# 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 $32 \times 32$ 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 $L$, 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 $H$.

### 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:

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

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

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

*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 $\mathrm{Err}_T$, a vypočítame ju nasledovne:

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

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 funkcií

Uvedieme ešte príklady používaných chybových funkcií. 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

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

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.

### 1.2.5 Diskusia o výpočtových detailoch

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

Namiesto konkrétneho trénovacieho algoritmu teda budeme predpokladať, že máme *všemohúci* algoritmus: taký, ktorý nám vždy vráti tú hypotézu z $H$, ktorá je z ERM hľadiska najlepšia (teda má najmenšiu chybu na trénovacej množine). Teda

$$\hat{h}_T = \mathop{\arg\min}_{h\in H} \left(\mathrm{Err}_T(h)\right).$$

Vo všeobecnosti to nie je realistický predpoklad; nie je jednoduché navrhnúť efektívny algoritmus, ktorý vždy nájde hypotézu s najmenšou chybou. Dokonca pre niektoré dostatočne zložité množiny hypotéz $H$ taký efektívny algoritmus ani nemusí existovať. Nájsť *nejaký* algoritmus ale väčšinou nie je problém, len potom beží v čase $O(t!^{t!})$ (len pre ilustráciu, nič faktické za týmto výrazom nehľadajte). Ak teda chceme holistický pohľad na učenie, nemôžeme len tak zanedbať časovú zložitosť a iné výpočtové záležitosti.

## 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).

Pripomeňme si, že náš algoritmus je všemohúci, vracia teda tú hypotézu z $H$, ktorá má najmenšiu chybu na trénovacej množine:

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

Označme $h^\star$ objektívne najlepšiu hypotézu z $H$, teda takú, ktorá má najmenšiu testovaciu chybu.

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

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

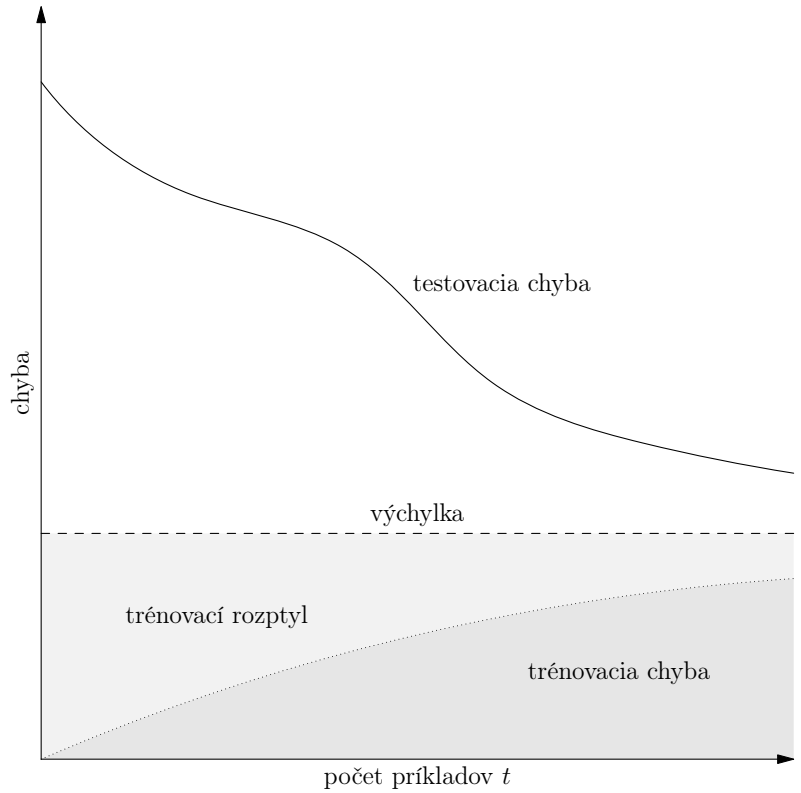
$$\mathrm{Err}_T(\hat{h}_T) \leq \mathrm{Err}_T(h^\star)$$
$$\mathrm{Err}(h^\star) \leq \mathrm{Err}(\hat{h}_T)$$

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

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



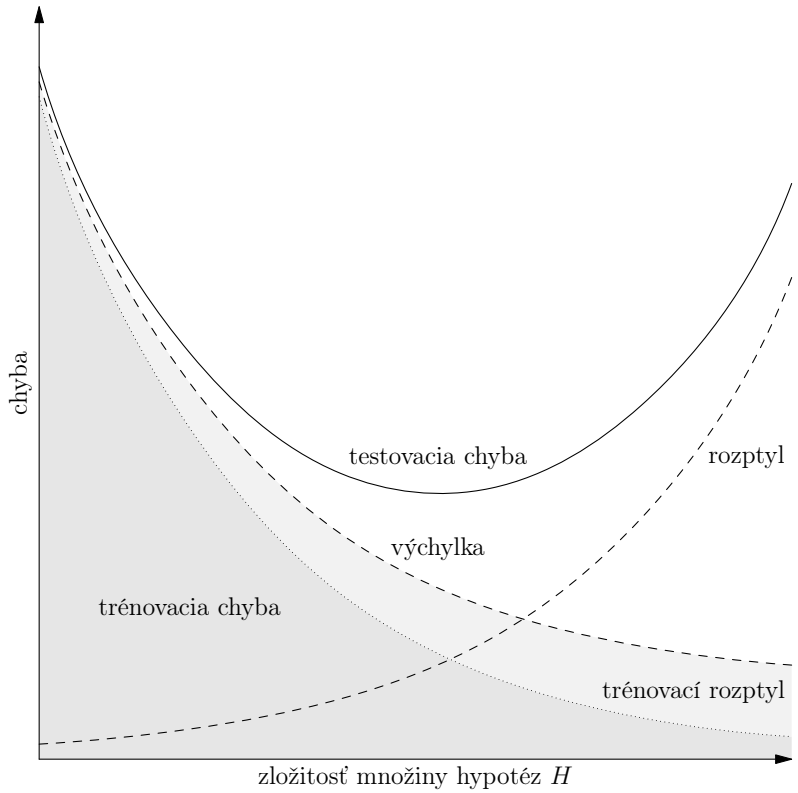

Obr. 1.1: Závislosť chyby algoritmu od počtu trénovacích príkladov $t$.

OTeCh nášho algoritmu sa od najlepšej možnej chyby líši o hodnotu $\epsilon_{\text{est.}} = \text{Err}(L) - \epsilon_{\text{approx.}}$, 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 $\epsilon^T_{\text{est.}} = \text{Err}(h^\star) - \text{Err}_T(L)$, 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 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).




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

### 1.3.3 Kompromis medzi výchylkou a rozptylom (bias-variance tradeoff)

V tomto odseku si ukážeme zaujímavý výsledok, ktorý nám za určitých predpokladov umožňuje vyjadriť chyby pomocou iných, jasnejších veličín: tzv. *výchylky* a *rozptylu*. Označme najlepšiu hypotézu z množiny $H$ ako $h^\star$, teda

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

Budeme upravovať výraz reprezentujúci OTeChAlg.

$$\mathrm{Err}(L) = \mathop{\mathrm{E}}_{T}\left[\mathrm{Err}(\hat{h}_T)\right] \tag{1.3}$$

$$= \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - y)^2\right]\right] \tag{1.4}$$

$$= \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[\Big((\hat{h}_T(x) - h^\star(x)) + (h^\star(x) - y)\Big)^2\right]\right] \tag{1.5}$$

Posledná rovnosť vychádza z netriviálneho technického kroku, ktorý si vyžaduje dodatočné predpoklady. Tieto technické detaily však prenecháme na koniec kapitoly.

$$\mathrm{Err}(L) = \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x))^2\right]\right] + \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(h^\star(x) - y)^2\right]\right]$$

Druhý zo sčítancov sa dá ešte zjednodušiť. Kedže $h^\star$ ani $y$ nezávisia od trénovacích dát, môžeme sa zbaviť vonkajšej strednej hodnoty. Dostávame tak výslednú rovnosť

$$\mathrm{Err}(L) = \underbrace{\mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x))^2\right]\right]}_{\text{rozptyl}} + \underbrace{\mathop{\mathrm{E}}_{x,y}\left[(h^\star(x) - y)^2\right]}_{\text{výchylka}}.$$

Prvý zo sčítancov budeme volať *rozptyl*. Trénovací algoritmus s malým rozptylom vracia funkcie, ktoré sú blízko optima v množine $H$. Tým, že mu zväčšíme množinu trénovacích dát, si veľmi neprilepšíme. Naopak, algoritmus s veľkým rozptylom vracia funkcie ďaleko od optima, vieme sa teda k optimu priblížiť tým, že zväčšíme množstvo trénovacích dát.

Druhý zo sčítancov budeme volať *výchylka*. Vyjadruje chybu, ktorá je spôsobená tým, že sa náš algoritmus obmedzil na nejakú konkrétnu množinu hypotéz $H$. Čím väčšia množina hypotéz, tým menšia výchylka (nakoľko $h^\star$ 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 zložitejšia množina hypotéz, tým väčší rozptyl.

Výchylku vieme upraviť ďalej. Hypotéza $h^\star$ ani $y$ nezávisia od trénovacej množiny $T$. Z ich pohľadu sú teda testovacie dáta $x, y$ a trénovacie dáta $x_i, y_i$ nerozlíšiteľné. Takže na meranie chyby $h^\star$ môžeme použiť trénovacie dáta, berúc v úvahu ich náhodný výber:

$$\text{výchylka} = \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x_i,y_i}\left[(h^\star(x_i) - y_i)^2\right]\right] \tag{1.6}$$

$$= \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x_i,y_i}\left[\Big((h^\star(x_i) - \hat{h}_T(x_i)) + (\hat{h}_T(x_i) - y_i)\Big)^2\right]\right] \tag{1.7}$$

Použitím ďalšieho technického kroku dostaneme:

$$\text{výchylka} = \underbrace{\mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x_i,y_i}\left[(h^\star(x_i) - \hat{h}_T(x_i))^2\right]\right]}_{\text{trénovací rozptyl}} + \underbrace{\mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x_i,y_i}\left[(\hat{h}_T(x_i) - y_i)^2\right]\right]}_{\text{očakávaná trénovacia chyba algoritmu}} \tag{1.8}$$

Prvý zo sčítancov budeme volať *trénovací rozptyl*. Uvedomme si, že pre ľubovoľné trénovacie dáta $T$ platí

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

nakoľko $\hat{h}_T$ je optimálna hypotéza pre danú množinu trénovacích dát. Hypotéza $h^\star$ síce je najlepšia pre $H$, trénovacie dáta sú ale len malá vzorka z $H$. Trénovací rozptyl teda môžeme chápať ako mieru toho, ako veľmi reprezentatívnu vzorku trénovacích dát sme dostali. Čím menší je, tým viac reprezentatívna je naša vzorka.

Druhý zo sčítancov je OTrChAlg. Je to stredná hodnota chyby, ktorej sa dopustí výstup z algoritmu $\hat{h}_T$ na tých istých dátach, pomocou ktorých sme $\hat{h}_T$ zostrojili.

Platí

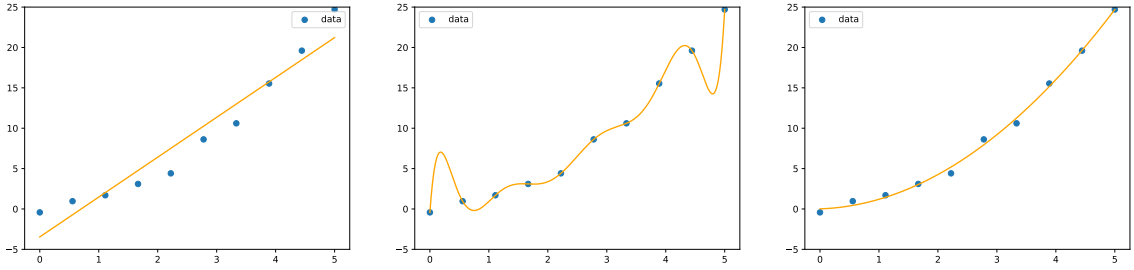
$$\mathrm{Err}_T(L) \leq \text{výchylka} \leq \mathrm{Err}(L).$$

Na konkrétnych trénovacích dátach ale nemusí platiť, že trénovacia chyba je menšia ako testovacia chyba: mohli sme si (síce s malou pravdepodobnosťou, ale predsa) vytiahnuť zlé trénovacie dáta, na ktorých sa žiadnej hypotéze z $H$ nedarí.

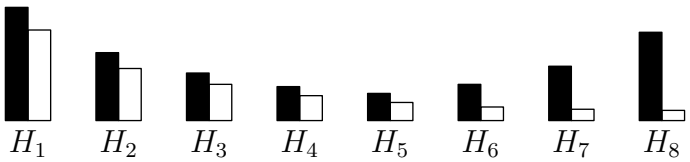
Na základe dosiaľ uvedeného vieme graficky znázorniť, ako sa zhruba správajú OTeChAlg, rozptyl, výchylka, trénovací rozptyl a OTrChAlg, v závislosti od veľkosti trénovacej množiny (obrázok 1.1) a od zložitosti množiny hypotéz (obrázok 1.2).

## 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*).




Obr. 1.3: Podučenie, preučenie, akurát.




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

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 $\varepsilon$ 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 presne 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.

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

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

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 $\lambda$ hovoriacim, ako veľké pokuty chceme udeľovať. Veľké $\lambda$ (v porovnaní s trénovacou chybou) nám hovorí, že sa snažíme hlavne dosiahnuť jednoduché hypotézy; s malým $\lambda$ 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

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

**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 $\theta$, 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 $\theta$ 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:*

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

*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.

$$\begin{aligned}
\mathrm{Err}_{|x=x}(L) &= \mathop{\mathrm{E}}_{T,y}\left[(\hat{h}_T(x)-y)^2\right] && (1.9)\\
&= \mathop{\mathrm{E}}_{T,y}\left[((\hat{h}_T(x)-h^\square(x))-\varepsilon)^2\right] && (1.10)\\
&= \mathop{\mathrm{E}}_{T,y}\left[(\hat{h}_T(x)-h^\square(x))^2+\varepsilon^2-2\cdot\varepsilon\cdot(\hat{h}_T(x)-h^\square(x))\right] && (1.11)\\
&= \mathop{\mathrm{E}}_{T}\left[(\hat{h}_T(x)-h^\square(x))^2\right]+\mathop{\mathrm{E}}_{y}\left[\varepsilon^2\right]-\mathop{\mathrm{E}}_{T,y}\left[2\cdot\varepsilon\cdot(\hat{h}_T(x)-h^\square(x))\right] && (1.12)\\
&= \mathop{\mathrm{E}}_{T}\left[(\hat{h}_T(x)-h^\square(x))^2\right]+\mathop{\mathrm{E}}_{y}\left[\varepsilon^2\right]-2\cdot\mathop{\mathrm{E}}_{y}\left[\varepsilon\right]\cdot\mathop{\mathrm{E}}_{T}\left[(\hat{h}_T(x)-h^\square(x))\right] && (1.13)\\
&= \mathop{\mathrm{E}}_{T}\left[(\hat{h}_T(x)-h^\square(x))^2\right]+\mathop{\mathrm{E}}_{y}\left[\varepsilon^2\right] && (1.14)
\end{aligned}$$

Výraz sme upravili, potom v kroku 1.11 roznásobili a v kroku 1.12 využili linearitu strednej hodnoty. Ďalej v kroku 1.13 sme využili, že stredná hodnota súčinu nezávislých premenných je súčinom stredných hodnôt tých premenných. Nakoniec v kroku 1.14 využívame $\mathrm{E}[\varepsilon]=0$. Zamerajme sa ďalej na prvý sčítanec.

$$\begin{aligned}
&= \mathop{\mathrm{E}}_{T}\left[(\hat{h}_T(x)-h^\square(x))^2\right] && (1.15)\\
&= \mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)^2+h^\square(x)^2-2\cdot\hat{h}_T(x)\cdot h^\square(x)\right] && (1.16)\\
&= \mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)^2\right]+\mathop{\mathrm{E}}_{T}\left[h^\square(x)^2\right]-\mathop{\mathrm{E}}_{T}\left[2\cdot\hat{h}_T(x)\cdot h^\square(x)\right] && (1.17)\\
&= \left(\mathop{\mathrm{Var}}_{T}\left(\hat{h}_T(x)\right)+\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]^2\right)+\left(\mathop{\mathrm{Var}}_{T}\left(h^\square(x)\right)+\mathop{\mathrm{E}}_{T}\left[h^\square(x)\right]^2\right)-\mathop{\mathrm{E}}_{T}\left[2\cdot\hat{h}_T(x)\cdot h^\square(x)\right] && (1.18)\\
& && (1.19)
\end{aligned}$$

V kroku 1.16 sme výraz roznásobili a potom v kroku 1.17 využili linearitu strednej hodnoty. V poslednom kroku (1.18) sme využili vzťah $\mathrm{Var}(a)=\mathrm{E}\left[a^2\right]-\mathrm{E}\left[a\right]^2$. Pokračujme ďalej v úpravách.

$$\begin{aligned}
&= \left(\mathop{\mathrm{Var}}_{T}\left(\hat{h}_T(x)\right)+\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]^2\right)+h^\square(x)^2-2\cdot\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]\cdot h^\square(x) && (1.20)\\
&= \mathop{\mathrm{Var}}_{T}\left(\hat{h}_T(x)\right)+\left(\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]^2+h^\square(x)^2-2\cdot\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]\cdot h^\square(x)\right) && (1.21)\\
&= \mathop{\mathrm{Var}}_{T}\left(\hat{h}_T(x)\right)+\left(\mathop{\mathrm{E}}_{T}\left[\hat{h}_T(x)\right]-h^\square(x)\right)^2 && (1.22)
\end{aligned}$$

V kroku 1.20 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

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

□

## 1.7 Appendix: technické detaily k bias-variance tradeoff

V tejto časti dokážeme tvrdenie, ktoré sme použili pri odvodení vzťahu (1.3.3).

Pripomeňme si, že $h^{\square}(x)$ je cieľová funkcia, ktorá je definovaná ako $h^{\square}(x) = E_{y|x}[y]$ a táto cieľová funkcia nemusí byť súčasťou triedy hypotéz $H$. Funkcia $h^{\star}(x)$ je taká funkcia z $H$, ktorá má najmenšiu možnú očakávanú chybu, t.j. $h^{\star} = \arg\min_{h\in H}\left(E_{x,y}[(h(x)-y)^2]\right)$. Túto definíciu vieme vztiahnuť ku funkcii $h^{\square}$ pomocou nasledujúcej lemy.

**Lemma 1.2.** *Pre funkciu $h^{\star}(x)$ platí:*

$$h^{\star} = \operatorname*{arg\,min}_{h\in H}\left(E_x[(h(x)-h^{\square}(x))^2]\right).$$

*Dôkaz.* Úpravou výrazu z definície $h^*$ získame:

$$\begin{aligned}
E_{x,y}[(h(x)-y)^2] &= E_{x,y}[((h(x)-h^{\square}(x))+(h^{\square}(x)-y))^2] \\
&= E_{x,y}[\underbrace{(h(x)-h^{\square}(x))^2}_{\text{nezávisí od } y}] + E_{x,y}[(h^{\square}(x)-y)^2] \\
&\quad +2E_{x,y}[((h(x)-h^{\square}(x)(h^{\square}(x)-y))] \\
&= E_x[(h(x)-h^{\square}(x))^2] + E_{x,y}[(h^{\square}(x)-y)^2] \\
&\quad +2E_x[(h(x)-h^{\square}(x))(h^{\square}(x)-E_{y|x}[y])]
\end{aligned}$$

Druhý sčítanec je konštanta vzhľadom na $h$, preto ho pri hľadaní minima nemusíme uvažovať. Z definície $h^{\square}(x) = E_{y|x}[y]$, preto tretí sčítanec bude nulový. Zostáva nám teda:

$$h^{\star} = \operatorname*{arg\,min}_{h\in H}\left(E_x[(h(x)-h^{\square}(x))^2]\right),$$

čo bolo treba dokázať. □

Z predchádzajúcej lemy teda vyplýva, že funkcia $h^{\star}$ je priemetom funkcie $h^{\square}$ do $H$. Pre zjednodušenie si v tejto kapitole zavedieme označenie $\langle f,g\rangle = \mathrm{E}_x[f(x)g(x)]$. Takisto si zavedieme pojem *vzdialenosti funkcií* $\mathrm{dist}^2(f,g) = \mathrm{E}_x[(f(x)-g(x))^2] = \langle f(x)-g(x), f(x)-g(x)\rangle$. Funkcia $h^{\star}(x)$ je tak vlastne definovaná ako funkcia z $H$ s najmenšou vzdialenosťou od $h^{\square}$.

**Lemma 1.3.** *Nech množina hypotéz $H$ je uzavretá na lineárne kombinácie. Potom pre ľubovoľnú funkciu $h \in H$ platí:*

$$\operatorname*{E}_x\left[h(x)(h^{\star}(x)-h^{\square}(x))\right] = \langle h, h^{\star}-h^{\square}\rangle = 0.$$

*Dôkaz.* Nech $h$ je ľubovoľná funkcia z $H$. Uvažujme triedu funkcií $F$, ktoré sú tvaru $f_{\Delta} = h^{\star} + \Delta h$, kde $\Delta$ je ľubovoľné reálne číslo. Všimnime si, že $F \subseteq H$, lebo $H$ je uzavretá na lineárne kombinácie.

Vzhľadom ku leme 1.2, ktorá charaktizuje $h^\star$, musí byť zo všetkých funkcií z $F$ najbližšou k $h^\square$ funkcia $f_0 = h^\star$, z čoho vyplýva, že derivácia funkcie $D(\Delta) = \text{dist}^2(f_\Delta, h^\square)$ musí byť v bode $\Delta = 0$ nulová.

Upravme funkciu $D(\Delta)$:

$$\begin{aligned} D(\Delta) &= \text{dist}^2(f_\Delta, h^\square) && (1.23)\\ &= \langle h^\star + \Delta h - h^\square, h^\star + \Delta h - h^\square\rangle && (1.24)\\ &= \langle (h^\star - h^\square) + \Delta h, (h^\star - h^\square) + \Delta h\rangle && (1.25)\\ &= \langle h^\star - h^\square, h^\star - h^\square\rangle + \Delta^2\langle h, h\rangle + 2\Delta\langle h^\star - h^\square, h\rangle && (1.26)\end{aligned}$$

Vypočítame teraz deriváciu $D(\Delta)$ podľa $\Delta$:

$$\begin{aligned} D'(\Delta) &= \frac{d\langle h^\star - h^\square, h^\star - h^\square\rangle}{d\Delta} + \frac{d\Delta^2\langle h, h\rangle}{d\Delta} + \frac{d(2\Delta\langle h^\star - h^\square, h\rangle)}{d\Delta} && (1.27)\\ &= 2\Delta\langle h, h\rangle + 2\langle h^\star - h^\square, h\rangle && (1.28)\end{aligned}$$

Teraz položíme $\Delta = 0$, a keďže derivácia musí byť v tomto bode nulová, dostávame rovnicu:

$$0 = d'(0) = 2\langle h^\star - h^\square, h\rangle, \tag{1.29}$$

čo bolo treba dokázať. □

Vráťme sa teraz k rovnici (1.3.3). Chceme ukázať, že

$$\mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[\left((\hat{h}_T(x) - h^\star(x)) + (h^\star(x) - y)\right)^2\right]\right] = \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x))^2\right]\right] + \mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(h^\star(x) - y)^2\right]\right],$$

čiže potrebujeme ukázať, že

$$\mathop{\mathrm{E}}_{T}\left[\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - y)\right]\right] = 0$$

Platí:

$$\begin{aligned} \mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - y)\right] &= \mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - h^\square(x) + h^\square(x) - y)\right]\\ &= \mathop{\mathrm{E}}_{x,y}[\underbrace{(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - h^\square(x))}_{\text{nezávisí od } y}]\\ &\quad + \mathop{\mathrm{E}}_{x,y}[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\square(x) - y)]\\ &= \mathop{\mathrm{E}}_{x}[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - h^\square(x))]\\ &\quad + \mathop{\mathrm{E}}_{x}[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\square(x) - \mathop{\mathrm{E}}_{y|x}[y])].\end{aligned}$$

Keďže z definície $h^\square(x) = \mathrm{E}_{y|x}[y]$, druhý sčítanec sa vynuluje a dostávame:

$$\begin{aligned} \mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - y)\right] &= \mathop{\mathrm{E}}_{x}[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - h^\square(x))]\\ &= \langle \hat{h}_T - h^\star, h^\star - h^\square\rangle.\end{aligned}$$

Uvedomme si, že funkcia $h = \hat{h}_T - h^\star$ je lineárnou kombináciou dvoch funkcií z $H$, a preto tiež patrí do $H$. Preto môžeme použiť Lemu 1.3 a dostávame pre ľubovoľnú trénovaciu množinu $T$:

$$\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - y)\right] = \langle h, h^\star - h^\square\rangle = 0,$$

a teda aj:

$$\mathop{\mathrm{E}}_{T}\mathop{\mathrm{E}}_{x,y}\left[(\hat{h}_T(x) - h^\star(x)) \cdot (h^\star(x) - y)\right] = 0,$$

čo sme chceli dokázať.

*Poznámka* 1.1. Označenie $\langle f, g\rangle$ sa vizuálne nepodobá na skalárny súčin vektorov náhodou. V skutočnosti je toto označenie priamočiarym rozšírením skalárneho súčinu vektorov na funkcie a množstvo vlastností skalárneho súčinu vektorov možno analogicky preniesť aj na takto definované skalárne súčiny funkcií. Analogicky ku kolmosti vektorov možno zadefinovať aj kolmosť funkcií $f \perp g$, ak $\langle f, g\rangle = 0$. Lema 1.3 je v takom prípade ekvivalentom tvrdenia o kolmosti v prípade projekcií vektorov do podpriestoru. Napríklad pre priamku $p$ a bod $x$ platí, že ak $y$ je najbližší bod priamky $p$ ku $x$, tak vektor $x - y$ je kolmý na $p$. V tomto texte je $H$ analógom priamky $p$, $h^\star$ je analógom bodu $y$ a $h^\square$ je analógom bodu $x$.

*Poznámka* 1.2. Všetky tvrdenia doposiaľ predpokladali, že v množine hypotéz $H$ existuje funkcia $h^*$ najbližšia ku $H$. Pri niektorých množinách hypotéz však tento predpoklad nemusí byť splnený. Môže sa nám totiž stať, že vieme zostrojiť nekonečnú postupnosť funkcií, kde každá ďalšia funkcia je bližšie ku $h^\square$, no táto postupnosť funkcií konverguje ku funkcii mimo $H$. Aby sme zaručili, že takýto prípad nenastane, musíme do predpokladov zahrnúť aj požiadavku, že $H$ je uzavreté na limity.

# 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:

1. "Akú OTeChAlg môžeme očakávať pri danej veľkosti trénovacej množiny $t$?"
2. "Pri danom $t$, s akou pravdepodobnosťou nám algoritmus vráti hypotézu, ktorej OTeCh je menšia rovná $\varepsilon$?"

Na základe odpovedí na tieto dve otázky potom budeme schopní zodpovedať nasledovné (dalo by sa povedať, že "inverzné") otázky:

3. "Akú veľkú trénovaciu množinu máme zvoliť, aby sme dosiahli dostatočne malú ($\leq \varepsilon$) OTeChAlg?"
4. "Aké $t$ máme zvoliť, aby sme iba s malou pravdepodobnosťou ($\leq \delta$) dostali zlú hypotézu (s OTeCh väčšou ako $\varepsilon$)?"

Odtiaľ sa odvíja názov *PAC učenie* (z anglického *probably approximately correct learning*). Presnejšie, PAC učenie sa zaoberá hlavne otázkami 2 a 4; my sa ale budeme zaoberať všeobecne tým, ako závisia rôzne metriky (OTeChAlg, ...) od počtu trénovacích príkladov $t$.

Uvedomme si rozdiel medzi otázkami (1, 3) a otázkami (2, 4). V prvom zmienenom sa pýtame na očakávaný prípad; ak ale chceme dosiahnuť dobrú hypotézu nie v očakávanom prípade, ale zaručene, vieme to zaručiť iba s určitou pravdepodobnosťou. Vždy sa totiž môže stať (s malou pravdepodobnosťou), že si vytiahneme nereprezentatívne trénovacie dáta. Je teda nutné hovoriť aj o veľkosti tejto záruky, takže hranica $\delta$ je potrebná.

## 2.1 Definície, označenia a predpoklady

Hypotézu $h$ nazveme *zlou*, ak jej OTeCh je väčšia ako $\varepsilon$. Inak ju nazveme *dobrou*. Tieto pojmy budeme používať tak, aby bola hranica $\varepsilon$ vždy jasná z kontextu.

Budeme riešiť hlavne binárne klasifikačné úlohy, v ktorých je cieľom pre každý vstup $x$ povedať, či spĺňa určitú podmienku ($y = 1$) alebo nespĺňa ($y = 0$). Napríklad môžme chcieť rozlíšiť obrázky $32 \times 32$ ktoré obsahujú aspoň jednu mačku od tých, ktoré mačky neobsahujú. Príklady, kde $y = 1$ nazveme *pozitívnymi*; ostatné príklady nazveme *negatívnymi*. Hypotézy v klasifikačných úlohách budeme nazývať *klasifikátormi*.

Pripomíname, že chyba klasifikátora $h$ sa počíta ako

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

Budeme pracovať so štandardným modelom učenia (1.1) spolu s predpokladom všemohúceho trénovacieho algoritmu (vždy vráti hypotézu s najmenšou chybou na trénovacej množine). Okrem toho budeme predpokladať, že množina hypotéz $H$ je *úplná*: existuje v nej "úplne správny

klasifikátor", ktorý na každom vstupe dáva správny výsledok, má teda nulovú testovaciu chybu. Tento klasifikátor je zrejme najlepšou možnou hypotézou v $H$, mohli by sme ho preto značiť štandardne $h^\star$. My ho ale budeme značiť $c$ ako "cieľová hypotéza". Treba si ale dať pozor, že správnych hypotéz v $H$ môže byť viac, a to v prípade, keď niektorým vstupy majú v rozdelení $P$ pravdepodobnosť 0. (V takom prípade je $c$ ľubovoľná jedna z týchto hypotéz.)

Uvedomme si, že nech si vytiahneme ľubovoľnú trénovaciu množinu $T$, vďaka úplnosti množiny $H$ v nej vždy vieme nájsť klasifikátor s nulovou chybou na trénovacej množine. (Napríklad $h^\star$, môžu tam ale byť aj ďalšie klasifikátory, s odlišnými výstupmi na vstupoch mimo $T$.) Preto trénovací algoritmus vždy nájde hypotézu s nulovou trénovacou chybou. Takýto klasifikátor budeme volať *konzistentný klasifikátor*; názov pochádza od toho, že jeho výstupy sú konzistentné s trénovacími príkladmi.

Za týchto predpokladov si budeme klásť nasledovnú otázku: ktoré množiny hypotéz $H$ sa dajú naučiť? Pod "naučením" sa rozumie to, že čím viac trénovacích príkladov algoritmu poskytneme, tým lepšiu hypotézu $\hat{h}_T$ môžeme očakávať; v limite $t \to \infty$ si môžeme byť takmer určite istí, že dostaneme veľmi dobrú hypotézu. Toto sformalizujeme do nasledovnej definície:

**Definícia 1.** Množina hypotéz je *PAC⋆ naučiteľná*, ak pre ľubovoľné pravdepodobnostné rozdelenie $P$ nad $X \times Y$, ľubovoľne malé hranice $\varepsilon > 0$ a $\delta > 0$ existuje hranica $t_0$ na počet trénovacích príkladov taká, že pre všetky počty príkladov $t \geq t_0$ platí: zlú hypotézu (s chybou aspoň $\varepsilon$) dostaneme iba s malou pravdepodobnosťou (nanajvýš $\delta$).

$$\forall P : \forall \varepsilon > 0, \delta > 0 : \exists t_0 : \forall t \geq t_0 : \Pr_{T \sim P^t} \Big( \mathrm{Err}(\hat{h}_T) \geq \varepsilon \Big) \leq \delta$$

Používame názov PAC⋆ naučiteľná preto, lebo skutočná (v literatúre zaužívaná) definícia PAC naučiteľnosti je o čosi komplikovanejšia: berie väčší ohľad na výpočtové detaily, ktorým sa my chceme zatiaľ vyhnúť.

## 2.2 Konečné množiny hypotéz

V tejto časti sa zameriame na konečné množiny hypotéz. Uvedieme niekoľko príkladov takých množín: logické formuly obmedzenej veľkosti, konečné automaty obmedzenej veľkosti, ... V princípe všetko, v čom nevystupujú reálne čísla; s tým že uvažujeme iba malé hypotézy.

### 2.2.1 Konečné množiny sú PAC⋆ naučiteľné

**Veta 2.1.** *Pomocou počtu trénovacích príkladov $t$ vieme zhora odhadnúť pravdepodobnosť, že nám algoritmus vráti zlú hypotézu, nasledovne:*

$$\Pr_T(\mathrm{Err}(\hat{h}_T) \geq \varepsilon) \leq |H| \cdot e^{-\varepsilon t}$$

*Dôkaz.* Predstavme si, že na začiatku vidíme 0 trénovacích príkladov, a postupne sa nám odkrývajú. Sledujme jednotlivé hypotézy: vždy, keď sa odkryje nejaký príklad, môže sa stať, že je nekonzistentný s hypotézou. Každú takúto hypotézu zo súťaže vylúčime. Po odkrytí všetkých $t$ trénovacích príkladov zostávajú už len tie hypotézy, ktoré sú s príkladmi konzistentné a môžu byť výsledkom algoritmu. Ak ani jedna z týchto hypotéz nie je zlá, výsledkom bude dobrá hypotéza. Aká je pravdepodobnosť, že aspoň jedna z týchto preživších hypotéz je zlá?

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

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

Zlých hypotéz je nanajvýš toľko, koľko je všetkých hypotéz, teda $|H|$. Pravdepodobnosť, že aspoň jedna z nich bude konzistentná s príkladmi, je nanajvýš súčet pravdepodobností jednotlivých zlých hypotéz:

$$\Pr_T(\text{prežije aspoň jedna zlá hypotéza}) \leq |H| \cdot e^{-\varepsilon t}$$

Odkiaľ dostávame požadovanú nerovnosť. □

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 hranice $\varepsilon$ a $\delta$.

**Veta 2.2.** *Platí*

$$\mathrm{Err}(L) \leq \frac{1}{t} \cdot (\ln |H| + \ln t + 1)\,.$$

*Dôkaz.* Ak nám algoritmus vráti dobrú hypotézu, vieme jej OTeCh odhadnúť zhora ako $\varepsilon$. Ak nám algoritmus vráti zlú hypotézu, jej OTeCh je nanajvýš 1. Z toho dostávame nasledovný horný odhad na OTeChAlg:

$$\mathrm{Err}(L) = \mathop{\mathrm{E}}_T\left[\mathrm{Err}(\hat{h}_T)\right] \leq \Pr_T\left(\mathrm{Err}(\hat{h}_T) < \varepsilon\right) \cdot \varepsilon + \Pr_T\left(\mathrm{Err}(\hat{h}_T) \geq \varepsilon\right) \cdot 1$$

Pritom sčítance na pravej strane vieme odhadnúť zhora:

$$\Pr_T\left(\mathrm{Err}(\hat{h}_T) < \varepsilon\right) \leq 1$$

a druhý sčítanec vieme odhadnúť pomocou predchádzajúcej vety (2.1). Dostávame tak odhad

$$\mathrm{Err}(L) \leq 1 \cdot \varepsilon + |H| \cdot e^{-\varepsilon t}.$$

My sme si ale mohli zvoliť $\varepsilon$ ľubovoľne. Ak teda chceme dostať čo najlepší odhad, nájdeme také $\varepsilon$, pre ktoré je výraz na pravej strane čo najmenší: zderivujeme a položíme 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 hranice $\varepsilon$ a $\delta$ dostatočne malé.

**Dôsledok 2.3.** *Ľubovoľná konečná množina hypotéz $H$ je PAC⋆ naučiteľná: pre ľubovoľné rozdelenie $P$, ľubovoľne malé hranice $\varepsilon > 0$ a $\delta > 0$, pre počet trénovacích príkladov $t$ spĺňajúci*

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

*platí, že dostaneme zlú hypotézu iba s malou pravdepodobnosťou.*

*Dôkaz.* Podľa vety 2.1 platí

$$\Pr_T\left(\mathrm{Err}(\hat{h}_T) > \varepsilon\right) < |H| \cdot e^{-\varepsilon t}.$$

Stačí nám teda zvoliť také $t$, aby bol výraz na pravej strane nanajvýš $\delta$. Odtiaľ dostaneme postačujúci počet trénovacích príkladov $t$.

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

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

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

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

□

### 2.2.2 Problém konjunkcie

Máme $n$ boolovských premenných. Množinou inštancii $X$ sú všetky možné ohodnotenia týchto premenných, napríklad pre $n = 3$ je jedným možným priradením $x_1 = 0$, $x_2 = 1$, $x_3 = 0$. Každé priradenie si vieme predstaviť ako postupnosť bitov dĺžky $n$. Je zrejmé, že rôznych priradení je $2^n$.

Množinou hypotéz su všetky formuly tvaru $x_{i_1} \wedge x_{i_2} \wedge \ldots \wedge x_{i_k}$. Sú to teda konjunkcie premenných, s tým že premenné vo formule nesmú vystupovať negované. Tieto formuly sú chápané ako funkcie, ktoré vracajú 1 vtedy, keď dané ohodnotenie premenných spĺňa túto formulu. Napríklad $x_1 \wedge x_3 \wedge x_4$ vráti 1 na všetkých vstupoch, kde $x_1 = 1$, $x_3 = 1$ a $x_4 = 1$ (ostatné premenné môžu mať ľubovoľnú hodnotu). Rôznych hypotéz je zrejme tiež $2^n$.

**Metafora.** Vyššie uvedený popis je možno príliš formálny, problém sa dá vysvetliť aj prirodzenejším spôsobom. Predstavme si, že náš kamarát má batoh, do ktorého dal niektoré z $n$ vecí. My chceme zistiť, ktoré z týchto vecí v batohu sú a ktoré v ňom nie sú. Množina inštancii sú teda jednotlivé veci, a hypotézy sú všetky možné obsahy batoha. Čo ale zodpovedá trénovacím príkladom? Predstavme si, že sa môžeme spýtať niekoľko otázok tvaru: "Obsahuje batoh iba týchto $k$ vecí?" (t.j. nič iné nesmie obsahovať) a kamarát nám na každú otázku odpovie áno/nie. Potom trénovacie príklady zodpovedajú práve takýmto otázkam, spolu s odpoveďami na ne.

Zavedieme prirodzenú notáciu pre túto metaforu: namiesto $h(x) = 1$ budeme písať $h \subseteq x$, a namiesto $h(x) = 0$ píšeme $h \not\subseteq x$.

**Všemohúci trénovací algoritmus.** Uvidíme, že konzistentnosť s trénovacími príkladmi je realistická požiadavka: ukážeme, ako sa dá na základe trénovacích príkladov zostrojiť nejaký konzistentný klasifikátor.

**Veta 2.4.** *Nech hypotéza $\hat{h}_T$ obsahuje všetky tie veci, ktoré sa vyskytujú v každom pozitívnom trénovacom príklade. Potom $\hat{h}$ je konzistentná so všetkými trénovacími príkladmi.*

*Dôkaz.* Ak $(x, y)$ je pozitívny príklad, v batohu môže byť iba nejaká podmnožina vecí z $x$. Keď túto úvahu zopakujeme pre všetky pozitívne príklady, dostaneme niekoľko množín "povolených vecí". Skutočný obsah batoha musí byť podmnožinou všetkých z nich, teda je podmnožinou ich prieniku. Toto zároveň zachytáva všetku informáciu zo všetkých pozitívnych príkladov, teda ľubovoľná hypotéza, ktorá je podmnožinou tohto prieniku, je konzistentná s pozitívnymi príkladmi.

Čo ale s negatívnymi príkladmi? Ak by sme zobrali príliš malú množinu vecí ako hypotézu, mohol by nastať problém s negatívnymi príkladmi. (Napríklad ak máme príklady $(0110, 0)$ a $(1100, 1)$, nemôžeme ako hypotézu zvoliť 0100 ani 0000, lebo by to bolo v rozpore s prvým príkladom.)

Ak ale zoberieme celý prienik ako našu hypotézu $\hat{h}_T$, nemôže sa to stať. Totiž, skutočný obsah batoha je podmnožinou oného prieniku, a teda ak $\hat{h}_T \subseteq x$, tak nutne aj $c \subseteq x$. Obmenou tejto implikácie dostaneme, že ak $c \not\subseteq x$, tak aj $\hat{h}_T \not\subseteq x$, takže naša hypotéza dáva na negatívnych príkladoch rovnaké výsledky. □

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

**Veta 2.5.** *Pre problém konjunkcie platí*

$$\mathrm{Err}(L) \leq \frac{1}{t} \cdot (n \ln 2 + \ln t + 1) = O\left(\frac{n + \ln t}{t}\right).$$

**Veta 2.6.** *Pre problém konjunkcie platí, že aby sme zlú hypotézu (s chybou aspoň $\varepsilon$) dostali iba s malou pravdepodobnosťou (nanajvýš $\delta$), stačí zvoliť veľkosť trénovacej množiny nasledovne:*

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

**Dolné odhady.** Tieto odhady sú relatívne tesné. Ako veľmi ich vieme zlepšiť? Vieme dostať aj nejaké dolné odhady, t.j. tvrdenia hovoriace, že ak sa chceme nejakú množinu hypotéz naučiť, potrebujeme aspoň toľkoto trénovacích príkladov?

Netriviálny dolný odhad na všetky prípady sa nedá dostať, nakoľko pravdepodobnostné rozdelenie $P$ môže byť degenerované a "uľahčiť algoritmu robotu". (Napríklad keď $P$ priradí všetkých 100% jednej dvojici $(x, y)$.) Prirodzenejšie je teda hľadať nejaké jedno ťažké rozdelenie $P$. Toto rozdelenie ale môže závisieť od konkrétneho trénovacieho algoritmu; nestačí len predpokladať, že algoritmus je všemohúci, nakoľko možných výsledných hypotéz môže byť viacero a ťažkosť rozdelenia $P$ závisí od toho, ako sa hypotéze darí mimo trénovacích príkladov.

Budeme teda hľadať tvrdenia tvaru: pre každý algoritmus $L$ existuje rozdelenie $P$, na ktorom potrebuje $L$ aspoň nejaké množstvo trénovacích príkladov, aby sa mu dobre darilo.

Dokazovať to v takomto tvare sa ale ukazuje byť ťažké. Problém je v tom, že nájsť konkrétne ťažké rozdelenie $P$ môže byť ťažké. V matematike nie vždy vieme ľahko nájsť konkrétny príklad s nejakou vlastnosťou; možno ale vieme ľahko dokázať, že taký objekt musí existovať. Tak tomu bude aj teraz; v dôkaze budeme využívať nasledovnú všeobecnú metavetu:

**Veta 2.7.** *Ak chceme dokázať, že existuje objekt $o$ spĺňajúci $f(o) \geq l$, kde $f$ reprezentuje nejakú vlastnosť objektu $o$ a hodnota $l$ je nejaká ľubovoľná hranica, stačí dokázať, že pre $o$ náhodne vybrané z nejakého rozdelenia $O$ platí*

$$\mathop{\mathrm{E}}_{o \sim O} [f(o)] \geq l.$$

*Dôkaz.* Ak je hodnota $f(o)$ v priemere aspoň $l$, tak aspoň v jednom konkrétnom prípade (t.j. pre jedno konkrétne $o$) musí byť tiež väčšia rovná $l$. Formálne zapísaná obmena tejto implikácie:

$$(\forall o \in O : f(o) < l) \implies \mathop{\mathrm{E}}_{o \sim O} [f(o)] < l.$$

□

V našom prípade namiesto hľadania jedného konkrétneho ťažkého rozdelenia $P$ ho zvolíme náhodne. Potom ukážeme, že OTeChAlg bude vysoká (pričom stredná hodnota sa berie aj cez náhodné voľby rozdelenia $P$).

**Veta 2.8.** *Pre problém konjunkcie platí, že pre ľubovoľný algoritmus $L$ existuje rozdelenie $P$ také, že pre $t \geq n - 1$ platí pre OTeChAlg platí nasledovný dolný odhad:*

$$\mathrm{Err}(L) \geq \frac{1}{2e} \cdot \frac{n-1}{t+1} = \Omega \left( \frac{n}{t} \right)$$

*Pre $t < n - 1$ platí iný dolný odhad:*

$$\mathrm{Err}(L) \geq \frac{1}{2e} = \Omega(1).$$

*Dôkaz.* Predstavme si, že v rozdelení $P$ vystupujú (t.j. majú nenulovú pravdepodobnosť) iba tie vstupy, ktoré obsahujú presne $n - 1$ vecí, teda práve jedna vec v množine nie je. Označme tieto vstupy $x^{(1)}, \ldots, x^{(n)}$ podľa toho, ktorá vec chýba.

Ľubovoľná hypotéza $h$ vráti na vstupe $x^{(i)}$ hodnotu 1 práve vtedy, keď vec $i$ v hypotéze nie je; hodnotu 0 vráti vtedy, keď tam vec $i$ je. Tieto vstupy teda zodpovedajú otázkam tvaru: "Je pravda, že vec $v$ nie je v batohu?"

Takže každý trénovací príklad nám dáva informáciu iba o prítomnosti jednej veci. Počas testovania teda vieme správnu odpoveď iba na tých vstupoch, ktoré sme dostali ako trénovacie príklady. Na ostatných môžeme vo všeobecnosti iba hádať, či v batohu príslušná vec je alebo nie je.

Takže na vstupoch, ktoré sú v trénovacej množine, budeme mať chybu 0. Ak bol obsah batoha (t.j. výstupy $y$) zvolený rovnomerne náhodne spomedzi všetkých $2^n$ možností, tak na ostatných vstupoch budeme mať očakávanú chybu $\frac{1}{2}$.

Označme si pravdepodobnosti pridelené jednotlivým vstupom $p_1, \ldots, p_n$. OTeChAlg vieme zmerať tak, že si náhodne vytiahneme trénovacie príklady a náhodne vytiahneme testovací príklad. Potom na trénovacích príkladoch natrénujeme hypotézu $\hat{h}_T$, zmeriame jej chybu na testovacom príklade, a spravíme strednú hodnotu tohto procesu. 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 pri testovaní vytiahneme vstup mimo trénovacej množiny, je potom súčtom týchto pravdepodobností cez všetky vstupy $x^{(i)}$:

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

Tieto prípady prispievajú do chyby 1/2, ostatné prípady prispievajú 0. Dostávame tak nasledovný dolný odhad na chybu algoritmu:

$$\mathrm{Err}(L) \geq \frac{1}{2} \cdot \left( \sum_{i=1}^{n} p_i \cdot (1 - p_i)^t \right)$$

Ako zvoliť $p_1, \ldots, p_n$ tak, aby sme dostali dobrý dolný odhad? Skúsme 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 $t \neq n - 1$, nemôžeme zvoliť všetky pravdepodobnosti takéto: pre $t > n - 1$ by bol súčet pravdepodobností primalý a pre $t < n - 1$ priveľký. Rozoberme tieto prípady osobitne.

**Prvý prípad ($t \geq n - 1$).** Všetkým vstupom okrem jedného priradíme pravdepodobnosť $1/(t + 1)$; posledný vstup dostane všetkú zvyšnú pravdepodobnosť.

$$p_1 = p_2 = \ldots = p_{n-1} = \frac{1}{t+1},\ p_n = 1 - \frac{n-1}{t+1}$$

Dosadíme a dostaneme tak odhad

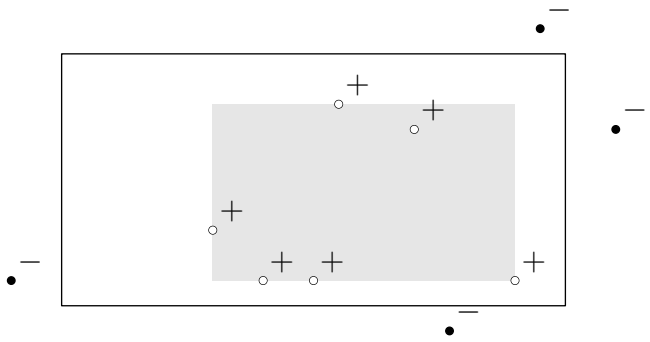
$$\mathrm{Err}(L) \geq \frac{1}{2} \cdot \left( (n-1) \cdot \frac{1}{t+1} \cdot \left(1 - \frac{1}{t+1}\right)^t + \frac{n-1}{t+1} \cdot \left(1 - \frac{n-1}{t+1}\right)^t \right).$$

Druhý sčítanec zdola odhadneme ako 0 a ďalej použijeme odhad

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

ktorý platí pre všetky prirodzené $t$. Dostaneme tak nerovnosť

$$\mathrm{Err}(L) \geq \frac{1}{2e} \cdot \frac{n-1}{t+1}.$$



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

**Druhý prípad ($t < n - 1$).** Prvým $t + 1$ vstupom priradíme pravdepodobnosť $1/(t + 1)$, ostatné vstupy dostanú pravdepodobnosť 0. Dostaneme tak odhad

$$\mathrm{Err}(L) \geq \frac{1}{2} \cdot (t+1) \cdot \frac{1}{t+1} \cdot \left(1 - \frac{1}{t+1}\right)^t \tag{2.7}$$

$$\mathrm{Err}(L) \geq \frac{1}{2e} \tag{2.8}$$

□

## 2.3 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.

### 2.3.1 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 2.9.** *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 2.10.** *Obdĺžnik $\hat{h}$ je podmnožinou cieľového obdĺžnika $c$.*

**Dôsledok 2.11.** *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 2.12.** *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 $\varepsilon$, 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.

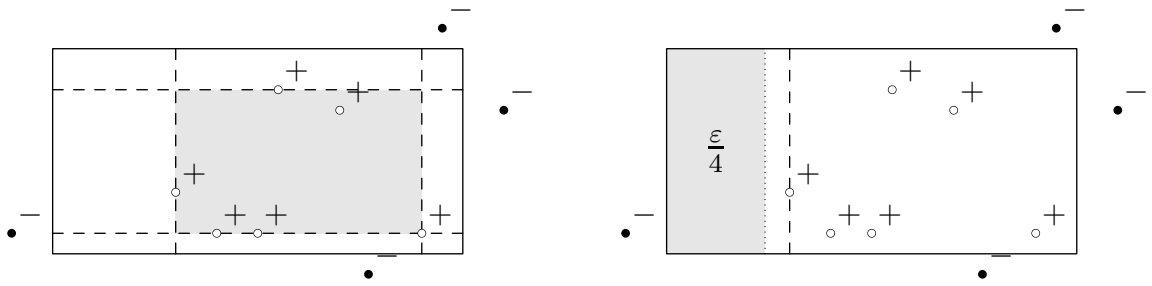
Kedy je váha ľavého pásika malá? Keď aspoň jeden trénovací príklad je dostatočne blízko k ľavému okraju. Keď sa na to pozrieme opačne, váha pásika je veľká, ak žiaden trénovací nie je blízko. Nech $\alpha$ je pravdepodobnosť toho, že trénovací príklad je blízko ľavého okraju. Pravdepodobnosť, že ani 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 $\frac{\delta}{4}$:

$$e^{-\alpha t} \leq \frac{\delta}{4} \tag{2.9}$$

$$-\alpha t \leq \ln \frac{\delta}{4} \tag{2.10}$$

$$t \geq \frac{1}{\alpha} \cdot \ln \frac{4}{\delta} \tag{2.11}$$

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




(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.

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)

□

### 2.3.2 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.

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.

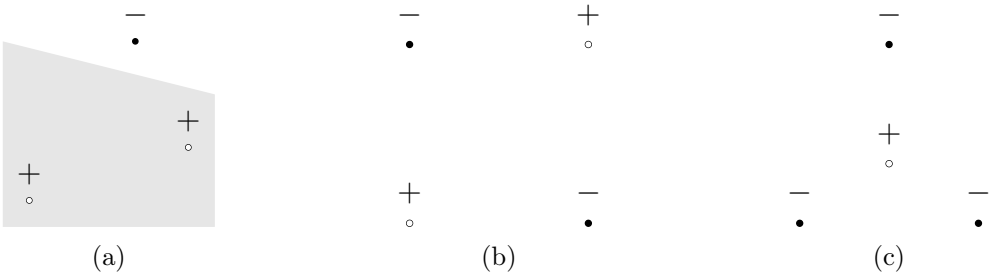
**Definícia 3.** Vapnik-Chervonenkisova dimenzia množiny hypotéz $H$, označovaná $VCD(H)$, je veľkosť najväčšej množiny $S \subseteq X$, ktorá sa dá rozbiť hypotézami v $H$. Ak sa dá rozbiť ľubovoľne veľká množina $S$, definujeme $VCD(H) = \infty$.

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 2.13.** *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). □

Obr. 2.3: Žiaden interval nevytvorí takéto označenie bodov.




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

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).

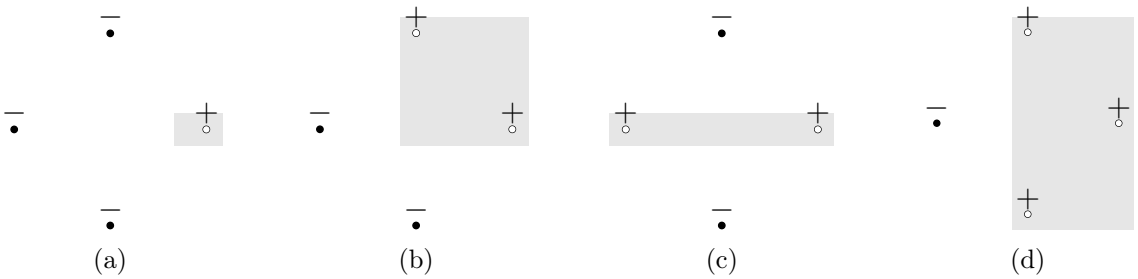
**Intervaly na reálnych číslach.** Nie je ťažké vidieť, že ľubovoľná dvojica bodov sa dá rozbiť. Na druhej strane, ak máme tri body, tak ich vieme označiť ako na obrázku 2.3, čo nie je konzistentné so žiadnym intervalom. Takže VC dimenzia je 2.

**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.

**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 2.14** (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 $t$*




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

*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 2.15** (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 2.16** (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.