + \Pr(\mathrm{err}(\hat{h}) > \varepsilon) \cdot 1$$

Pritom pravdepodobnosti na pravej strane vieme odhadnúť zhora: $\Pr(\mathrm{err}(\hat{h}) \leq \varepsilon) \leq 1$, a druhú vieme odhadnúť pomocou vety 2.1. Dostávame tak odhad

$$\text{očakávaná testovacia chyba} \leq 1 \cdot \varepsilon + |H| \cdot e^{-\varepsilon t}.$$

My sme si ale mohli zvoliť ε ľubovoľne. Ak teda chceme dostať čo najlepší odhad, nájdeme ε, pre ktoré je výraz na pravej strane čo najmenší. Zderivujme a položme rovné nule:

$$1 - |H| \cdot t \cdot e^{-\varepsilon t} = 0 \tag{2.1}$$

$$\varepsilon = \frac{1}{t} \cdot (\ln |H| + \ln t) \tag{2.2}$$

Odtiaľ dosadením dostaneme požadovaný odhad na chybu algoritmu. □

Na základe týchto dvoch viet vieme sformulovať postačujúce podmienky na t také, aby boli príslušné chyby (ε, δ a chyba algoritmu) dostatočne malé. Sformulujeme a dokážeme jednu z nich.

Dôsledok 2.3. *Množina hypotéz H je PAC-naučiteľná: pre každé $\varepsilon > 0$, $\delta > 0$, ľubovoľné pravdepodobnostné rozdelenie P a ľubovoľný cieľový koncept $f \in H$ existuje počet trénovacích príkladov t taký, že platí*

$$\Pr(\mathrm{err}(\hat{h}) \leq \varepsilon) \geq 1 - \delta.$$

Ekvivalentne,

$$\Pr(\mathrm{err}(\hat{h}) > \varepsilon) \leq \delta.$$

Dôkaz. Podľa vety 2.1 platí

$$\Pr(\mathrm{err}(\hat{h}) > \varepsilon) < |H| \cdot e^{-\varepsilon t}.$$

Stačí nám teda zvoliť také t, aby bol výraz na pravej strane menší rovný δ. Odtiaľ dostaneme postačujúci počet trénovacích príkladov t:

$$|H| \cdot e^{-\varepsilon t} \leq \delta \tag{2.3}$$

$$\ln |H| - \varepsilon t \leq \ln \delta \tag{2.4}$$

$$\varepsilon t \geq \ln |H| - \ln \delta \tag{2.5}$$

$$t \geq \frac{1}{\varepsilon} \cdot \left(\ln |H| + \ln \frac{1}{\delta}\right) \tag{2.6}$$

□

2.1.2 Problém konjunkcie

Jedným príkladom problému, kde je množina hypotéz konečná, je *problém konjunkcie pozitívnych literálov*. Na ňom si ukážeme, že (aspoň v niektorých problémoch) sú vyššie uvedené odhady relatívne tesné.

Nech je dané n. Množina vstupov sú všetky možné ohodnotenia boolovských premenných $x_1, \ldots, x_n$. Napríklad $x_1 = 0$, $x_2 = 1$, $x_3 = 0$ je priradenie hodnôt. Priradenia vieme zapísať vo vektorovom tvare: vyššie uvedený príklad by sme zapísali ako $x = (0, 1, 0)$. Všetkých vstupov je zrejme 2^n.

Množina konceptov C sú všetky konjunkcie nad pozitívnymi literálmi $x_1, \ldots, x_n$. Tieto konjunkcie sú chápané ako funkcie, ktoré vracajú 1 iba ak dané ohodnotenie premenných spĺňa túto konjunkciu. Napríklad $x_1 \wedge x_3 \wedge x_4$ vráti 1 na všetkých tých vstupoch, kde táto konjunkcia platí: musí platiť $x_1 = 1$, $x_3 = 1$, $x_4 = 1$, ale ostatné premenné už môžu mať ľubovoľnú hodnotu. Všetkých konceptov je zrejme tiež 2^n. Množina hypotéz H bude rovnaká, ako množina konceptov.

Aby sme videli, že konzistentnosť s trénovacími príkladmi je realistická požiadavka, ukážeme si, ako sa dá na základe trénovacích príkladov zostrojiť nejaký konzistentný klasifikátor.

Veta 2.4. *Nech hypotéza $\hat{h}$ je konjunkciou všetkých tých premenných, ktoré sa vyskytujú vo všetkých pozitívnych trénovacích príkladoch. Potom $\hat{h}$ je konzistentná so všetkými trénovacími príkladmi.*

Dôkaz. Ak (x, y) je pozitívny príklad, v cieľovej hypotéze c môžu byť jedine tie premenné, ktoré majú v x priradenú hodnotu 1. Keď túto úvahu zopakujeme pre všetky pozitívne príklady, dostaneme niekoľko množín "povolených premenných". Cieľová premenná musí byť podmnožinou všetkých z nich, teda je podmnožinou ich prieniku. Pritom ich prienik je práve hypotéza $\hat{h}$.

Z toho vyplýva, že ak nie je splnené c, nemôže byť splnené ani h. Naša hypotéza totiž kladie ešte väčšie požiadavky na hodnoty premenných. Teda,

$$(c(x) = 0) \implies (h(x) = 0),$$

takže h je konzistentná s negatívnymi príkladmi.

Čo sa týka pozitívnych príkladov, v h sú všetky tie premenné, ktoré sme do nej mohli dať tak, aby bola konzistentná so všetkými pozitívnymi príkladmi. Teda jej konzistentnosť s pozitívnymi príkladmi vyplýva priamo z jej konštrukcie. □

Uvedieme teraz niektoré výsledky z predchádzajúcej časti tak, ako platia pre problém konjunkcie.

Dôsledok 2.5. *Platí*

$$\text{očakávaná testovacia chyba} \leq \frac{1}{t} \cdot (n \ln 2 + \ln t + 1) = O\left(\frac{n + \ln t}{t}\right).$$

Dôsledok 2.6. *Aby sme mali zaručené (s pravdepodobnosťou aspoň $1-\delta$), že dostaneme hypotézu s chybou nanajvýš ε, stačí zvoliť veľkosť trénovacej množiny nasledovne:*

$$t \geq \frac{1}{\varepsilon} \cdot \left(n \ln 2 + \ln \frac{1}{\delta}\right) = \Omega\left(\frac{n + \ln \frac{1}{\delta}}{\varepsilon}\right)$$

Tieto odhady sú relatívne tesné. Vo všeobecnom prípade je ťažké dostať nejaký dolný odhad, nakoľko pravdepodobnostné rozdelenie P a cieľový koncept c môžu byť degenerované a "uľahčiť algoritmu robotu". Uvidíme ale, že pre niektoré "ťažké" prípady vieme spraviť dolný odhad.

Veta 2.7. *Existuje pravdepodobnostné rozdelenie P a cieľový koncept $c \in C$ také, že nech je trénovací algoritmus ľubovoľný, pre jeho chybu platí nasledovný dolný odhad:*

$$\text{očakávaná testovacia chyba} \geq \frac{1}{2e} \cdot \frac{n-1}{t+1} = \Omega\left(\frac{n}{t}\right)$$

V znení vety je trochu obmedzujúce, že cieľový koncept musí byť pevne vybraný. Ukážeme teda najprv, že pokiaľ tvrdenie dokážeme pre náhodne vybrané c, bude z neho plynúť pôvodné tvrdenie.

Lemma 2.8. *Nech P_C je pravdepodobnostné rozdelenie nad množinou konceptov C. Cieľovú hypotézu vyberieme náhodne podľa P_C. Ak pre nejaké číslo k platí, že stredná hodnota chyby algoritmu je aspoň k, tj.*

$$\mathop{\mathrm{E}}_{c\sim P_C}[\text{očakávaná testovacia chyba}] \geq k,$$

tak existuje voľba cieľového konceptu $c \in C$ taká, že chyba algoritmu bude tiež aspoň k.

Dôkaz. Ak by na každom cieľovom koncepte bola chyba algoritmu menšia ako k, potom by aj ľubovoľný vážený priemer chýb (zodpovedajúci strednej hodnote s rozdelením P_C) bol menší ako k. To by bol spor. □

Ďalej pokračujeme dôkazom vety 2.7. V ňom si už môžeme dovoliť vyberať cieľový koncept náhodne.

Dôkaz. Zvolíme pravdepodobnostné rozdelenie, ktoré priradí nenulovú pravdepodobnosť iba nasledovnej sade vstupov. Konkrétne pravdepodobnosti určíme neskôr v dôkaze tak, ako sa nám to bude hodiť.

$$x^{(1)} = (0, 1, 1, \ldots, 1, 1) \tag{2.7}$$
$$x^{(2)} = (1, 0, 1, \ldots, 1, 1) \tag{2.8}$$
$$\vdots \tag{2.9}$$
$$x^{(n)} = (1, 1, 1, \ldots, 1, 0) \tag{2.10}$$

Tieto vstupy majú nasledujúcu vlastnosť. Pre ľubovoľný koncept c platí, že sa v ňom nachádza konjunkcia x_i vtedy a len vtedy, keď $f(x^{(i)}) = 0$. Každý z týchto vstupov sa dá teda chápať ako "test na niektorú premennú".

Trénovacie príklady potom môžeme chápať tak, že algoritmu dávajú informáciu o jednotlivých premenných: "Je alebo nie je v cieľovej hypotéze?" Pokiaľ ale niektorý zo vstupov $x^{(i)}$ nie je medzi trénovacími príkladmi, nemáme žiadnu informáciu o tom, či sa tam príslušná premenná nachádza. Ak si potom $\hat{h}$ vytiahne pri testovaní takýto vstup, môže jedine hádať, aká je správna odpoveď. Šanca úspechu pri hádaní bude $\frac{1}{2}$, pokiaľ vyberieme c rovnomerne náhodne z celej množiny konceptov.

Označme si pravdepodobnosti pridelené jednotlivým vstupom $p_1, \ldots, p_n$. Aká je šanca, že pri trénovaní vstup $x^{(i)}$ nedostaneme a potom si ho pri testovaní vytiahneme?

$$p_i \cdot (1 - p_i)^t$$

Celková pravdepodobnosť, že si vytiahneme pri testovaní vstup mimo trénovacej množiny, je potom súčet jednotlivých pravdepodobností (nakoľko sú jednotlivé udalosti dizjunktné):

$$\sum_{i=1}^{n} p_i \cdot (1 - p_i)^t$$

V týchto prípadoch bude mať $\hat{h}$ chybu $\frac{1}{2}$. V ostatných prípadoch sme si vytiahli počas testovania nejaký príklad, ktorý bol aj v trénovacej množine. Pokiaľ sme si ho zapamätali, tak budeme mať chybu 0, v každom prípade bude ale chyba aspoň 0. Dostávame tak nasledovný dolný odhad na chybu algoritmu:

$$\mathop{\mathrm{E}}_{c\in C}[\text{očakávaná testovacia chyba}] \geq \frac{1}{2} \cdot \left(\sum_{i=1}^{n} p_i \cdot (1 - p_i)^t \right)$$

Ako ale zvoliť $p_1, \ldots, p_n$ tak, aby sme dostali dobrý dolný odhad? Skúsime zvoliť p_i také, pre ktoré nadobúda výraz $p_i \cdot (1-p_i)^t$ maximum. Zderivovaním a položením rovné 0 dostaneme

$$p_i = \frac{1}{t+1}.$$

Pokiaľ ale $t \neq n-1$, nemôžeme zvoliť všetky pravdepodobnosti takéto: pre $t < n-1$ je súčet pravdepodobností priveľký a pre $t > n-1$ primalý. Prvý prípad nás nezaujíma, nakoľko je to "len konštanta" (v zmysle $t \to \infty$). V druhom prípade si vieme zvoliť jedného "obetného baránka" p_1, ktorému priradíme celú zvyšnú pravdepodobnosť.

$$p_1 = 1 - \frac{n-1}{t+1} \tag{2.11}$$

$$p_2 = \frac{1}{t+1} \tag{2.12}$$

$$\vdots \tag{2.13}$$

$$p_n = \frac{1}{t+1} \tag{2.14}$$

Dosadíme a dostaneme tak odhad:

$$\mathop{\mathrm{E}}_{c \in C}[\text{očakávaná testovacia chyba}] \geq \frac{1}{2} \cdot \left((n-1) \cdot \frac{1}{t+1} \cdot \left(1 - \frac{1}{t+1}\right)^t + \frac{n-1}{t+1} \cdot \left(1 - \frac{n-1}{t+1}\right)^t \right)$$

Odignorujeme druhý sčítanec, čím sa nám výraz na pravej strane môže len zmenšiť, takže nerovnosť sa zachová. Ďalej použijeme odhad

$$\left(1 - \frac{1}{t+1}\right)^t \geq \frac{1}{e},$$

ktorý platí pre všetky prirodzené t. (Tento odhad sa dá dokázať napríklad tak, že sa dokáže, že daný výraz je rastúci od t. Odhad z toho už plynie ľahko, nakoľko jeho limita pre $t \to \infty$ je práve $\frac{1}{e}$.) Dostaneme tak požadovanú nerovnosť:

$$\mathop{\mathrm{E}}_{c \in C}[\text{očakávaná testovacia chyba}] \geq \frac{1}{2e} \cdot \frac{n-1}{t+1}$$

□

Dolný odhad zodpovedajúci dôsledku 2.6 uvedieme bez dôkazu.

Veta 2.9. *Pre ľubovoľný trénovací algoritmus existuje pravdepodobnostné rozdelenie P a cieľová hypotéza $f \in H$, ktoré vynútia, že algoritmus bude potrebovať aspoň*

$$t = \Omega\left(\frac{1}{\varepsilon} \cdot \left(n + \ln\frac{1}{\delta}\right)\right),$$

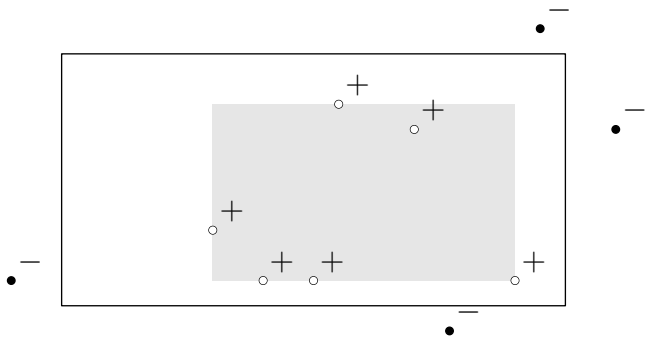
aby platilo

$$\Pr(\mathrm{err}(\hat{h}) > \varepsilon) \leq \delta.$$

TODO dôkaz

2.2 Nekonečné množiny hypotéz

Vyššie uvedený postupy pre konečné množiny hypotéz zlyhajú, ak $|H| = \infty$: nedostaneme z nich žiaden odhad. Aj pre nekonečné množiny hypotéz je ale možné odvodiť horné odhady. Tie sa už nebudú odvíjať od veľkosti množiny H, ale od nejakej jej miery zložitosti. Touto mierou zložitosti bude tzv. *Vapnik-Chervonenkisova dimenzia*. Predtým sa ale pozrieme na jednoduchý príklad.



Obr. 2.1: Obdĺžniková hra: najmenší obdĺžnik konzistentný s príkladmi.

2.2.1 Obdĺžniková hra

Ide o klasifikačnú úlohu. Množina vstupov sú všetky body v rovine. Množina konceptov sú všetky obdĺžniky, ktorých strany sú rovnobežné so súradnicovými osami. Pre každý bod v rovine sa teda pýtame, či je vo vnútri cieľového obdĺžnika c alebo nie. Body na okraji obdĺžnika považujeme, že sú vnútri.

Pre jednoduchosť argumentu použijeme konkrétny trénovací algoritmus. Dôkaz by sa dal zovšeobecniť na prípad, kedy jediné, čo o algoritme predpokladáme je, že nám vracia konzistentný klasifikátor. S nasledovným konzistentným klasifikátorom to ale bude jednoduchšie.

Z trénovacích príkladov vezmime tie pozitívne. Ako hypotézu $\hat{h}$ zvolíme najmenší (vzhľadom na inklúziu) osovorovnobežný obdĺžnik, ktorý obsahuje všetky body vo vybraných príkladoch. Na obrázku 2.1 ilustrujeme konštrukciu $\hat{h}$ a porovnávame ho s c.

Lemma 2.10. *Každá množina bodov $(x_1, y_1), \ldots, (x_t, y_t)$ má unikátny* bounding box*: najmenší (vzhľadom na inklúziu) osovorovnobežný obdĺžnik, ktorý ich všetky obsahuje.*

Dôkaz. Každý obdĺžnik je jednoznačne určený x-ovými súradnicami ľavého a pravého okraja a y-ovými súradnicami dolného a horného okraja. Označme ich $x_\leftarrow, x_\rightarrow$ a $y_\downarrow, y_\uparrow$. Vo vnútri sú práve tie body (x, y), ktoré spĺňajú

$$x_\leftarrow \leq x \leq x_\rightarrow,\ y_\downarrow \leq y \leq y_\uparrow.$$

Ak teda majú byť všetky spomínané body vo vnútri, musí platiť

$$x_\leftarrow \leq \min x_i,\ x_\rightarrow \geq \max x_i,\ y_\downarrow \leq \min y_i,\ y_\uparrow \geq \max y_i$$

Najmenší osovorovnobežný obdĺžnik, ktorý toto spĺňa, je ten, pre ktorý nastávajú v jednotlivých nerovnostiach rovnosti. Teda $x_\leftarrow = \min x_i$, $x_\rightarrow = \max x_i$, $y_\downarrow = \min y_i$ a $y_\uparrow = \max y_i$. □

Dôsledok 2.11. *Obdĺžnik $\hat{h}$ je podmnožinou cieľového obdĺžnika c.*

Dôsledok 2.12. *Obdĺžnik $\hat{h}$ je konzistentný klasifikátor.*

Dôkaz. Konzistentnosť na pozitívnych príkladoch vyplýva z konštrukcie $\hat{h}$. Každý negatívny príklad je mimo c a teda aj mimo $\hat{h}$, je teda konzistentný aj s negatívnymi príkladmi. □

Veta 2.13. *Obdĺžniková hra je PAC naučiteľná: pre každý cieľový obdĺžnik $c \in C$, $\varepsilon > 0$, $\delta > 0$ a pravdepodobnostné rozdelenie P, ak vyššie popísanému algoritmu dáme t trénovacích príkladov z P a platí*

$$t \geq \frac{4}{\varepsilon} \cdot \ln \frac{4}{\delta},$$

tak nám algoritmus s vysokou pravdepodobnosťou vráti hypotézu s nízkou chybou:

$$\Pr(\mathrm{err}(\hat{h}) > \varepsilon) \leq \delta.$$

Dôkaz. Pod *váhou* množiny budeme rozumieť pravdepodobnosť, že náhodný bod z rozdelenia P padne do tejto množiny.

Zle klasifikované body sú práve tie, ktoré sú vo vnútri c ale nie sú vo vnútri $\hat{h}$. Každý z týchto bodov padne do aspoň jedného z okrajových "pásikov" (nerátajúc jeden z okrajov), ako je zobrazené na obrázku 2.2a. Označme $p_{\leftarrow}, p_{\rightarrow}, p_{\uparrow}, p_{\downarrow}$ postupne váhy ľavého, pravého, horného, resp. dolného pásika, chyba $\hat{h}$ sa potom dá zhora odhadnúť ako súčet týchto váh:

$$\mathrm{err}(\hat{h}) \leq p_{\leftarrow} + p_{\rightarrow} + p_{\uparrow} + p_{\downarrow}.$$

Ak by sme voľbou dostatočne veľkého t vedeli zaručiť (s pravdepodobnosťou aspoň $1-\delta$), že dostaneme takú hypotézu $\hat{h}$, pre ktorú je výraz na pravej strane nanajvýš ε, vyhrali by sme.

Ukážeme, že vieme zaručiť (s pravdepodobnosťou aspoň $1-\frac{\delta}{4}$) každú zo štyroch nerovností:

$$p_{\leftarrow}, p_{\rightarrow}, p_{\uparrow}, p_{\downarrow} \leq \frac{\varepsilon}{4}.$$

Potom pravdepodobnosť, že budú všetky štyri nerovnosti platiť súčasne, bude aspoň $1-\delta$, čo je presne to, čo chceme. Postup bude vo všetkých štyroch prípadoch ten istý, ukážeme si to teda len na ľavom pásiku.

Kedy je váha ľavého pásika malá? Keď aspoň jeden trénovací príklad je dostatočne blízko k ľavému okraju. Keď sa na to pozrieme opačne, váha pásika je veľká, ak žiaden trénovací nie je blízko. Nech α je pravdepodobnosť toho, že trénovací príklad je blízko ľavého okraju. Pravdepodobnosť, že ani jeden z trénovacích príkladov nie je blízko, je potom $(1-\alpha)^t \leq e^{-\alpha t}$. My chceme zvoliť také t, aby pravdepodobnosť tohto zlého prípadu bola nanajvýš $\frac{\delta}{4}$:

$$e^{-\alpha t} \leq \frac{\delta}{4} \tag{2.15}$$

$$-\alpha t \leq \ln \frac{\delta}{4} \tag{2.16}$$

$$t \geq \frac{1}{\alpha} \cdot \ln \frac{4}{\delta} \tag{2.17}$$

Stačí teda zvoliť $t \geq \frac{1}{\alpha} \cdot \ln \frac{4}{\delta}$. Pre úplnosť ešte ukážeme dolný odhad na pravdepodobnosť α.

Čo presne znamená byť "dostatočne blízko" k ľavému okraju? Bod (x, y) je blízko, ak váha všetkých bodov naľavo od neho je nanajvýš $\frac{\varepsilon}{4}$. Inak zapísané,

$$\Pr_{(x',y')\sim P}(x' < x) \leq \frac{\varepsilon}{4}.$$

Vyznačme si teda všetky takéto body. Množina týchto bodov bude tvoriť súvislý úsek od ľavého okraja (ako na obrázku 2.2b), s tým, že hranica úseku v ňej môže ale nemusí byť. Ukážeme, že váha týchto bodov je aspoň $\frac{\varepsilon}{4}$, čím dostaneme odhad $\alpha \geq \frac{\varepsilon}{4}$. (Nie je ťažké vidieť, že ak je rozdelenie P spojité, tak je táto váha dokonca presne $\frac{\varepsilon}{4}$.)

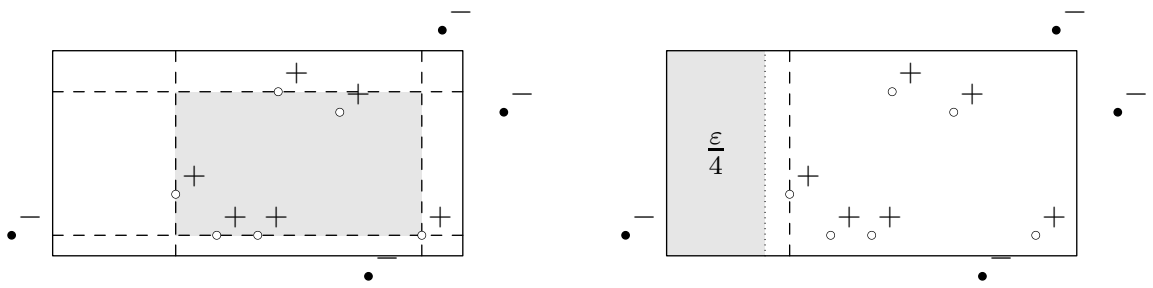
TODO dôkaz (eww, grc)

□

2.2.2 Vapnik-Chervonenkisova dimenzia

Vapnik-Chervonenkisova dimenzia (alebo skrátene VC dimenzia) je miera zložitosti množiny hypotéz, pomocou ktorej sme schopní tvoriť tvrdenia v štýle PAC učenia. Na jej definíciu budeme potrebovať niekoľko pomocných pojmov.

Definícia 2. Majme konečnú podmnožinu vstupov $S \subseteq X$. Hovoríme, že množina hypotéz H *rozbíja* množinu S, ak platí: nech označíme vstupy v S ako pozitívne alebo negatívne akokoľvek, v množine H existuje hypotéza konzistentná s týmto označením.



(a) Krajné pásiky.

(b) Sivou farbou je vyznačená množina všetkých bodov, pre ktoré má pásik naľavo od nich váhu najvýš $\frac{\varepsilon}{4}$.

Obr. 2.2: Obdĺžniková hra.

Obr. 2.3: Žiaden interval nevytvorí takéto označenie bodov.

Intuitívne, množina S je rozbitá hypotézami H, ak nie je pre hypotézy v H príliš náročné modelovať príklady v S: sú schopné ich modelovať akokoľvek.

Definícia 3. Vapnik-Chervonenkisova dimenzia množiny hypotéz H, označovaná $VCD(H)$, je veľkosť najväčšej množiny $S \subseteq X$, ktorá sa dá rozbiť hypotézami v H. Ak sa dá rozbiť ľubovoľne veľká množina S, definujeme $VCD(H) = \infty$.

Teda na to, aby sme ukázali dolný odhad $VCD(H) \geq d$, stačí nám nájsť jednu množinu S veľkosti d, ktorá sa dá rozbiť. Aby sme ale ukázali horný odhad $VCD(H) \leq d$, musíme ukázať, že žiadna množina veľkosti $d + 1$ sa nedá rozbiť: teda že existuje také označenie vstupov, pre ktoré neexistuje v H konzistentný klasifikátor. Z tohto dôvodu je obvykle ťažšie dokázať horný odhad na VC dimenziu, ako dolný odhad.

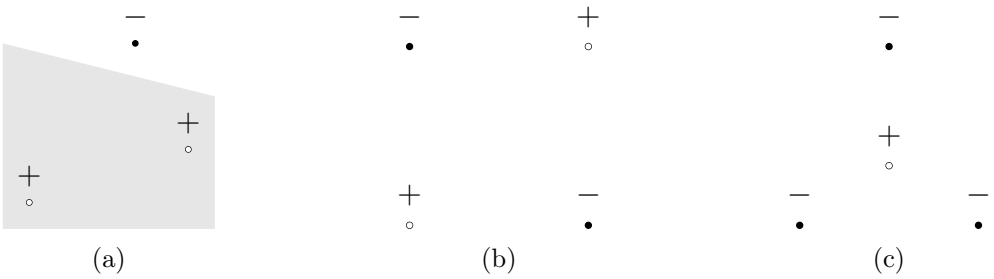
Lemma 2.14. *Majme dve množiny hypotéz H a H' nad tou istou množinou vstupov x. Ak platí $H \supseteq H'$, tak potom platí $VCD(H) \succeq VCD(H')$ (kde $\succeq$ je relácia $\geq$ prirodzene rozšírená na $\mathbb{N} \cup \{\infty\}$).*

Dôkaz. Ak sa množina S dá rozbiť pomocou H', potom, pretože každá hypotéza v H' je aj v H, dá sa táto množina rozbiť aj pomocou H (rovnakým spôsobom). □

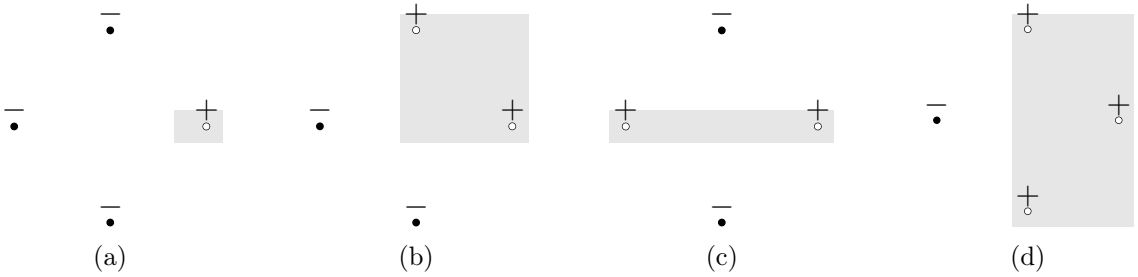
Než uvedieme vety hovoriace o PAC naučiteľnosti, uvedieme príklady niektorých nekonečných množín hypotéz, a neformálne zdôvodníme ich VC dimenzie. Väčšinou budú geometrického charakteru (čo je prirodzené, ak máme mať nekonečnú množinu hypotéz).

Intervaly na reálnych číslach. Nie je ťažké vidieť, že ľubovoľná dvojica bodov sa dá rozbiť. Na druhej strane, ak máme tri body, tak ich vieme označiť ako na obrázku 2.3, čo nie je konzistentné so žiadnym intervalom. Takže VC dimenzia je 2.

Polroviny v rovine. Každá trojica bodov tvoriacich nedegenerovaný trojuholník sa dá rozbiť, ako je ilustrované na obrázku 2.4. Na druhej strane, žiadna štvorica bodov sa rozbiť nedá: ak tvoria konvexný štvoruholník, tak jednu protiľahlú dvojicu označíme kladne a druhú záporne. Ak tvoria nekonvexný štvoruholník, jeden z bodov je vo vnútri trojuholníka tvoreného ostatnými tromi: tento bod označíme kladne a ostatné body záporne. Takže VC dimenzia je 3.



Obr. 2.4: Polroviny v rovine: v situácii (a) vieme nájsť deliacu priamku, v (b) a (c) nie.



Obr. 2.5: Osovorovnobežné obdĺžniky: štvorica bodov, ktorá sa dá rozbiť. Jednotlivé obrázky zobrazujú všetky netriviálne označenia bodov a príslušné konzistentné klasifikátory.

Osovorovnobežné obdĺžniky. Štvorica bodov, ktorá sa dá rozbiť, je ilustrovaná na obrázku 2.5, nie každá štvorica bodov sa ale dá rozbiť. Na druhej strane, žiadna pätica bodov sa nedá rozbiť. Rozoberieme dva prípady: ak je aspoň jeden z bodov vo vnútri bounding boxu (nie na okraji), tak ho označíme záporne a všetky body na okraji kladne. Ak sú všetky body na obvode bounding boxu, na jednej strane toho obdĺžnika musia ležať aspoň dva body. Jeden z nich označíme kladne a druhý záporne. Takže VC dimenzia je 4.

Nakoniec hlavný výsledok, kvôli ktorému je VC dimenzia zaujímavá.

Veta 2.15 (Blumer et. al., 1989 [1])**.** *Nech C je ľubovoľná množina konceptov. Nech H je množina hypotéz s VC dimenziou rovnou d. Nech L je ľubovoľný algoritmus, ktorý pre ľubovoľnú sadu t trénovacích príkladov vráti hypotézu $\hat{h}$ konzistentnú s príkladmi. Potom L je PAC trénovací algoritmus pre množinu konceptov C s použitím hypotéz v H, pokiaľ je t dostatočne veľké:*

$$t \geq c_0 \left(\frac{1}{\varepsilon} \ln \frac{1}{\delta} + \frac{d}{\varepsilon} \ln \frac{1}{\varepsilon} \right)$$

pre nejakú konštantu $c_0 > 0$.

TODO dôkaz

Veta 2.16 (Haussler et. al., 1994 [3])**.** *Ak VC dimenzia je d, tak ľubovoľný trénovací algoritmus, ktorý vždy vracia hypotézy konzistentné s trénovacími príkladmi, má chybu nanajvýš*

$$\mathrm{E}\left[\mathrm{err}(\hat{h})\right] = O\left(\frac{d}{t} \ln \frac{t}{d}\right).$$

TODO dôkaz

Čo ale v prípade, že neexistuje konzistentný klasifikátor? Taká situácia nastane, keď buď cieľový koncept nie je v množine hypotéz, alebo keď v probléme vystupuje šum. Ukazuje sa, že aj v takom prípade sa dá odhadnúť chyba hypotézy.

Veta 2.17 (Vapnik & Chervonenkis, 1971 [2])**.** *Nech $h^\star$ je hypotéza v H s najmenšou (testovacou) chybou a nech $\hat{h}$ je hypotéza v H s najmenšou trénovacou chybou. Ak $VCD(H) = d$, tak pre počet trénovacích príkladov*

$$t \geq c_0 \left(\frac{1}{\varepsilon} \ln \frac{1}{\delta} + \frac{d}{\varepsilon} \ln \frac{1}{\varepsilon} \right),$$

kde c_0 je nejaká konštanta, platí, že s veľkou pravdepodobnosťou je chyba hypotézy $\hat{h}$ dostatočne malá (berúc v úvahu najmenšiu dosiahnuteľnú chybu):

$$\Pr(\mathrm{err}(\hat{h}) > \mathrm{err}(h^\star) + \varepsilon) \leq \delta.$$

TODO dôkaz

Literatúra

[1] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[2] AIA Chervonenkis and VN Vapnik. Theory of uniform convergence of frequencies of events to their probabilities and problems of search for an optimal solution from empirical data(average risk minimization based on empirical data, showing relationship of problem to uniform convergence of averages toward expectation value). *Automation and Remote Control*, 32:207–217, 1971.

[3] David Haussler, Nick Littlestone, and Manfred K Warmuth. Predicting {0, 1}-functions on randomly drawn points. *Information and Computation*, 115(2):248–292, 1994.