UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

Gabriela Andrejková
Ľubomír Antoni

Strojové učenie

Košice 2020, Univerzita Pavla Jozefa Šafárika v Košiciach

EURÓPSKA ÚNIA
Európsky sociálny fond
Európsky fond regionálneho rozvoja

OPERAČNÝ PROGRAM
ĽUDSKÉ ZDROJE

MINISTERSTVO
ŠKOLSTVA, VEDY,
VÝSKUMU A ŠPORTU
SLOVENSKEJ REPUBLIKY

it akadémia

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu
a Európskeho fondu regionálneho rozvoja v rámci Operačného programu Ľudské zdroje

www.minedu.sk www.employment.gov.sk/sk/esf/ www.itakademia.sk

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

STROJOVÉ UČENIE

Spracované v rámci národného projektu
IT Akadémia – vzdelávanie pre 21. storočie

Gabriela Andrejková, Ľubomír Antoni

Košice 2020

Tento projekt sa realizuje vďaka podpore z Európskeho sociálneho fondu v rámci Operačného programu Ľudské zdroje.

Spracované s finančnou podporou národného projektu
IT Akadémia – vzdelávanie pre 21. storočie

Strojové učenie
Vysokoškolský učebný text

Autori:

doc. RNDr. Gabriela Andrejková, CSc.
Ústav informatiky, Prírodovedecká fakulta UPJŠ v Košiciach

RNDr. Ľubomír Antoni, PhD.
Ústav informatiky, Prírodovedecká fakulta UPJŠ v Košiciach

Recenzenti:

doc. PaedDr. Jozef Kapusta, PhD.
Katedra informatiky, Fakulta prírodných vied UKF v Nitre

RNDr. Dávid Hudák, PhD.
VSL Software, a.s.

Umiestnenie: www.unibook.upjs.sk
Dostupné od: 20. 10. 2020

ISBN 978-80-8152-912-2 (e-publikácia)

Kapitola 1

Úvod

Aj keď je strojové učenie často spájané so všeobecnejšou oblasťou umelá inteligencia, jeho realistickejší popis je, že sa zaoberá otázkami ako počítačové programy automaticky rozpoznávajú zložité vzory a robia rozhodnutia na základe dát.

Algoritmus strojového učenia sa[1] je teda algoritmus, ktorý sa učí z dát. Ale čo máme na mysli tým, že "sa učí"? Mitchell (1997) uvádza definíciu: "Počítačový program sa učí zo skúsenosti **E** s ohľadom na určitú triedu úloh **T** a meranie výkonnosti **P**, ak sa jeho výkon pri úlohách v triede **T**, meraný pomocou **P**, zlepšuje použitím skúseností z **E**."V rámci tejto definície si môžeme si predstaviť veľmi širokú škálu skúseností **E**, úloh **T** a meraní výkonnosti **P**.

V týchto vysokoškolských učebných textoch prezentujeme formálnu definíciu toho, čo môže byť použité pre každú z týchto entit. Ale tiež budeme pracovať intuitívne, pretože zavádzanie formalizmu v niektorých prípadoch by značne zväčšilo rozsah tejto publikácie. Budeme sa zaoberať viacerými triedami úloh, skúsenosti budú vyjadrené v dátach a na "učenie sa" budeme používať rôzne algoritmy.

Úlohy

Strojové učenie nám umožňuje riešiť úlohy, ktoré sú veľmi náročné na riešenie použitím programov napísaných a navrhnutých ľuďmi. Z vedeckého a filozofického hľadiska je strojové učenie sa zaujímavé, pretože rozvíjanie nášho chápania strojového učenia znamená rozvíjať naše chápanie princípov, ktoré tvoria základ umelej inteligencie.

V tomto relatívne formálnom vymedzení slova „úloha“, samotný proces „učenia sa“ nie je úlohou. Učenie sa je náš prostriedok na dosiahnutie schopnosti splniť úlohu. Napríklad, ak chceme, aby bol robot schopný chodiť, potom chôdza je úloha. Mohli by sme robota naprogramovať, aby sme ho naučili chodiť, alebo by sme sa mohli pokúsiť priamo napísať ručne program, ktorý špecifikuje, ako má chodiť.

[1] V slovenčine sa používajú slovesá „učiť“ a „učiť sa“, preto je vhodné uvedomiť si, o ktorú aktivitu ide. Algoritmy, ktoré adaptujú svoje parametre na základe nejakých poznatkov „sa učia“.

Typickými príkladmi úloh sú klasifikačné, regresné a aproximačné úlohy.

Meranie výkonnosti

Aby sme mohli vyhodnotiť schopnosti algoritmu strojového učenia, musíme navrhnúť kvantitatívne meranie jeho výkonnosti. Zvyčajne toto meranie P je špecifické pre úlohu T vykonávanú systémom.

V prípade úloh, ako je klasifikácia, často meriame presnosť modelu. Presnosť predstavuje pomer príkladov, pre ktoré model produkuje správny výstup. Ekvivalentné informácie môžeme získať meraním miery chybovosti, podielu príkladov, pri ktorých model produkuje nesprávny výstup.

Voľba typu merania výkonnosti sa môže zdať jednoduchá a objektívna, ale často je veľmi dôležité vybrať si tak, aby meranie zodpovedalo požadovanému správaniu sa systému. V niektorých prípadoch je preto dôležité rozhodnúť, čo sa má merať. Napríklad, pri vykonávaní regresnej úlohy by sme mali penalizovať systém viac, ak často robí stredne veľké chyby alebo ak zriedka robí veľmi veľké chyby? Tieto možnosti výberu závisia od aplikácie.

Skúsenosti

Učiace sa algoritmy môžu byť charakterizované podľa typu učenia sa: bez dozoru alebo pod dohľadom. Ale aké skúsenosti môžu mať počas procesu učenia sa? Väčšina učiacich sa algoritmov v našom prípade bude používať celý súbor údajov. Súbor údajov je súbor mnohých príkladov. Niekedy ich budeme tiež nazývať príklady dátových bodov. Skúsenosť sa teda bude nachádzať v príkladoch. Napríklad, pomocou príkladu sa dá vyjadriť, že to, čo má štyri nohy, operadlo a sedaciu časť, je stolička.

V týchto vysokoškolských učebných textoch venujeme prvé tri kapitoly teoretickým poznatkom z oblasti strojového učenia a výpočtového učenia, v ktorých ukážeme tvorbu konceptov pre objekty, s ktorými budeme pracovať, prípravu konceptov – hypotéz, ktoré budú predstavovať naučený algoritmus a proces učenia sa z príkladov. Táto časť je motivovaná knihou Anthony [1].

Ďašie tri kapitoly sú venované lineárnemu modelovaniu a klasifikácii predovšetkým do dvoch tried. Lineárnemu modelovaniu je venovaná pozornosť aj z hľadiska dôveryhodnosti dosahovaných výsledkov. Klasifikácia je analyzovaná pomocou perceptrónov a pomocou metódy podporných vektorov (Support Vector Machines). Motivácia pre tieto kapitoly a desiatu kapitolu bola nájdená v knihe Rogers & Girolami [6].

Rozhodovacie stromy a metódy zhlukovania sú predmetom ďalších dvoch kapitol, ktoré sú prezentované pomocou pojmov definovaných v knihe Han a kol. [3]. Posledná kapitola sa zaoberá Bayesovským prístupom k strojovému učeniu, v ktorej predovšetkým na príklade vysvetľujeme pravdepodobnosti vystupujúce v Bayesovom vzorci.

Vysokoškolské učebné texty sú určené pre študentov bakalárskych, magisterských, inžinierskych a doktorandských študijných programov informatických odborov.

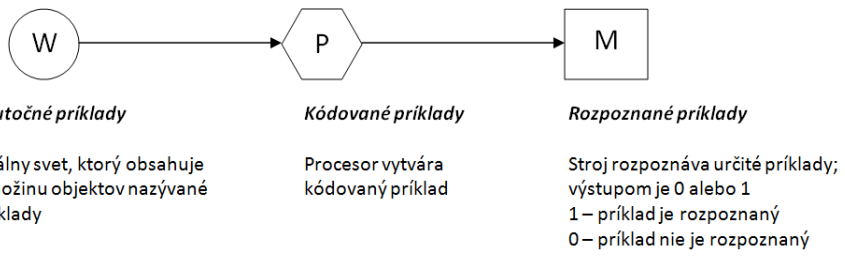
Kapitola 2

Koncepty, hypotézy, učiace sa algoritmy

Existuje viacero typov aktivít označovaných ako "učenie sa". V úvode budeme študovať matematický model takého procesu. Tento model sa zdá byť použitelný, pretože zachytáva základ určitých aktivít, ktoré boli popísané predtým pomocou nepresných výrazov, a zároveň umožnuje vytvoriť netriviálne matematické tvrdenia, ktoré môžu byť dokázané. Terminológia a text tejto kapitoly a nasledujúcich dvoch kapitol je silne motivovaný učebnicou Anthony (1997).

V reálnom svete máme množstvo objektov, ktoré používame v rôznych situáciách, napríklad stoly, stoličky, a pod. Množina rôznych stoličiek tvorí množinu príkladov stoličiek, na základe ktorých sme si vytvorili akýsi model (vzor) stoličky. Napríklad, môžeme brať do úvahy nasledujúce atribúty pre stoličku: má 4 nohy, má operadlo, má sedaciu časť, nemá chvost, na farbe nezáleží. Takýto abstraktný vzor budeme nazývať koncept. Na základe definovaného konceptu budeme vedieť podobné objekty zaradiť alebo nezaradiť k stoličkám.

Na obrázku 2.1 je uvedený postup rozpoznávania objektu, ktorý je použiteľný pre nejaký robot alebo stroj. Stroj dostane zakódované príklady a musí mať algoritmus, na základe ktorého vytvorí svoju odpoveď.



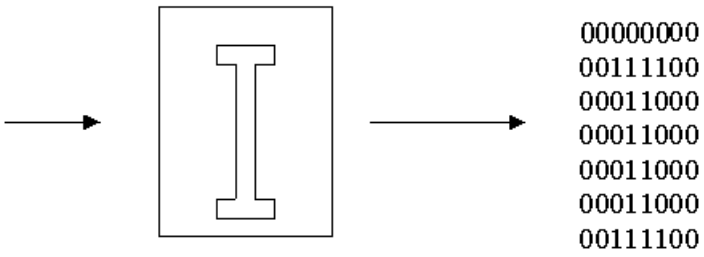
Obr. 2.1: Spracovanie príkladov reálneho sveta pomocou natrénovaného stroja

Kvalita odpovedí stroja M závisí od kvality algoritmu a bolo by vhodné mať algoritmus, ktorý sa dokáže modifikovať na základe spracovaných správne (aj nesprávne) príkladov. Proces adaptácie algoritmu z príkladov budeme nazývať *učenie sa*.

2.1 Koncepty

Sformalizujeme pojem koncept, ktorý budeme popisovať pomocou množiny príkladov. [1]

Nech množina Σ je abeceda na popis príkladov. Napríklad, $\Sigma = \{0, 1\}$, $\Sigma = \mathcal{R}$, množina reálnych čísel.
Množina $\Sigma^n, n \geq 1$, predstavuje množinu všetkých reťazcov dĺžky n v abecede Σ. Do množiny Σ^* patrí prázdny reťazec a všetky reťazce konečnej dĺžky.



Obr. 2.2: Popis písmena I pomocou bitmapy v abecede $\Sigma = \{0, 1\}$.

Koncept, priestor príkladov

Definícia 2.1 *Nech $X \subseteq \Sigma^*$. Množina X sa nazýva priestor príkladov.* **Koncept** *v abecede Σ je funkcia c, $c : X \to \{0, 1\}$. Prvok x, $x \in X$ sa nazýva príklad. Ak pre $x \in X$ platí $c(x) = 1$, tak x je* **pozitívny príklad**, *ak platí $c(x) = 0$, tak x je* **negatívny príklad**.

Zjednotenie množiny kladných a záporných príkladov je definičný obor funkcie c. Teda za predpokladu, že definičný obor je známy, c určuje a je určované množinou svojich pozitívnych príkladov.

Príklad 2.1 *Príklady konceptov*

1. *Koncept PARITA pre reťazce vytvorené v abecede $\Sigma = \{0, 1\}$*

[1] Koncept (z lat.) môže byť:

- prvé predbežné spracovanie, návrh, náčrt (najmä textu);
- nákres, osnova, koncepcia, poňatie
- vo filozofii: formulácia, rozumový obraz, všeobecná myšlienka, pojem
- v logickej sémantike: zmysel mena

$\Sigma = \{0, 1\}$, *príkladový priestor* $X = \Sigma^*$, $c : \Sigma^* \to \{0, 1\}$.

c *definujeme takto: Nech* $y \in \Sigma^*, y = y_1 \dots y_n$, *potom*

$$c(y) = \begin{cases} 1 & \textit{ak v } y \textit{ je nepárny počet jedničiek} \\ 0 & \textit{ak v } y \textit{ je párny počet jedničiek} \end{cases}$$

1011101 *predstavuje kladný príklad,* 1000001 *záporný príklad.*

2. *Koncept* PALINDROM

   $\Sigma = \{0, 1\}$, *príkladový priestor* $X = \Sigma^*$, $c : \Sigma^* \to \{0, 1\}$.

   c *definujeme takto: Nech* $y \in \Sigma^*, y = y_1 \dots y_n$, *potom*

$$c(y) = \begin{cases} 1 & \textit{ak } y_i = y_{n-i+1} \quad i = 1, 2 \dots \frac{n}{2} \\ 0 & \textit{inak} \end{cases}$$

   1011100 *predstavuje záporný príklad,* 1000001 *kladný príklad.*

3. *Koncept* n-ROZMERNÁ JEDNOTKOVÁ GUĽA

   $\Sigma = R$, *príkladový priestor* $X = \Sigma^n$, $c : \Sigma^n \to \{0, 1\}$.

   c *definujeme takto: Nech* $y \in X, y = y_1 \dots y_n$, *potom*

$$c(y) = \begin{cases} 1 & \textit{ak } {y_1}^2 + {y_2}^2 + \ldots + {y_n}^2 \leq 1 \\ 0 & \textit{inak} \end{cases}$$

V prípade $n = 3$, $(0.2, 0.3, 0.1)$ *predstavuje kladný príklad a* $(2, 3, 1)$ *záporný príklad.*

2.2 Trénovanie a učenie sa

Sú dve množiny konceptov používaných v rámci učenia sa popísaného na obr. 2.1.

Konceptový priestor

Prvou množinou je množina konceptov odvodených z reálneho sveta, ktorá je predkladaná na rozpoznanie (napríklad, je to písmeno A). Táto množina môže obsahovať koncepty ako "písmeno A", "písmeno B", ...; každé písmeno môže byť zakódované. Každý koncept má svoje množiny kladných a záporných príkladov. Keď je množina konceptov určovaná týmto spôsobom, budeme pre ňu používať výraz *konceptový priestor* a označovať **K**.

Hypotézový priestor

Druhá množina konceptov obsiahnutých v rámci učenia na tom istom obrázku je množina, ktorú stroj M je schopný rozpoznať. Budeme predpokladať, že M je možné nastaviť do rôznych stavov a v danom stave bude klasifikovať niektoré vstupy ako kladné (výstup 1) a zvyšok ako záporné (výstup 0). Teda stav stroja M určuje koncept, ktorý môžeme chápať ako *hypotézu*. Množina všetkých konceptov, ktoré M určuje, bude nazývaná *hypotézový priestor* a označovaná **H**.

Ak sa vrátime k množine stoličiek, tak vidíme, že na základe konkrétnych príkladov je vytvorený koncept stoličky. Takých konceptov môže byť viac a tvoria konceptový priestor (napríklad konceptový priestor pre nábytok). Hypotézový priestor obsahuje hypotézy, na základe ktorých stroj dáva na výstupe zaradenie vstupného objektu medzi stoličky alebo nie.

Príklad 2.2 *Stoličku je možné popísať pomocou atribútov: 4 nohy, chvost, sedací priestor, zafarbenie, žije. Príklady reprezentácie:*

(4	*nohy,*	*chvost,*	*sedací priestor,*	*zafarbenie,*	*žije),*	*b*
(1		*0*	*1*	*0*	*0),*	*1*
(1		*0*	*1*	*1*	*0),*	*1*

Cieľ procesu učenia sa

Cieľom procesu učenia sa je vytvoriť hypotézu, ktorá v nejakom zmysle zodpovedá konceptu z konceptového priestoru vzhľadom na vyššie uvedené úvahy. V ďalšom texte uvedieme detaily, kedy a akým spôsobom je možné toto urobiť.

Máme teda 2 množiny konceptov:

- K - konceptový priestor – vychádzajúci z reality (koncept sa vytvára z príkladov),
- H - hypotezový priestor – množina hypotéz, ktoré popisujú koncepty zovšeobecneným spôsobom

a **problémom** je ku každému konceptu $k, k \in K$, nájsť nejaké $h, h \in H$, ktoré je jeho dobrou aproximáciou.

V reálnych situáciách sú hypotézy tvorené na základe určitých informácií, ktoré nedávajú explicitnú definíciu k. My budeme predpokladať, že táto informácia je poskytovaná postupnosťou kladných a záporných príkladov patriacich do X. Chceme nájsť hypotézu $h \in H$, ktorá bude popisovať koncept k čo najlepšie.

V praxi sú kladené obmedzenia na výpočtové zdroje, a preto sa musíme uspokojiť s hypotézou h , ktorá "pravdepodobne" reprezentuje k (aproximuje k) v nejakom vopred definovanom zmysle.

Nech $X \subseteq \Sigma^*$ je príkladový priestor. $\Sigma = \{0, 1\}$ alebo $\Sigma = R$. Vzorka dĺžky m je postupnosť m príkladov, t.j. $m-$tica $\overline{x} = (x_1, x_2, \ldots, x_m) \in X^m$, kde x_i sú príklady. Postupnosť môže obsahovať rovnaké príklady viackrát. Niekedy budeme predpokladať, že sú navzájom rôzne (bez újmy na všeobecnosti).

Tréningová vzorka príkladov

Tréningová vzorka $\overline{s}$ je postupnosť príkladov patriaca do množiny $(X \times \{0, 1\})^m$, t.j.

$$\overline{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m)),$$

kde x_i sú príklady a b_i vyjadruje, či príklad je kladný alebo záporný.

Budeme predpokladať, že vo vzorke nie sú žiadne **sporné príklady**, t.j. ak

$$x_i = x_j \quad \Rightarrow \quad b_i = b_j.$$

To teda znamená, že existuje funkcia f, definovaná ako $f(x_i) = b_i \quad (1 \leq i \leq m)$.

Definícia 2.2 *Budeme hovoriť, že* $\overline{s}$ **je tréningová vzorka vhodná pre cieľový koncept** $t \in H$*, ak platí* $b_i = t(x_i)$*, pre* $1 \leq i \leq m$.

Príklad 2.3 *Príklad tréningovej vzorky pre koncept "PALINDROM"*

$$((0010, 0), (1001001001, 1), (111, 1), (010101, 0), (111101, 0))$$

Cieľový koncept $t \in H$*:*

Nech $x = (x^1 \ldots x^n), x \in X$ *je príklad, potom*

$$t(x) = \begin{cases} 1 & ak\ x^1 = x^{n-i+1} \quad pre\ 1 \leq i \leq n \\ 0 & inak \end{cases}$$

Tento cieľový sme vedeli vyjadriť presne jednoduchou funkciou. V praxi pre zložitejšie problémy to nie je také jednoduché.

Učiaci algoritmus pre (K, H)

Uvažujme teraz o povahe učiaceho procesu, ktorý tu chceme študovať. Majme dané a príklady v abecede Σ, K - konceptový priestor a H - hypotézový priestor.

Definícia 2.3 Učiaci algoritmus pre (K, H)*, niekedy nazývaný* (K, H)*-učiaci algoritmus je procedúra, ktorá akceptuje vstupné tréningové vzorky pre koncepty v* K *a výstupy sú v tvare hypotéz v* H.

Aby táto procedúra mohla byť považovaná za algoritmus, musí skončiť a musí byť efektívna. Ak ignorujeme problém efektívnosti, tak učiaci algoritmus pre (K, H) je teda funkcia L, ktorá k ľubovoľnej tréningovej vzorke $\overline{s}$ pre cieľový koncept $t \in C$ priradí funkciu (hypotézu) $h \in H$. Píšeme $h = L(\overline{s})$.

Poznamenajme, že L by mala byť definovaná na celom príkladovom priestore X (mala by byť všeobecnejšia), aj keď $L(\overline{s})$ bola konštruovaná na konečnej podmnožine X (lebo zahŕňa len príklady vzorky $(x_1, ..., x_m)$.

Konzistentná hypotéza, konzistentný algoritmus

Hypotéza $h \in H$ je **konzistentná** s tréningovou vzorkou $\overline{s}$ alebo súhlasí s $\overline{s}$, ak platí $h(x_i) = b_i$, pre $1 \leq i \leq m$.

Vo všeobecnosti nepredpokladáme, že $L(\overline{s})$ je konzistentná s $\overline{s}$, ale keď táto podmienka platí pre všetky tréningové vzorky $\overline{s}$, hovoríme že L **je konzistentný algoritmus**. V tomto prípade je funkcia $L(\overline{s})$ rozširujúca funkcia na celom príkladovom priestore X.

Vo všeobecnosti platí, nie každé rozšírenie tréningovej vzorky bude vhodným zovšeobecnením, pretože cieľový koncept je len parciálne definovaný príkladmi vzorky. Okrem toho tréningová vzorka môže byť nereprezentatívna, alebo zavádzajúca. Napríklad, ak vhodne zakódujeme všetky zvieratá a cieľový koncept je "mačka", tak sa môže stať, že tréningová vzorka pozostáva z bezchvostových mačiek.

Budeme predpokladať, že nereprezentatívne vzorky sú nepravdepodobné a že väčšina vzoriek je dostatočne reprezentatívna, takže rozšírenia funkcií sú vyhovujúce.

Príklad 2.4 *Ak uvažujeme koncept lineárnej separácie bodov v rovine (dve triedy oddelené priamkou), tak konceptovým priestorom sú všetky možné dvojice disjunktných množín bodov.*

Hypotézovým priestorom budú hypotézy, ktoré správne klasifikujú body zadaných množín, napríklad priamky, ktoré oddelia tieto body do dvoch polrovín.

Učiacim algoritmom pre (K, H) tu môže byť učiaci algoritmus pre Rosenblattov perceptrón.

V ďalšom uvedieme dva veľmi jednoduché a veľmi všeobecné algoritmy, ktoré sú ale neefektívne. Neskôr budeme venovať pozornosť efektívnejším algoritmom.

2.3 Učenie sa pomocou konštrukcie

Nech X je príkladový priestor, t cieľový koncept, X^+, $X^+ \subseteq X$ množina kladných príkladov.

Jeden spôsob učenia sa pre cieľový koncept t je skonštruovať množinu X^+ explicitne. Môžme začať s prázdnou množinou prechodom cez tréningovú vzorku pridať každý pozitívny príklad. Formálne to môžeme vyjadriť pomocou nasledujúceho algoritmu.

Algoritmus 2.1 *Učenie sa pomocou konštrukcie*

begin
for all $x \in X$ *set* $h(x) = 0$*;*
for i: = 1 to m do if $b_i = 1$ *then set* $h(x_i) = 1$*;*
$L(\overline{s}) = h$*;*
end

Je zrejmé, že pre záporné príklady a príklady, ktoré neboli v tréningovej vzorke algoritmus bude dávať odpoveď 0. Niektoré otázky, týkajúce sa algoritmu:

1. Čo ak X je nekonečný priestor príkladov? Odpoveď: Tréningová vzorka je vždy konečná, teda zostane nekonečne veľa záporných príkladov.

2. Ako vhodne vyjadriť hypotézový priestor tak, aby hypotézy boli vhodne vyjadriteľné?

Ak odsunieme otázku efektívnosti, zaujímavé sú nasledujúce poznámky. Zrejme, výstupná hypotéza $L(\overline{s})$ je rovná cieľovému konceptu $t \iff$ keď $\overline{s}$ obsahuje všetky kladné príklady pre t. Pretože $\overline{s}$ je konečná postupnosť, to znamená, že len koncepty s konečným počtom kladných príkladov môžu byť naučené úplne.

Napríklad, koncept "PARITA" je definovaný nad celým $\{0,1\}^*$, teda algoritmus nemôže skonštruovať celú množinu kladných príkladov. Ak sa obmedzíme na paritu reťazcov dĺžky n, tak koncept "PARITA" nad $\{0,1\}^n$ je naučiteľný, počet kladných príkladov je 2^{n-1}, preto musíme voliť počet príkladov tréningovej vzorky aspoň tak veľký, t. j. $m = 2^{n-1}$.

Tento algoritmus má aj dobré vlastnosti:

1. je **konzistentný** t.j. výstupná hypotéza $L(\overline{s})$ klasifikuje všetky príklady vyskytujúce sa v $\overline{s}$ korektne.

2. každý príklad tréningovej vzorky sa použije práve 1x. Toto je veľmi silná vlastnosť .... on-line vlastnosť. V praxi to znamená, že príklady môžu byť prezentované učiacemu algoritmu, bez nutnosti mať extra "pamäť", ktorá ich uloží pre ďalšie použitie.

Definícia 2.4 *Hovoríme, že algoritmus je* **bezpamäťový (on-line) algoritmus**, *ak pre danú tréningovú vzorku $\overline{s}$ vytvára postupnosť hypotéz $h_0, h_1, \ldots h_m$, takých, že h_{i+1} závisí len od h_i a od priebežne spracovávaného príkladu vzorky (x_i, b_i).*

2.4 Učenie sa očíslovaním

Nasledujúca metóda učenia určite nie je bezpamäťový on-line algoritmus. Predpokladáme, že hypotézový priestor H je spočítateľný a má explicitné očíslovanie hypotéz, $H = \{h^{(1)}, h^{(2)}, \ldots\}$

Predpokladajme, že $\overline{s}$ je tréningová vzorka pre cieľový koncept t.

Metóda: Porovnať každú hypotézu s každým príkladom v $\overline{s}$, odmietnuť hypotézu, ak nesúhlasí s hodnotou niektorého príkladu. Po odmietnutí hypotézy je tréningová vzorka testovaná tým istým spôsobom pre ďalšiu hypotézu. Proces sa zastaví, keď je nájdená hypotéza, ktorá vyhovuje všetkým príkladom tréningovej vzorky. Formálny zápis je vyjadrený v Algoritme 2.2.

Algoritmus 2.2 *Učenie sa očíslovaním*

Nech r - poradové číslo hypotézy, i - poradové číslo vzorky.

begin
r: = 1; i: = 1;
repeat
if $h^{(r)}(x_i) \neq b_i$ *then*
begin r: = r+ 1; i : = 1 end
else i: = i + 1;
until $i = m + 1$;
$L(\overline{s}) := h^{(r)}$;
end

Množina H môže byť konečná, a teda môže sa stať, že sa vhodná hypotéza nenájde. Modifikáciu algoritmu vieme ľahko urobiť. V praxi sa musíme vyhnúť používaniu neprimeraných veľkých hypotézových priestorov. Počet všetkých hypotéz $h : \{0,1\}^n \rightarrow \{0,1\}$ je 2^{2^n}. Ak $n = 10, \quad 2^{2^n} = 2^{1024} = 4^{512} = 8^{256} = 16^{128}$

Z poznámok vyplýva, že na to, aby sa táto metóda stala vhodnou metódou učenia, je potrebné urobiť určité obmedzenia na hypotézový priestor H a jeho vzťah k priestoru konceptov K. Toto vedie k pojmu **"inductive bias"**.Je to predpoklad, že „učiaci sa“ má nejakú vopred nastavenú ideu o tom, akú metódu klasifikácie učenie sa používa, t.j. učiaci sa vie, alebo má nejaké informácie o konceptovom priestore.

Najjednoduchší spôsob modelovania takého predpokladu je stanoviť $H = K$ a v tomto prípade hovoríme o učiacom algoritme pre H, čo znamená, že pracujeme s (H, H). Väčšina preberaných algoritmov v ďalšom bude tohto typu.

2.5 Úlohy

1. Aký je počet kladných príkladov konceptu "palindrom", keď príkladový priestor je $\{0,1\}^n$?
2. Nech w je nasledujúci koncept: $\{0,1\}^n$, $y \in \{0,1\}^n$, $y = y_1 \ldots y_n$:
$$w(y) = \begin{cases} 1 & \text{ak y obsahuje najviac 2 jedničky} \\ 0 & \text{inak} \end{cases}$$
Ukážte, že počet kladných príkladov v tomto koncepte je kvadratickou funkciou n.
3. Predpokladajme, že v konečnom "učení sa očíslovaním" sme si istí, že hypotézy sú očíslované tak, že tá, ktorú chceme, je v prvej polovici. Ak môžeme spracovať 1 milión hypotéz za sekundu a príkladový priestor je $\{0,1\}^9$, koľko to bude trvať v najhoršom prípade?

Kapitola 3

Booleovské formuly (hypotézy) a ich reprezentácie

Booleovské formuly tvorené logickými spojkami, "AND ($\wedge$), OR ($\vee$), NOT ($\neg$)", nám poskytujú zjednodušený pohľad na koncepty a učiace algoritmy. Keďže sa tu pracuje v abecede $\{0, 1\}$ je jednoduché odhadnúť potrebné počty príkladov tréningovej vzorky, je ľahké vidieť nekonzistentnosť tréningových vzoriek. Budeme sa tu zaoberať dvoma priestormi konceptov, a síce množinou booleovských formúl, ktoré obsahujú len konjunkcie a negácie ("AND, NOT", t.j. monočlenmi) a zložitejším konceptom, v ktorom sú do formúl pridané aj disjunkcie ("AND, NOT, OR").

3.1 Učiaci sa algoritmus pre monočleny

Medzi najjednoduchšie priestory konceptov patrí množina *monočlenov.*

Monočlen

Monočlen je booleovská funkcia vyjadrená ako konjunkcia literálov, t. j. premenných alebo ich negácií. Konjunkcie vo formulách nebudeme vyjadrovať žiadnou spojkou (v tejto kapitole), negáciu premennej vyjadríme čiarou nad ňou. Napríklad formula x_1x_2 vyjadruje $x_1\, AND\, x_2$.

Literály sú premenné a ich negácie. Ak má formula n premenných, môže mať najviac $2n$ rôznych literálov.

Idea algoritmu pochádza od Valianta (1984). Konjunkcia nadobúda hodnotu 1 (true), ak sú všetky literály v nej pravdivé. Odstránením literálov, ktoré to kazia, dostaneme pravdivú formulu.

Začíname bez informácií, t.j. predpokladáme výskyt všetkých $2n$ rôznych literálov vo výslednej formule.

$$h_u : \qquad u_1\bar{u_1}u_2\bar{u_2}\ldots u_n\bar{u_n}$$

Každý pozitívny príklad $y = y_1 \ldots y_n$ umožňuje odstránenie tých literálov u_j, pre ktoré $y_j = 0$ a tých literálov $\bar{u_j}$, pre ktoré $y_j = 1$.

Algoritmus 3.1 *Učenie monočlenov*

Predpokladajme, že $\overline{s}$ *je tréningová vzorka*

$$\overline{s} = ((x_1, b_1), \ldots (x_m, b_m))$$

$$x_i = ((x_i)_1 (x_i)_2 \ldots (x_i)_n), \quad 1 \leq i \leq m$$

h_U *... monočlenná funkcia obsahujúca literály v množine* U.

begin
set $U = \{u_1, \bar{u_1}, \ldots, u_n, \bar{u_n}\}$;
for i:=1 to m do
if b_i*=1 then*
for j:=1 to n do
if $(x_i)_j = 1$ *then delete* $\bar{u_j}$ *from* U
else delete u_j *from* U;
$L(\overline{s}) = h_U$;
end

Tento algoritmus sa nazýva *štandardný učiaci algoritmus pre monočleny.*

Redukovaná tréningová vzorka

Redukovaná tréningová vzorka vznikne z pôvodnej tréningovej vzorky vynechaním tých položiek v tréningových príkladoch, ktoré neovplyvnia hodnotu h_U, t.j. im odpovedajúce premenné (ani ich negácie) sa vo formule h_U nevyskytujú.

Veta 3.1 *Štandardný učiaci algoritmus pre monočleny je konzistentný s redukovanou tréningovou vzorkou, ak je bezosporná.*

Dôkaz: To znamená, že výsledná neprázdna formula je konzistentná s redukovanou tréningovou vzorkou, každý príklad je konzistentne klasifikovaný.

V každom kroku algoritmu je odstránených niekoľko (možno žiadny) literálov. Nech V je množina literálov, z ktorých je vytvorená výsledná formula.

Ak $h_U(x) = 1$ a $V \subset U$, tak $h_V(x) = 1$. Po každej prezentácii kladného príkladu x, mazacia procedúra zaručí, že $h_U(x) = 1$ a teda klasifikácia x je korektná. Teda koncová hypotéza $L(\overline{s})$ korektne klasifikuje všetky pozitívne príklady.

Ľubovoľný negatívny príklad pre cieľový koncept t (t.j. výsledkom t pre tento príklad je 0) je založený na niektorom literále v t, ktorý nebol odstránený.

Redukovaná vzorka je bezosporná, takže kladný a záporný príklad sa líšia v aspoň jednej položke. Nech je to i-tá položka.

Nech u_i je odpovedajúci literál v t. Ak u_i je premenná, tak všetky pozitívne príklady mali v tejto premennej nastavenú hodnotu 1. Záporný príklad bude mať v tejto položke 0. A teda výsledná hodnota t pre tento záporný príklad bude 0.

Ak u_i je negácia premennej, tak všetky pozitívne príklady mali v tejto premennej nastavenú hodnotu 0. Záporný príklad bude mať v tejto položke 1. A teda výsledná hodnota t pre tento záporný príklad bude 0.

Teda všetky negatívne príklady redukovanej tréningovej vzorky pre t (a čiastočne tie vo vzorke) sú korektne klasifikované pomocou $L(\overline{s})$.

Príklad 3.1 *Predpokladajme, že počet premenných vo formule je* $n = 8$ *a je daná tréningová vzorka*

$$(10101010, 1), (00001111, 0), (10101111, 1), (10100011, 1)$$

Na začiatku predpokladáme, že vo formule je všetkých 16 *literálov. Kroky úpravy po použití jednotlivých príkladov:*

1. *Pred 1. príkladom:* $u_1\overline{u}_1u_2\overline{u}_2u_3\overline{u}_3u_4\overline{u}_4u_5\overline{u}_5u_6\overline{u}_6u_7\overline{u}_7u_8\overline{u}_8$
2. *Po 1. príklade:* $u_1\overline{u}_2u_3\overline{u}_4u_5\overline{u}_6u_7\overline{u}_8$
3. *Po 2. príklade:* $u_1\overline{u}_2u_3\overline{u}_4u_5\overline{u}_6u_7\overline{u}_8$. *Druhý príklad je negatívny, teda nič neovplyvní.*
4. *Po 3. príklade:* $u_1\overline{u}_2u_3\overline{u}_4u_5u_7$
5. *Po 4. príklade:* $u_1\overline{u}_2u_3\overline{u}_4u_7$

Algoritmus pre učenie monočlenov má časovú zložitosť $O(m.n)$, kde m je počet príkladov tréningovej vzorky a n je počet premenných monočlena. Počet príkladov je najviac 2^n. Počet možných formúl je 3^n (premenná, jej negácia alebo ani jedna z nich).

3.2 Učenie disjunkcií malých monočlenov

Pri práci s booleovskými premennými sa stretneme s ich štandardnými tvarmi, s ktorými sa dobre pracuje. Sú to

- **Disjunktívna normálna forma (DNF)**

  $$\mu_1 \vee \mu_2 \vee \cdots \vee \mu_n$$

  kde μ_i je monočlen, $1 \leq i \leq r$.

- **Konjuktívna normálna forma (KNF)**

  $$\gamma_1 \wedge \gamma_2 \wedge \cdots \wedge \gamma_n$$

  kde γ_i je, $1 \leq i \leq n$ je klauzula, t.j. disjunkcia literálov.

Tvary booleovských funkcií vieme ovplyvniť napríklad tým, že určíme ohraničenia na počty premenných v klauzulách alebo v monočlenoch.

Označenie:

- M_n-množina monočlenov nad $\{0,1\}^n$,
- $M_{n,k}$-množina monočlenov nad $\{0,1\}^n$, z ktorých každý má najviac k literálov,
- $D_{n,k}$-množina disjunkcií členov z množiny $M_{n,k}$.

Nasledujúci algoritmus navrhol Valiant (1984). Algoritmus začína počiatočnou hypotézou, ktorá je disjunkciou všekých monočlenov dĺžky najviac k. Každý ďalší krok aplikuje jednoduchú logickú dedukciu, a síce: pre záporný príklad je potrebné, aby všetky monočleny mali hodnotu 0, preto odstránime tie monočleny, ktoré to kazia (nadobúdajú hodnotu true).

Algoritmus 3.2 *Učenie disjunkcií malých monočlenov*

Predpokladáme, že dĺžka tréningovej vzorky je m, μ je monočlen a $\mu(x_i)$ predstavuje boolevskú hodnotu monočlena μ pre príklad x_i.

begin
h:= disjunkcia všetkych monočlenov dlžky najviac k;
for i:=1 to m do
if $(b_i = 0) \wedge (h(x_i) = 1)$ *then*
vymazať monočleny μ, pre ktore $\mu(x_i) = 1$;
$L(\overline{s}) := h$;
end

Je zrejmé, že pre negatívne príklady naučený algoritmus bude dávať správne odpovede. V prípade pozitívnych príkladov je potrebné opäť uvažovať bezospornú redukovanú vzorku príkladov, na ktorej sa bude chovať konzistentne.

Časová zložitosť algoritmu je $O(m.d)$, kde d je počet všetkých uvažovaných monočlenov.

Príklad 3.2 *Použitie algoritmu pre učenie disjunkcií malých monočlenov. Budeme pracovať s formulami v $D_{3,2}$.*

Zoznam všetkých relevantných monočlenov je:

$$U = \{u_1, u_2, u_3, \overline{u}_1, \overline{u}_2, \overline{u}_3, u_1u_2, u_1u_3, u_2u_3\},$$

$$\overline{u}_1u_2, \overline{u}_1u_3, \overline{u}_2u_3, u_1\overline{u}_2, u_1\overline{u}_3, u_2\overline{u}_3, \overline{u}_1\overline{u}_2, \overline{u}_1\overline{u}_3, \overline{u}_2\overline{u}_3.$$

Majme tréningovú vzorku:

$$(000, 1), (001, 1), (010, 1), (011, 1), (100, 0), (101, 0), (110, 1), (111, 0)$$

Algoritmus nepoužije prvé štyri príklady, lebo sú kladné. V ďalších príkladoch jeho chovanie je nasledujúce:

1. *z U budú odstránené monočleny $u_1, \overline{u}_2, \overline{u}_3, u_1\overline{u}_2, u_1\overline{u}_3, \overline{u}_2\overline{u}_3$;*
2. *z U budú odstránené monočleny $u_3, u_1u_3, \overline{u}_2u_3, \overline{u}_1\overline{u}_3$;*
3. *U zostane nezmenené;*
4. *z U budú odstránené monočleny $\overline{u}_3, u_1u_2, u_1u_3, u_2u_3$;*

Výsledná formula má tvar $h = \overline{u}_1 \vee \overline{u}_1u_2 \vee \overline{u}_1u_3 \vee u_2\overline{u}_3 \vee \overline{u}_1\overline{u}_2 \vee \overline{u}_1\overline{u}_3$.
Túto formulu je možné zjednodušiť na tvar: $\overline{u}_1 \vee u_2\overline{u}_3$.

3.3 Reprezentácia hypotézového priestoru

Učenie prediskutované v tejto časti malo zjednodušené predpoklady, a síce že konceptový priestor je ten istý ako hypotézový. V skutočnosti sme uvažovali o tom, že cieľové koncepty majú nejaký popis pomocou logických formúl alebo strojov. Hoci tento predpoklad sa môže zdať reštriktívny, je prirodzený pri hľadaní riešení viacerých problémov.

3.4 Čas behu učiacich algoritmov

Učiace algoritmy popísané v predchádzajúcom texte sa zaoberajú booleovskými konceptami. V týchto prípadoch príkladový priestor je $\{0, 1\}^n$ pre nejaké pevné n a hypotézový priestor je množina funkcií definovaných na príkladovom priestore. Pre každý z týchto algoritmov parameter n je ľubovoľný v zmysle, že algoritmus je definovaný pre ľubovoľné n a navyše operuje v podstate tým istým spôsobom pre každú hodnotu n.

Napríklad, štandartný učiaci algoritmus pre priestor M_n monočlenov je definovaný celkom všeobecne, hoci výpočet vždy prebieha pre danú hodnotu n. Chceme kvantifikovať chovanie sa učiacich algoritmov vzhľadom na n, a je obvykle používať nasledujúce definície.

Definícia 3.1 *Hovoríme, že zjednotenie hypotézových priestorov $H = \bigcup H_n$ je odstupňované (graded) príkladmi veľkosti n, ak H_n označuje hypotézový priestor definovaný len na príkladoch z X^n.*

Definícia 3.2 *Nech je daný učiaci algoritmus pre* $H = \bigcup H_n$*, t. j.* $L : X^* \to H$*, taký, že ak* $\overline{s}$ *je tréningová vzorka pre* $h \in H_n$*, tak* $L(\overline{s})) \in H_n$*. Hovoríme, že* L *podporuje stupňovanie (grading) priestoru* H*.*

Uvažujme učiaci algoritmus L pre booleovský hypotézový priestor $H = \bigcup H_n$, odstupňovaný veľkosťou príkladov. Vstup do L je tréningová vzorka, ktorá pozostáva z m n-bitových vektorov spolu s m 1-bitovými označeniami. Celkový počet bitov na vstupe je $m(n+1)$ a bolo by možné použiť toto jediné číslo ako mieru veľkosti vstupu. Avšak je výhodné sledovať aj m aj n oddelene.

Použijeme $R_L(m, n)$ **na označenie najhoršieho času behu** L na tréningovej vzorke m n-bitových vektorov.

Príklad 3.3 *Nech* L *je učiaci algoritmus pre monočleny popísaný vyššie. Hypotézový priestor je zjednotenie* $\bigcup M_n$*. Hlavný krok algoritmu vyžaduje kontrolu každého bitu u každého pozitívneho príkladu a možno vymazanie niektorých literálov. V najhoršom prípade, každý príklad v tréningovej vzorke môže byť pozitívny príklad, a tak by sme mali ošetriť tento krok* m*-krát, každý krok obsahuje kontrolu* n *bitov. Iné časti výpočtu vyžadujú porovnateľne toľko operácií, takže môžme hovoriť, že čas behu* $R_L(m, n)$ *je v tomto prípade* $O(m * n)$*.*

3.5 Konzistencia tréningovej vzorky

Nech $H = \bigcup H_n$ je hypotézový priestor booleovských funkcií odstupňovaný príkladmi veľkosti n. Problém konzistencie tréningovej vzorky pre H môže byť stanovený nasledovne:

H - konzistencia:

- Inštancia: Tréningová vzorka $\overline{s}$ vyjadrená n-bitovými ohodnotenými príkladmi.
- Otázka: Existuje hypotéza v H_n konzistentná s $\overline{s}$?

Ukážeme, že v niektorých netriviálnych prípadoch je tento problém NP-ťažký. Na to, aby sme vyjadrili praktický dosah tohto výsledku, potrebujeme urobiť niekoľko všeobecných komentárov.

Označme C_n^k klauzulu, t. j. disjunkciu vytvorenú výberom najviac k literálov z n premenných. Ukážeme, že pre pevné $k \geq 3$, problém konzistencie pre $C^k = \bigcup C_n^k$ je NP-ťažký. Teda nie je pravda, že existuje polynomiálny učiaci algoritmus pre C_n^k, ktorý produkuje konzistentnú hypotézu.

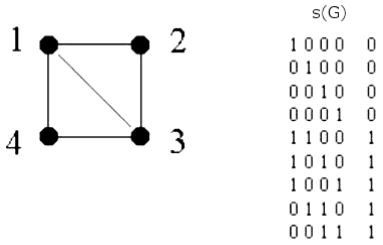
Dôkaz spočíva v prevedení problému farbenia grafu s n vrcholmi pomocou k farieb na tento problém, čo je NP-úplný problém pre $k \geq 3$ (Garey, Johnson 1979).

Problém farbenia grafu:

$G = (V, E), k$-farbenie je funkcia $\chi : V \to \{1, 2, \ldots, k\}$ s vlastnosťou: ak $\langle v_i, v_j \rangle \in E$, potom $\chi(v_i) \neq \chi(v_j)$.

Predpokladajme, že máme $G = (V, E), V = \{1, 2, \ldots k\}$. Skonštruujeme tréningovú vzorku $s(G)$ nasledovne: Pre každý vrchol $i \in V$ určíme záporný príklad vektor v_i, ktorý má 1 v pozícii i-tej súradnice a 0 inde. Pre každú hranu $\langle i, j \rangle \in E$ vezmeme ako pozitívny príklad vektor $v_i + v_j$.

Príklad 3.4 *Na obrázku je znázornený graf* $G = (\{1, 2, 3, 4\}, \{12, 13, 14, 23, 34\})$. *tréningová vzorka je vytvorená príkladmi kvrcholom a hranám.*



Obr. 3.1: Graf spolu s pripravenou tréningovou vzorkou

Veta 3.2 *Existuje funkcia* $h \in C_n^k$, *ktorá je konzistentná so vzorkou* $s(G) \iff$ *graf* G *je* k*-zafarbiteľný.*

Dôkaz:

$\Rightarrow$
Predpokladajme, že $h \in C_n^k$ a je konzistentná s tréningovou vzorkou. Podľa definície h je konjunkcia $h = h_1 \wedge h_2 \wedge \ldots h_k$ klauzúl. Pre každý vrchol $i \in V$, $h(v_i) = 0$ a teda musí ex. aspoň 1 klauzula h_f, pre ktorú $h_f(v_i) = 0$. Funkcia $\chi : V \to \{1, 2, \ldots k\}$ taká, že:

$$\chi(i) = min\{f \quad | \quad h_f(v_i) = 0\}$$

Zostáva ukázať, že χ je farbenie grafu G; inak povedané, ak i a j sú dva vrcholy, pre ktoré $\chi(i) = \chi(j)$, potom $\langle i, j \rangle \notin E$. Predp., že $\chi(i) = \chi(j) = f$ a teda $h_f(v_i) = h_f(v_j) = 0$. Pretože h_f je klauzula, každý literál, ktorý sa v nej vyskytuje musí byť 0 na v_i a na v_j. Teraz v_i má 1 len v i-tej pozícii a tak $h_f(v_i) = 0$ implikuje, že len jeden negovaný

literál, ktorý sa môže vyskytnúť v h_f je $\overline{u_i}$. Pretože to isté platí pre $\overline{u_j}$, dostávame, že h_f obsahuje len niektoré literály u_z, pre ktoré $z \neq i, j$. Teda $h_f(v_i + v_j) = 0$ a $h(v_i + v_j) = 0$. Ak by $\langle i, j \rangle$ bola hranou v G, potom by platilo $h(v_i + v_j) = 1$, pretože sme predpokladali, že h je konzistentná s $s(G)$. Teda (i, j) nie je hranou v G a χ je farbenie.

$\Leftarrow$
Predpokladajme, že je dané farbenie $\chi : V \to \{1, 2, \ldots, k\}$. Pre $1 \leq f \leq k$ definujme h_f ako klauzulu

$$\langle \vee u_{i_{\chi(i) \neq f}} \rangle$$

a definujme $h = h_1 \wedge h_2 \wedge \cdots \wedge h_k$. Tvrdíme, že h je konzistentná s $s(G)$.

Najprv, predpokladajme, že pre vrchol i $\chi(i) = g$. Klauzula h_g je definovaná tak, že obsahuje len tie (nie negované) literály, ktoré zodpovedajú vrcholom nezafarbeným farbou g, a teda u_i sa nenachádza v h_g. Teda $h_g(v_i) = 0$ a $h(v_i) = 0$.

Ďalej, nech (i, j) je hrana v G. Pre každú farbu f aspoň jedno v_i alebo v_j nemá farbu f; označme vhodný výber $i(f)$. Potom h_f obsahuje literál $u_{i(f)}$, ktorý je 1 na $v_i + v_j$. Teda klauzula h_f je 1 na $v_i + v_j$ a $h(v_i + v_j) = 1$, ako sme požadovali.

Príklad 3.5 *Daný graf je 3-zafarbiteľný. Odpovedajúce formuly sú:* $h_1 = v_2 \vee v_3 \vee v_4$, $h_2 = v_1 \vee v_3$, $h_3 = v_1 \vee v_2 \vee v_4$, $h = h_1 \wedge h_2 \wedge h_3$

Obr. 3.2: Graf s ofarbením vrcholov

3.6 Úlohy

1. Napíšte postupnosť hypotéz generovaných algoritmom učenia monočlenov, keď na vstupe je prezentovaná tréningová vzorka

   $$(11100101, 1), (00100011, 0), (11001001, 1)$$

   Ak cieľový koncept je $\langle u_2 \overline{u}_4 u_8 \rangle$, doplňte príklady do vzorky, ktoré sú pre to nutné.

2. Napíšte DNF formuly pre koncepty *parita* a *palindrom* na príkladových priestoroch $\{0, 1\}^5$.

3. Uveďte príklad booleovskej funkcie 3 premenných, ktorá nie je $D_{3,2}$.

4. Prečo je obyčajne vhodné uvažovať veľkosť vstupu v tvare $\lg n$ namiesto n, keď uvažujeme o otázkach efektívnosti?

5. Nasledujúci algoritmus je rýchlym algoritmom pre výpočet m-tej mocniny daného čísla u. (výstupom je finálna hodnota uložená v *bot*.)

```
bot:=1; top:=u; q:=m;
while q>0 do
begin
   if q mod 2 =1 then bot:=top*bot;
   top:=sqr(top);
   q:=q div 2;
end;
```

   Ukážte, že efektívnosť algoritmu je $O(s)$, kde s je miera veľkosti m, ako to bolo povedané v predchádzajúcom príklade.

6. Navrhnite algoritmus, ktorý rozhodne, či daný n-bitový reťazec je palindrom a odhadnite jeho efektívnosť vzhľadom na veľkosť vstupu n.

7. Nech $G = (V, E)$ je graf s množinou vrcholov $V = \{1, 2, 3, 4, 5\}$ a množinou hrán
   $E = \{12, 13, 15, 23, 25, 34, 35\}$. Vytvorte odpovedajúcu tréningovú vzorku $s(G)$ podľa postupu v dôkaze. Nájdite najmenšiu možnú hodnotu k, pre ktorú je funkcia $h \in C_5^k$ konzistentná s $s(G)$ a danú formulu vyjadrite explicitne.

8. Nech $C_n = C_n^1$, čo je priestor booleovských funkcií na $\{0, 1\}^n$, ktorý môže byť reprezentovaný jednou klauzulou. Sformulujte konzistentný učiaci algoritmus pre C_n, ktorý je "duálny" k štandardnému algoritmu pre monočleny a vyhodnoďte tvrdenie, že jeho čas behu je polynomiálny v m a n.

Kapitola 4

Pravdepodobnostné učenie

4.1 Algoritmus pre učenie lúčov

V tejto kapitole sa budeme zaoberať veľmi jednoduchým algoritmom pre učenie sa v reálnom hypotézovom priestore. Budeme sa snažiť odhadnúť, aký presný je dosiahnutý výsledok a určiť do akej miery je dôveryhodný.

Pôjde o učenie sa jednej reálnej meranej hodnoty (napríklad objemu telesa), ktorá existuje, z postupnosti už nameraných hodnôt, pričom vieme, že namerané hodnoty nemôžu byť menšie než skutočná hodnota (skutočný kladný objem).

Lúč

Koncept, ktorým sa budeme zaoberať je **zľava uzavretý a sprava otvorený interval (lúč)** v množine reálnych čísel R, t.j. $\langle \Theta, \infty)$ pre ľubovoľné reálne číslo Θ. Budeme ho označovať r_Θ. Tento koncept je definovaný na príkladovom priestore R funkciou

$$r_\Theta(y) = y \iff y \geq \Theta$$

Algoritmus pre učenie v hypotézovom priestore $H = \{r_\Theta | \Theta \in R\}$ je založený na idee, že za aktuálnu hypotézu vezmeme "najmenší" lúč obsahujúci všetky pozitívne príklady v tréningovej vzorke.

V prípade, že neexistujú kladné príklady, vtedy budeme hovoriť o **prázdnom lúči**. Bude označovaný $r_{+\infty}$.

Pre danú tréningovú vzorku

$$\overline{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m))$$

výstupná hypotéza $L(\overline{s})$ by mala byť r_λ, kde

$$\lambda = \lambda(\overline{s}) = \min_{1 \leq i \leq m} \{x_i | b_i = 1\}$$

$\lambda = +\infty$, ak vzorka neobsahuje kladné príklady. Jednoduchá modifikácia algoritmu, ktorý počíta minimum konečnej množiny je postačujúca pre naše účely. Toto poskytuje nasledujúci bezpamäťový on-line algoritmus:

Algoritmus 4.1 *Učenie lúčov*

Je daná tréningová vzorka $\overline{s} = ((x_1, b_1), (x_2, b_2), \ldots, (x_m, b_m))$ **Učenie lúčov**

begin
set $\lambda = +\infty$*;*
for i:=1 to m do
if $(b_i = 1)$ *and* $(x_i < \lambda\ )$ *then set* $\lambda = x_i$*;*
$L(\overline{s})$*:=*r_λ*;*
end

Je ľahké vidieť, že ak tréningová vzorka je pre cieľovú hypotézu r_Θ, potom $L(\overline{s})$ bude lúč r_λ s $\lambda = \lambda(s) \geq \Theta$. Pretože je len konečný počet príkladov v tréningovej vzorke a celý príkladový priestor je nespočítateľný, nemôžeme očakávať, že $\lambda = \Theta$. Avšak, zdá sa, že ak dĺžka tréningovej vzorky rastie, tak by sa mala zvyšovať pravdepodobnosť toho, že chyba medzi r_λ a r_Θ je veľmi malá (prípadne klesá).

Prakticky táto vlastnosť môže byť charakterizovaná nasledovne:

Predpokladajme, že spustíme algoritmus s veľkou tréningovou vzorkou a potom sa rozhodneme použiť výstupnú hypotézu r_λ pre cieľovú (neznámu) hypotézu r_Θ. Inak povedané, uspokojíme sa s tým, že **"učiaci sa"** bol adekvátne trénovaný. Ak λ nie je blízke k Θ, toto indikuje, že pozitívne príklady, ktoré by boli blízke Θ sú relatívne nepravdepodobné a nevyskytovali sa v tréningovej vzorke. Z toho vyplýva, keď klasifikujeme niektoré ďalšie príklady, ktoré sú síce prezentované podľa toho istého rozloženia, aj tak môžme urobiť niekoľko chýb ako dôsledok použitia r_λ namiesto r_Θ.

Zaujíma nás, akej chyby sa dopustíme a do akej miery je výsledok dôveryhodný. Je zrejmé, že výsledok bude závisieť od tréningovej vzorky, resp. od pravdepodobnostného rozdelenia príkladov.

4.2 Pravdepodobnostné aproximačne správne učenie, PAC učenie

[1]

Uvažujme model, v ktorom tréningová vzorka $\overline{s}$ pre cieľový koncept t je generovaná výberom príkladov $x_1, x_2, \ldots, x_m$ z X "náhodne" podľa nejakého známeho, pevne daného, pravdepodobnostného rozdelenia. Učiaci algoritmus L produkuje hypotézu $L(\overline{s})$, ktorá je očakávaná ako dobrá aproximácia pre t. Vyžadujeme nasledjúce: ak počet príkladov m v trénujúcej vzorke vzrastie, tak z pravdepodobnosti vyplynie, že chyba, ktorá je

[1] PAC - Probably Approximately Correct learning

výsledkom použitia $L(\overline{s})$ namiesto t, je menšia.

Základné pojmy:

X - pravdepodobnostný priestor,

$\mathbf{A}$ - trieda podmnožín množiny X,

μ - pravdepodobnostné rozdelenie, miera pravdepodobnosti, z $\mathbf{A}$ do intervalu $\langle 0, 1\rangle$.

$$\mathbf{A} \rightarrow \quad \langle 0, 1\rangle.$$

Od triedy $\mathbf{A}$ sa vyžaduje, aby bola uzavretá vzhľadom na operácie komplementu, konečného prieniku a spočítateľného zjednotenia.
Prvok $A \in \mathbf{A}$, sa nazýva *udalosť* a $\mu(A)$ je pravdepodobnosť udalosti A.

Od μ sa vyžaduje, aby splňovala nasledujúce podmienky:

$$\mu(\emptyset) = 0, \mu(X) = 1,$$

a pre ľubovoľné po dvoch disjunktné množiny $A_1, A_2, \ldots \in \mathbf{A}$

$$\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i).$$

Pre nás X je príkladový priestor, a príklady sú buď booleovské alebo reálne. V prvom prípade je X konečná alebo spočítateľná množina, a teda $\mathbf{A}$ môže byť triedou všetkých podmnožín X. V reálnom prípade môžeme zobrať za $\mathbf{A}$ ľubovoľnú dostatočne veľkú triedu obsahujúcu množiny, ktoré potrebujeme uvažovať; postačí uvažovať triedu borelovských množín v R^n.

V oboch prípadoch budeme používať vhodnú triedu bez explicitného vyjadrovania detailov.

Budeme jednoducho hovoriť *pravdepodobnostné rozdelenie μ na X*, čím mienime funkciu μ definovanú na vhodnej triede $\mathbf{A}$ a splňujúcej axiómy uvedené vyššie. Treba zdôrazniť, že v aplikáciách, o ktorých sme sa zmieňovali, nerobíme žiadne predpoklady o μ, okrem podmienok uvedených v definícii. Situácia, ktorú sme modelovali, je vyjadrená množinou príkladov prezentovaných učiacemu sa a chovaúcej sa (modeluje) podľa nejakého pevného, ale neznámeho rozdelenia. Učiteľovi je povolené klasifikovať príklady ako pozitívne a negatívne, ale nemôže riadiť postupnosť, v ktorej príklady budú prezentované.

Budeme pokračovať s predpokladom, že cieľový koncept patrí do hypotézového priestoru H, ktorý je dostupný učiacemu sa. K danému cieľovému konceptu $t \in H$ definujeme chybu ľubovoľnej hypotézy $h \in H$ vzhľadom na t a bude to pravdepodobnosť udalosti $h(x) \neq t(x)$, t. j.

$$err_{\mu}(h, t) = \mu\{x \in X \quad | \quad h(x) \neq t(x)\}.$$

v kučeravých zátvorkách je error set - chybová množina a predpokladáme, že existuje udalosť taká, že tejto udalosti môže byť priradená pravdepodobnosť. Keď pôjde o t známe z konceptu, budeme tiež používať označenie $err_\mu(h)$.

Príklad 4.1 *Pravdepodobnosť chyby*

Nech $X = \{0,1\}^3$, predpokladajme, že cieľový koncept je $\langle u_1 \rangle$. Chybová množina pre hypotézu $\langle u_1 \overline{u_2} \rangle$ obsahuje dva príklady, 110 a 111. Tak

$$err_{<u_1\overline{u_2}>} = \mu\{110, 111\}.$$

Napríklad, ak μ - rovnomerné rozdelenie na X - $\frac{1}{8}$ je pravdepodobnosť každého príkladu, potom

$$err_{\langle u_1\overline{u_2}\rangle} = \frac{1}{4}.$$

Ak z nejakých dôvodov príklady, u ktorých je $y_2 = 1$ sú málo pravdepodobné, potom err_μ bude o niečo menšia.

Keď je daná množina X so štruktúrou pravdepodobnostného priestoru, karteziánsky súčin množín X^m preberá pravdepodobnostnú štruktúru X. Detaily nebudeme rozoberať, je postačujúce poznamenať, že konštrukcia nám umožňuje považovať komponenty m-tice $(x_1, x_2, \ldots, x_m)$ za nezávislé premenné, rozdelenie každej z nich je podľa pravdepodobnostného rozdelenia μ na X. Odpovedajúce pravdepodobnostné rozdelenie na X^m je označované μ^m.

Neformálne, pre dané $Y \subseteq X^m$ budeme interpretovať hodnotu $\mu^m(Y)$ ako **"pravdepodobnosť, že náhodná vzorka m príkladov vybratých z X podľa rozdelenia μ patrí do Y"**.

Nech $S(m,t)$ označuje množinu tréningových vzoriek dĺžky m pre daný cieľový koncept t, kde príklady sú vyberané z príkladového priestoru X. Ľubovoľná postupnosť príkladov $x \in X$ determinuje a je determinovaná tréningovou vzorkou $\overline{s} \in S(m,t)$:

ak $x = (x_1, x_2, \ldots, x_m)$, potom $\overline{s} = ((x_1, t(x_1)), (x_2, t(x_2)), \ldots)$.

Inak povedané, existuje zobrazenie Φ

$$\Phi : X^m \to S(m,t), \quad \text{pre ktorú} \quad \Phi(x) = \overline{s}$$

Teda môžeme interpretovať pravdepodobnosť, že $\overline{s} \in S(m,t)$ má nejakú danú vlastnosť P, nasledujúcim spôsobom. Definujeme

$$\mu^m\{\overline{s} \in S(m,t) \quad | \quad \overline{s} \text{ má vlastnosť P } \}$$

to znamená

$$\mu^m\{x \in X^m \quad | \quad \Phi(x) \in S(m,t) \text{má vlastnosť P } \}$$

Z toho vyplýva, že keď príkladový priestor X je vybavený pravdepodobnostným rozdelením μ, môžeme zaviesť precíznejšiu interpretáciu pre

(i) chybu hypotézy, ktorá vznikne, keď učiaci algoritmus L pracuje s $\overline{s}$; Táto veličina pracuje s $err_\mu(L(\overline{s}, t))$ alebo zjednodušene $err_\mu(L(\overline{s}))$.

(ii) a pre pravdepodobnosť, že táto chyba je menšia než vopred zvolené $0 < \epsilon < 1$.

Druhá je pravdepodobnosť vzhľadom na μ^m, teda že $\overline{s}$ má vlastnosť $err_\mu(L(\overline{s})) < \epsilon$

PAC - algoritmus

Definícia 4.1 PAC - algoritmus

Hovoríme, že algoritmus L je **probably approximately correct** *(pravdepodobnostne aproximačne správny) učiaci algoritmus, pre hypotézový priestor H, ak*

- *k ľubovoľnému reálnemu číslu δ, ... dôveryhodnosť , $0 \leq \delta \leq 1$*
- *k ľubovoľnému reálnemu číslu ϵ, ... pravdepodobnosť chyby učenia sa, $0 \leq \epsilon \leq 1$*
- *existuje kladné celé číslo $m_0 = m_0(\delta, \epsilon)$ také ,že*
- *pre ľub. cieľový koncept $t \in H$,*
  pre ľub. pravdepodobnostné rozloženie μ na X pre všetky $m \geq m_0$ platí

$$\mu^m\{\overline{s} \in S(m,t) \quad | \quad err_\mu(L(\overline{s}, t)) < \epsilon\} > 1 - \delta$$

t.j.

$$\forall_{(0\leq\delta\leq1)}\forall_{(0\leq\epsilon\leq1)}\exists_{(m_0=m_0(\epsilon,\delta))}\forall_{(t\in H)}\forall_{(\mu \text{ na } X)}\forall_{(m\geq m_0)}$$
$$\mu^m\{\overline{s} \in S(m,t) \quad | \quad err_\mu(L(\overline{s}, t)) < \epsilon\} > 1 - \delta$$

Skutočnosť, že m_0 závisí od δ a ϵ, ale nie od t a μ odráža to, že učiaci sa môže byť schopný špecifikovať predpokladanú úroveň dôvery a presnosti, aj keď cieľový koncept a rozloženie príkladov sú neznáme. Dôvodom k tomu, že je možné splniť podmienku pre ľubovoľné μ je, že vyjadruje vzťah medzi dvoma veličinami, ktoré obsahujú μ: chyba err_μ a pravdepodobnosť vzhľadom na μ^m určitej množiny.

PAC učenie je, v istom zmysle, najlepšie, v čo môžeme dúfať pri tomto pravdepodobnostnom pohľade. Nereprezentatívne tréningové vzorky, hoci málo pravdepodobné, budú príležitostne prezentované učiacemu algoritmu, ale môžeme očakávať, že je viac pravdepodobné, že je prezentovaná použiteľná tréningová vzorka. Naopak, aj keď máme reprezentatívnu tréningovú vzorku, rozšírenie tréningovej vzorky nebude vo všeobecnosti koincidovať s cieľovým konceptom, takže aj tak výstupná hypotéza bude len aproximačne správna.

4.3 Učenie lúčov je PAC

Veta 4.1 *Algoritmus L pre učenie lúčov je PAC.*

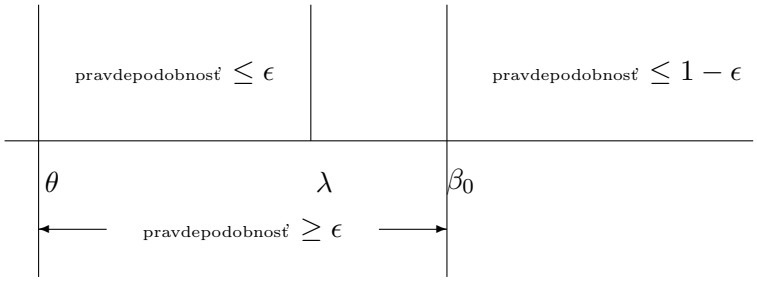
Proof. Predpokladajme, že $\delta, \epsilon, r_\Theta$ a μ sú zvolené, ale sú pevné. Nech $\overline{s}$ je tréningová vzorka dĺžky m pre r_Θ a nech $L(\overline{s}) = r_\lambda$. Zrejme, chybový interval je $\langle \Theta, \lambda)$. Pre danú hodnotu ϵ a dané μ definujme

$$\beta_0 = \beta_0(\epsilon, \mu) = \sup\{\beta \quad | \quad \mu\langle\Theta, \beta) < \epsilon\}$$

Ak uvažujeme $\lambda \leq \beta_0$ tak máme

$$err_\mu(L(\overline{s}, t)) = \mu\langle\Theta, \lambda) \leq \mu\langle\Theta, \beta_0) \leq \epsilon$$

Udalosť, že $\overline{s}$ má vlastnosť $\lambda \leq \beta_0$ je práve udalosť, že aspoň jeden príklad v $\overline{s}$ je v intervale $\langle \Theta, \beta_0\rangle$.



Obr. 4.1: V tejto situácii chyba výstupu je aspoň ϵ

Pretože $\mu\langle\Theta, \beta_0\rangle \geq \epsilon$, pravdepodobnosť, že jeden príklad nie je v tomto intervale, je najviac $1-\epsilon$. Preto pravdepodobnosť, že žiadny z m príkladov vzorky $\overline{s}$ nie je v tomto intervale je najviac $(1-\epsilon)^m$. Keď budeme uvažovať komplementárnu udalosť (existuje príklad, ktorý je z tohto intervalu), z toho vyplýva, že pravdepodobnosť, že $\lambda \leq \beta_0$ je aspoň $1 - (1 - \epsilon)^m$.

Ako sme už poznamenali vyššie, že udalosť $\lambda \leq \beta_0$ implikuje udalosť $err_\mu(L(\overline{s}, t)) \leq \epsilon$ a tak

$$\mu^m\{s \in S(m, r_\Theta) \quad | \quad err_\mu(L(\overline{s}, t)) \leq \epsilon\} \geq 1 - (1 - \epsilon)^m$$

Položíme

$$m \geq m_0 = \frac{1}{\epsilon} * ln\frac{1}{\delta}$$

$$(1 - \epsilon)^m \leq (1 - \epsilon)^{m_0} < e^{-\epsilon m_0} < e^{ln\delta} = \delta$$

tento výpočet ukazuje, že algoritmus je PAC.

Dôkaz korektnosti poskytuje explicitnú formulu pre dĺžku vzorky postačujúcu na to, aby boli splnené predpísané hodnoty presnosti a dôveryhodnosti. Predpokladajme, že $\delta = 0.001$, $\epsilon = 0.01$
$m_0 = \frac{1}{0.01} * \ln \frac{1}{0.001} = 100 * \ln 1000 = 691.$

Takže aspoň 691 príkladov je treba, aby sme si boli istí na 99,9%, že najviac 1% príkladov bude klasifikovaných nesprávne, za predpokladu, že sú z toho istého zdroja ako tréningová vzorka.

Odvodenie vzťahu:

$\delta = (1-\epsilon)^m$
$\ln \delta = m * ln(1-\epsilon) = m * \frac{-\epsilon}{1-\epsilon} \qquad \ln \delta \leq m * \frac{-\epsilon}{1-\epsilon}$
$\ln(1-\epsilon) \leq ln(1) + f'(0) * \frac{\epsilon}{1!} \leq 0 + \frac{1}{1-\epsilon} * (-1) * \frac{\epsilon}{1!}$
$-\ln \delta = m * \frac{\epsilon}{1-\epsilon} \qquad \frac{1-\epsilon}{\epsilon} ln \frac{1}{\delta} \leq m \Rightarrow (\frac{1}{\epsilon} - 1) \ln \frac{1}{\delta} \leq m$
$\frac{1}{\epsilon} * ln \frac{1}{\delta} + \ln \delta \leq m$

4.4 Exaktné učenie

Keď príkladový priestor X je konečný, pojem PAC - učenie má ďalšie dodatočné obmedzenia. Začneme tým, že ľubovoľné pravdepodobnostné rozdelenie na konečnej množine X je determinované hodnotami na jej 1-prvkových množinách $\{x\}$, použitím axiómy o aditivite. Budeme písať $\mu(x)$ namiesto $\mu(\{x\})$. Ak budú nejaké príklady, pre ktoré $\mu(x) = 0$, s pravdepodobnosťou 1 sa nebudú vyskytovať v konečnej náhodnej vzorke a môžu byť ignorované. Inými slovami, môžme, ak je to nutné, predefinovať X tak, že $\mu(x) > 0$ pre všetky $x \in X$. Pretože X je konečný, veličina

$$\epsilon_\mu = min_{\{x \in X\}} \mu(x) > 0$$

je dobre definovaná.

Predpokladajme, že máme algoritmus L, ktorý je PAC pre hypotézový priestor H definovaný na X. Vo význame definície PAC algoritmu máme dané δ, ϵ, μ, a t v ich obvyklom význame

$$m \geq m_0 \Rightarrow \mu^m \{\overline{s} \in S(m,t) \quad | \quad err_\mu(L(\overline{s})) < \epsilon\} > 1 - \delta$$

Predpokladajme, že presnosť ϵ je vybratá tak, aby nebola väčšia než ϵ_μ. Potom podmienka $err_\mu(L(\overline{s})) < \epsilon$ implikuje, že chybová množina pre $L(\overline{s})$ je prázdna, pretože neexistujú žiadne príklady, ktoré majú pravdepodobnosť menšiu než ϵ. Teda podmienka implikuje, že $L(\overline{s}) = t$, t.j. výstupná hypotéza je presne rovná cieľovému konceptu t.

Záverom týchto úvah je, že učenie na konečnom priestore je presné PEC- „probably exactly correct“. Ale je v tom háčik. Jednoduchá vzorka dĺžky m_0 v definícii PAC-učenia závisí od parametrov δ, ϵ ale nezávisí od μ (a t).

Argument uvedený vyššie obsahuje výber ϵ pomocou ϵ_μ, a tak hodnota m_0 vyžadovaná pre exaktné učenie bude závisieť od δ a μ. Toto je v spore s naším originálnym cieľom dokazovania výsledkov, ktoré nie sú závislé od rozdelenia μ (možno neznáme rozdelenie príkladov).

Príklad 4.2 *Štandartný učiaci algoritmus pre monočleny na $\{0,1\}^n$ pre pevné n. Uvedieme "PEC" vlastnosť. Kľúčovým zdelením tu je, že algoritmus poskytuje správne hypotézy, poskytované všetkými kladnými príkladmi, ktoré boli zahrnuté do tréningovej vzorky. Dĺžka tréningovej vzorky rastie a rastie tiež pravdepodobnosť, že vzorka obsahuje všetky kladné príklady; dôsledkom toho pravdepodobnosť spôsobí, že výstup je korektný. Presnejšie, nech ϵ_μ bude najmenšia hodnota $\mu(x)$, ktorú uvažujeme nad množinou príkladov x z $\{0,1\}^n$ s nenulovou pravdepodobnosťou. Potom pravdepodobnosť, že tréningová vzorka dĺžky m neobsahuje daný príklad je najviac $(1-\epsilon_\mu)^m$. Pravdepodobnosť, že existuje jeden z danej množiny p príkladov, ktoré nie sú v tréningovej vzorke je preto $p*(1-\epsilon_\mu)^m$. Ak X^+ je množina kladných príkladov pre daný cieľový koncept t, pravdepodobnosť, že existuje príklad v X^+, ktorý nie je vo vzorke je najviac*

$$|X^+|(1-\epsilon_\mu)^m$$

Potrebujeme vyjadriť m, teda

$$|X^+|(1-\epsilon_\mu)^m < \delta$$
$$m \lg(1-\epsilon_\mu) + \lg|X^+| < \lg\delta$$
$$\lg|X^+| - \lg\delta < -m\lg(1-\epsilon_\mu) < m\epsilon_\mu$$

Použijeme nejaké známe skutočnosti:

$$|X^t| \leq |X| \leq 2^n \qquad a \qquad 1-\epsilon_\mu < exp(-\epsilon_\mu)$$
$$log|X^+| \leq n \qquad log(1-\epsilon_\mu) < -\epsilon_\mu$$
$$m \geq \left\lceil \frac{n}{\epsilon_\mu} ln2 + \frac{1}{\epsilon_\mu} ln\frac{1}{\delta} \right\rceil$$

Poznamenajme, že dĺžka vzorky je nezávislá od t, ale závisí od rozloženia cez parameter ϵ_μ.

Poznámka: Niekoľko variantov PAC - učenia bolo dosiahnutých tak, že bolo umožnené, že učiaci algoritmus a vzorka dostatočnej dĺžky m_0 záviseli nejakým spôsobom buď na rozdelení pravdepodobnosti μ alebo na cieľovom koncepte t. Toto nie je umelé: v mnohých učiacich problémoch je niečo známe - rozdelenie alebo cieľ. Výsledné definície naučiteľnosti sú menej atraktívne než bezkonceptové a bez-rozdelenia PAC definície, ale sú veľmi často ľahko splniteľné. Je možné nájsť veľa publikovaných prác skúmajúcich takéto "neuniformné" PAC učenie.

4.5 Úlohy

1. Napíšte postupnosť hypotéz generovaných algoritmom uvedeným vyššie pre učenie lúčov, ak tréningová vzorka je nasledujúca: (6.1436, 1), (1.5987, 0), (4.2381, 1), (5.7462, 1), (4.3964, 1), (4.2167, 1).

2. Aká je dĺžka tréningovej vzorky pre algoritmus učenia lúčov, aby bola dôveryhodnosť 99.5% a najviac 25% príkladov vybraných podľa nejakého rozdelenia do vzorky bolo zle klasifikovaných?

3. Modifikujte algoritmus pre učenie lúčov tak, že namiesto prázdneho lúča na začiatku vezmeme nejaké veľké číslo. Je tento algoritmus konzistentný?

4. Dané sú dve reálne čísla a, b, a je mešie alebo rovné b. Intervalový koncept $C_{a,b}$ je definovaný takto:

   $C_{a,b}(y) = 1$, ak y je v intervale $[a, b]$, inak je rovné 0.

   Nech H je hypotézový priestor všetkých intervalov, do ktorého patrí aj prázdny interval.

   Nasleduje učiaci algoritmus pre H:

```
    empty:=true;
    for i:-1 to m do
if bi=1 then
   if empty then begin  a:=xi ; b:=xi;
                                empty:= false;
                  end else begin
                            if xi > b then b:=xi;
                            if xi < a then a:=xi end;
    if empty then L(s):=prazdny interval
    else          L(s):=C<a,b>
```

   Dokážte, že L je PAC s vhodne zvolenou dĺžkou vzorky. Akou?

5. Modifikujte algoritmus z predchádzajúceho cvičenia pre priestor intervalových konceptov, ktorý nepoužíva funkciu identicky rovnú nule.

Kapitola 5

Lineárna regresia, logistická regresia

Dôležitým problémom v strojovom učení, ktorý má široké použitie, je odvodenie funkcionálnych vzťahov (učenie sa) medzi atribútovými premennými a s nimi asociovanou odozvou alebo cieľovými premennými tak, že je možné predpovedať odozvu pre ľubovoľnú množinu hodnôt atribútových premenných. Napríklad chceme vytvoriť „model“, ktorý bude predpovedať príchod geomagnetickej búrky o niekoľko hodín. [1] Na to je potrebné mať k dispozícii namerané údaje ďalších veličín, od ktorých je geomagnetická búrka závislá a poznať ich vlastnosti. Iným príkladom by mohla byť predikcia transakcií na burze.

5.1 Lineárne modelovanie

Najjednoduchším príkladom učiacich sa problémov je lineárne modelovanie, t. j. učenie sa lineárnych vzťahov medzi atribútmi a odozvami. V tabuľke 5.1 sú rekordné výsledky v skoku do diaľky u mužov. Na obrázku 5.1 sú tieto výsledky[2] znázornené graficky v paneli A. Rekordy tvoria rastúcu funkciu. Náš cieľ je použiť tieto údaje na učenie pre model funkcionálnej závislosti (ak existuje) medzi rokmi rekordov a dosiahnutými výsledkami. Je nám jasné, že rok nie je jediným faktorom, ktorý ovplyvňuje rekord. Keď by sme chceli predpovedať seriózne, budeme musieť brať do úvahy aj iné faktory, napríklad formu súťažiacich. Avšak na prvý kontakt s lineárnym modelom nám tieto údaje postačia.

Nakoniec by sme chceli vedieť, v ktorom roku by sme mohli očakávať ďalší rekord.

[1] Modelom v strojovom učení bude vlastne program, ktorý rieši daný problém.
[2] wikipedia: *https : //sk.wikipedia.org/wiki/Skok_do_diaľky*

Tab. 5.1: Muži, skok do diaľky

Výkon (m)	Atlét	Miesto	Dátum
7,61	Sp. kráľovstvo Peter O'Connor	Dublin	5. 8. 1901
7,69	USA Edwin Gourdin	Cambridge	23. 7. 1923
7,76	USA Robert LeGendre	Paríž	7. 7. 1924
7,89	USA William DeHart Hubbard	Chicago	13. 6. 1925
7,90	USA Edward Hamm	Cambridge	7. 7. 1928
7,93	Haiti Sylvio Cator	Paríž	9. 9. 1928
7,98	Japonsko Chuhei Nambu	Tokio	27. 10. 1931
8,13	USA Jesse Owens	Ann Arbor	25. 5. 1935
8,21	USA Ralph Boston	Walnut	12. 8. 1960
8,24	USA Ralph Boston	Modesto	27. 5. 1961
8,28	USA Ralph Boston	Moskva	16. 7. 1961
8,31	ZSSR Igor Ter-Ovanesyan	Jerevan	10. 6. 1962
8,31	USA Ralph Boston	Kingston	15. 8. 1964
8,34	USA Ralph Boston	Los Angeles	12. 9. 1964
8,35	USA Ralph Boston	Modesto	29. 5. 1965
8,35	ZSSR Igor Ter-Ovanesyan	Mexiko	19. 10. 1967
8,90	USA Bob Beamon	Mexiko	18. 10. 1968
8,95	USA Mike Powell	Tokio	30. 8. 1991

5.1.1 Definícia modelu

Začneme definíciou nášho modelu ako funkcie, ktorá mapuje naše *vstupné* atribúty, v tomto prípade roky rekordov v skoku do diaľky na *výstupné* alebo cieľové hodnoty - dĺžky skokov.

Je mnoho funkcií, ktoré by sme mohli použiť na toto mapovanie. Ak označíme x vstupný atribút (rok) a t výstupný atribút (dĺžku skoku), matematicky to budeme zapisovať $t = f(x)$. Okrem vstupných atribútov (premenných) funkcia spravidla obsahuje ďalšie parametre, z ktorých niektoré sú modifikovateľné (napríklad učením). Parametre učiaceho sa modelu sú ústrednou témou strojového učenia. Budeme používať zápis $t = f(x; a)$ pre vyjadrenie toho, že funkcia pracuje na atribútoch x a má parametre a. Je zrejmé, že aj premenných aj parametrov môže byť viac.

Prvý lineárny model

Náš prvý model bude lineárny a bude mať jednu premennú a dva parametre,

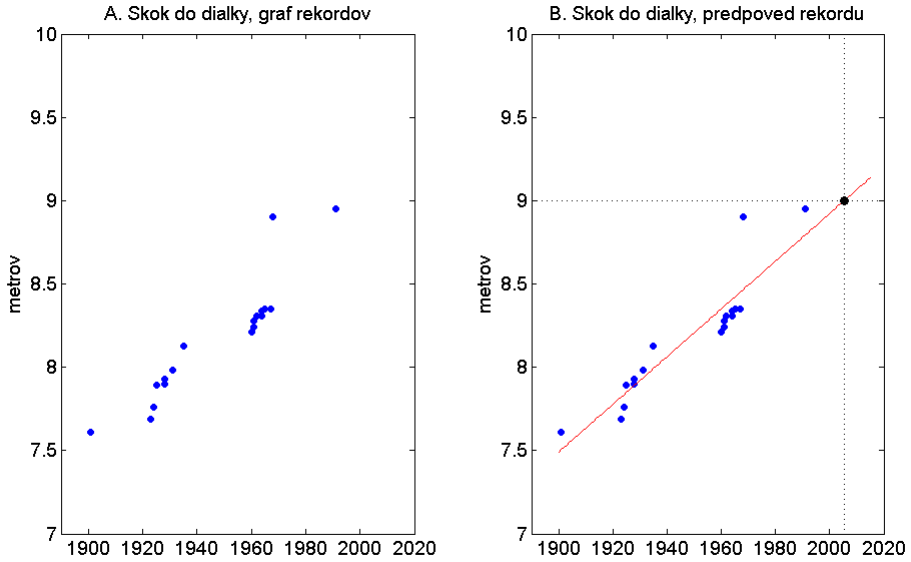
$$y = f(x) = kx + q,$$

kde k, q sú konštanty a x je premenná a z geometrického hľadiska táto funkcia predstavuje priamku.

Predpoklady pre modelovanie Aby sme vedeli vybrať model, potrebujeme určiť predpoklady charakterizujúce vzťah medzi x a t.

Pre prvý model sme stanovili, že vzťah bude lineárny. Dáta na obrázku 5.1 v paneli A by mohli byť modelované pomocou priamky. Teda

$$y = f(x; k, q) = kx + q, \tag{5.1}$$



Obr. 5.1: Graf rekordov skoku do diaľky mužov a predpoveď ďalšieho rekordu

Ohodnotenie modelu

K tomu, aby sme pre k a q vybrali najlepšie hodnoty, potrebujeme definovať, čo to znamená *najlepšie*. Naším cieľom bude, aby model pre všetky vstupné údaje dával výstupy, ktoré sa budú od skutočných výstupných hodnôt líšiť čo najmenej (chyba modelu bude čo najmenšia). Predpokladajme, že sú dané dvojice vstupných a výstupných hodnôt $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$.

Chyba modelu pre jednu dvojicu je

$$E_n(t_n; f(x_n; k, q) = (t_n - f(x_n; k, q))^2 \tag{5.2}$$

Pre všetky dvojice budeme používať kvadratickú chybovú funkciu

$$E = \frac{1}{N} \sum_{n=1}^{N} E_n(t_n; f(x_n; k, q)) \tag{5.3}$$

Najlepším modelom bude taký, pre ktorý bude chyba 5.3 najmenšia. Matematicky to zapíšeme

$$\arg min_{k,q} \frac{1}{N} \sum_{n=1}^{N} E_n(t_n; f(x_n; k, q)), \tag{5.4}$$

čo znamená *nájsť argumenty, ktoré minimalizujú súčet chýb.*

5.1.2 Odvodenie vzťahov pre výpočet k a q

Pre dané dvojice vstupných a výstupných hodnôt $(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)$ máme chybu

$$E = \frac{1}{N}\sum_{n=1}^{N} E_n(t_n; f(x_n; k, q)) = \frac{1}{N}\sum_{n=1}^{N}(t_n - (kx_n + q))^2$$

$$= \frac{1}{N}\sum_{n=1}^{N}(t_n^2 - 2t_n(kx_n + q) + k^2x_n^2 + 2kqx_n + q^2)$$

$$= \frac{1}{N}\sum_{n=1}^{N}(k^2x_n^2 + 2kx_n(q - t_n) + q^2 - 2t_nq + t_n^2) \tag{5.5}$$

K výpočtu minima vypočítame najprv parciálne derivácie podľa k a q Parciálna derivácia podľa k je

$$\frac{\partial E}{\partial k} = \frac{2k}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\sum_{n=1}^{N}(x_n(q - t_n)) \tag{5.6}$$

Parciálna derivácia podľa q je

$$\frac{\partial E}{\partial q} = \frac{2k}{N}\sum_{n=1}^{N} x_n + 2q - \frac{2}{N}\sum_{n=1}^{N} t_n \tag{5.7}$$

Označme $\overline{x}$ priemer hodnôt $x_1, \ldots, x_n$ a $\overline{t}$ priemer hodnôt $t_1, \ldots, t_n$, t. j.

$$\overline{x} = \frac{1}{N}\sum_{n=1}^{N} x_n, \quad \overline{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$$

Položíme (5.7) rovné 0

$$\frac{2k}{N}\sum_{n=1}^{N} x_n + 2q - \frac{2}{N}\sum_{n=1}^{N} t_n = 2k\overline{x} + 2q - 2\overline{t} = 0$$

teda $q = \overline{t} - k\overline{x}$.

Označme $\overline{x^2}$ priemer hodnôt $x_1^2, \ldots, x_n^2$ a $\overline{xt}$ priemer hodnôt $x_1t_1, \ldots, x_nt_n$, t. j.

$$\overline{x^2} = \frac{1}{N}\sum_{n=1}^{N} x_n^2, \quad \overline{xt} = \frac{1}{N}\sum_{n=1}^{N} x_nt_n$$

Položíme (5.6) rovné 0

$$\frac{2k}{N}\sum_{n=1}^{N} x_n^2 + \frac{2}{N}\sum_{n=1}^{N}(x_n(q - t_n)) = 2k\overline{x^2} + 2q\overline{x} - 2\overline{xt} = 0$$

Dosadením $q = \bar{t} - k\bar{x}$ dostaneme

$$2k\overline{x^2} + 2(\bar{t} - k\bar{x})\bar{x} - 2\overline{xt} = 0 \tag{5.8}$$

Úpravou dostaneme výrazy pre k a q

$$k = \frac{\overline{xt} - \bar{x}\bar{t}}{\overline{x^2} - \bar{x}^2}, \quad q = \bar{t} - k\bar{x} \tag{5.9}$$

Výpočtom druhej derivácie sa presvedčíme, že ide o minimum.

Pre náš príklad dostávame výsledky uvedené v tabuľke 5.1.2.

Tab. 5.2: Muži, skok do diaľky a vypočítané hodnoty

Por. č	Rok	Skok	Rok*Skok	Rok*Rok
1	1901	7,61	14466,61	3613801
2	1923	7,69	14787,87	3697929
3	1924	7,76	14930,24	3701776
4	1925	7,89	15188,25	3705625
5	1928	7,90	15231,20	3717184
6	1928	7,93	15289,04	3717184
7	1931	7,98	15409,38	3728761
8	1935	8,13	15731,55	3744225
9	1960	8,21	16091,60	3841600
10	1961	8,24	16158,64	3845521
11	1961	8,28	16237,08	3845521
12	1962	8,31	16304,22	3849444
13	1964	8,31	16320,84	3857296
14	1964	8,34	16379,76	3857296
15	1965	8,35	16407,75	3861225
16	1967	8,35	16424,45	3869089
17	1968	8,90	17515,20	3873024
18	1991	8,95	17819,45	3964081
$(1/N)\sum_{n=1}^{N}$	1947,7	8,1739	15927,40	3793921

Hodnoty pre k, q a E sú

$$k = 0.0143; \quad q = -19.7139; \quad E = 0.0193$$

V paneli B na obrázku 5.1 sa nachádza vypočítaná priamka aj nakreslená a vidíme, aký trend mali rekordy v skoku do diaľky.

5.1.3 Predikcia rekordu

Trendovú priamku môžeme použiť na predikciu, aký rerord by sme mohli očakávať v niektorom z nasledujúcich rokov. Očakávaný rekord v roku 2018 je $y = 0.0143 * 2018 - 19.7139 = 9,14\,m.$ a v roku 2019 je $y = 0.0143 * 2019 - 19.7139 = 9,16\,m.$

Treba tu poznamenať, že tieto presné predikcie sú len orientačné. Toto sa prejavuje aj v predchádzajúci rokoch.

5.2 Zovšeobecnené lineárne modely

Doteraz sme uvažovali lineárne modely, lineárne v parametroch aj v premenných. Ak zavedieme indexované označenie parametrov $\mathbf{w} = (w_1, w_0) = (k, q)$, tak (5.1) môžeme zapísať

$$y = f(x; k, q) = kx + q = w_1 x + w_0 = f(x; \mathbf{w}), \tag{5.10}$$

Model je stále lineárny v parametroch, ale kvadratický v dátach.

Všeobecnejší model je polynomiálny

$$f(x; \mathbf{w}) = \sum_{k=0}^{K} w_k x^k, x^0 = 1; \tag{5.11}$$

Matica dát má tvar

$$\mathbf{X} = \begin{pmatrix} x_1^0 & x_1^1 & x_1^2 & \dots & x_1^K \\ x_2^0 & x_2^1 & x_2^2 & \dots & x_2^K \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^0 & x_N^1 & x_N^2 & \dots & x_N^K \end{pmatrix}$$

Nie je nutné používať len polynomiálne funkcie, je možné použiť K iných funkcií, $h_k(x)$.

Matica dát má potom tvar

$$\mathbf{X} = \begin{pmatrix} h_1(x_1) & h_2(x_1) & h_3(x_1) & \dots & h_K(x_1) \\ h_1(x_2) & h_2(x_2) & h_3(x_2) & \dots & h_K(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_1(x_N) & h_2(x_N) & h_3(x_N) & \dots & h_K(x_N) \end{pmatrix}$$

5.3 Úlohy

1. Analyzujte skoky do diaľky u žien.
2. Na predchádzajúcich dátach vyskúšajte použitie parametricky lineárneho modelu s funkciami:

   $h_1(x) = 1;\ h_2(x) = x;\ h_3(x) = \sin(\frac{x-a}{b})$

   Zvoľte rôzne a a b.

Kapitola 6

Lineárne modelovanie, maximálna dôveryhodnosť

Lineárna regresia je jedným z najzákladnejších algoritmov strojového učenia a jednoducho sa snaží analyzovať naše údaje pomocou preloženej priamky s najmenšou chybou.

V tejto kapitole uvidíme lineárnu regresiu z inej perspektívy, a síce z hľadiska pravdepodobnosti, najmä pomocou odhadu maximálnej dôveryhodnosti - Maximum Likelihood (ML). Budeme explicitne modelovať **šum** (chyby medzi modelom a pozorovaniami) v dátach pomocou **náhodnej premennej**.

6.1 Chyby ako šum

Pri vytváraní lineárneho modelu sme predpokladali lineárny vzťah medzi nezávislými premennými a závislou premennou. Tento model zachytáva všeobecný trend v dátach, ale ignoruje to, že v niektorých bodoch sa dopúšťa veľkej chyby. Z hľadiska modelu je ťažké tieto chyby obhájiť. Vieme, že chyby v dátach budú a mali by sme ich pri tvorbe modelu zohľadniť. Je možné s modelom experimentovať, modifikovať jeho parametre, ale je tu ešte iný prístup. Budeme brať náš modelovací problém ako **generatívny**, to znamená, že môžeme vytvárať model, ktorý bude použitý na vytvorenie (generovanie) datasetu, ktorý vyzerá podobne ako náš skutočný dataset.

Poznamenajme si niektoré vlastnosti chýb (majme na mysli náš príklad rekordov):

- Chyby sú rôzne v každom roku (bode x). Niekdy sú pozitívne, niekedy negatívne a majú rôzne hodnoty.
- Nie je žiadny vzťah medzi veľkosťou chýb v rokoch (bodoch x). Chyby nie sú funkciou roka (x).

Použijeme náhodnú premennú ϵ_n na vyjadrenie chyby v roku $\mathbf{x_n}$ a náš model teraz bude vyzerať takto:

$$t_n = \mathbf{w}^T \mathbf{x}_n + \epsilon_n \tag{6.1}$$

Na doplnenie definície modelu potrebujeme rozhodnúť o štatistickom rozdelení ϵ_n. Najprv si vyjasníme, že rozdiel medzi modelom a dĺžkami skokov je spojitá veličina. Preto ϵ_n je spojitá náhodná premenná. Tiež si uvedomíme, že nemáme len jednu náhodnú premennú, ale máme pre každý rok jednu. Môžeme predpokladať, že tieto hodnoty sú nezávislé preto pre pravdepodobnosť p všetkých n skokov platí

$$p(\epsilon_1, \ldots, \epsilon_n) = \Pi_{n=1}^{N} p_n(\epsilon_n) \tag{6.2}$$

Ďalší predpoklad sa týka tvaru rozdelenia $p_n(\epsilon_n)$. Budeme predpokladať, že pre všetky n je to Gaussovo (normálne) rozdelenie s priemerom 0 a varianciou σ^2. Vieme, že tak ϵ_n môže nadobúdať kladné aj záporné hodnoty a má zaujímavé modelovacie vlastnosti.

Použitím normálneho rozdelenia pre ϵ, t.j.

$$p(y|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{1}{2\sigma^2}(y-\mu)^2) \tag{6.3}$$

s $\mu = 0$ a $\sigma^2 = 0.05$ dostaneme realistickejší dataset.

Náš model teraz pozostáva z dvoch komponentov: **Trend a šum**

- deterministický komponent ($\mathbf{w}^T\mathbf{x_n}$), ktorý sa nazýva **trend** alebo **drift**,
- náhodný komponent (ϵ_n), ktorý sa nazýva **šum**.

Je pravda, že šum môže mať iné štatistické rozdelenie než Gaussovo a tiež nemusí byť aditívny, ale multiplikatívny ($t_n = \mathbf{w}^T\mathbf{x}_n\epsilon_n$), ktorý sa často využíva pri zašumení obrázkov.

6.2 Dôveryhodnosť (hodnovernosť)

Náš model je teraz v nasledujúcom tvare

$$t_n = f(\mathbf{x}_n; \mathbf{w}) + \epsilon_n, \epsilon_n \sim \mathcal{N}(0, \sigma^2), \tag{6.4}$$

kde $\mathcal{N}(0, \sigma^2)$ predstavuje Gaussovo (normálne) rozdelenie s priemerom μ a varianciou σ^2.

Potrebujeme nájsť optimálnu hodnotu pre $\mathbf{w}$, ktorú označíme $\hat{\mathbf{w}}$. Tiež tu máme ďalší parameter σ^2, ktorý je potrebné nastaviť. V regresnom

modeli chyba bola meraná pomocou rozdielu pozorovanej hodnoty a hodnoty, ktorú predikoval model.

Efekt pridania náhodnej premennej do modelu je, že výstup modelu, t, je teraz sám náhodná premenná. Inak povedané, neexistuje len jedna hodnota t_n pre partikulárne $\mathbf{x}_n$. Teda nie je možné použiť chybu ako prostriedok pre optimalizáciu $\mathbf{w}$ a σ^2.

Pridanie konštanty ($\mathbf{w}^T\mathbf{x}_n$) do Gaussovej náhodnej premennej je ekvivalentné inej Gaussovej náhodnej premennej s priemerom posunutým o uvedenú konštantu:

$$y = a + z$$
$$p(z) = \mathcal{N}(m, s)$$
$$p(y) = \mathcal{N}(m + a, s)$$

Preto náhodná premenná t_n má nasledujúcu funkciu hustoty

$$p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{w}^T\mathbf{x}_n, \sigma^2) \tag{6.5}$$

t_n teda závisí od partikulárnych hodnôt $\mathbf{x}_n$ a $\mathbf{w}$ (určujú priemer) a od σ^2 (určuje varianciu).

Aby sme videli, ako môžeme toto použiť na hľadanie optimálnej hodnoty $\mathbf{w}$ a σ^2, vyberme jeden rok z datasetu - 1980. Na základe modelu predchádzajúceho (základného) lineárneho modelu majme určené (w_0, w_1) a predpokladajme, že $\sigma^2 = 0.05$. Na obrázku 6.2 je znázornená $p(t_n|x_n = 1980, \mathbf{w}, \sigma^2)$ ako funkcia t_n. Pre tento rok dostaneme

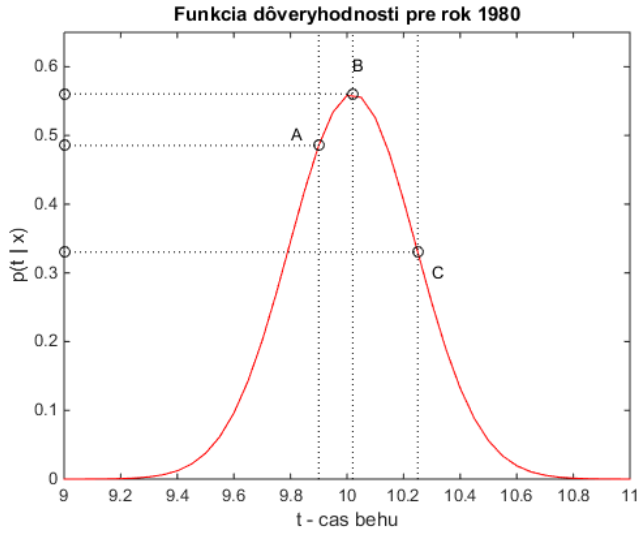
$$p(t_n|\mathbf{x}_n = [1, 1980]^T, \mathbf{w} = [36.416, -0.0133]^T, \sigma^2 = 0.05), \tag{6.6}$$

pri Gaussovom rozdelení $\mu = 36.416, -0.0133 * 1980 = 10.02$ a $\sigma^2 = 0.05$.

Pripomeňme, že pre spojitú premennú $t, p(t)$ nemôže byť interpretovaná ako pravdepodobnosť. Výška krivky v partikulárnej hodnote pre t môže byť interpretovaná takto: nakoľko je hodnoverné (dôveryhodné), že budeme pozorovať túto partikulárnu hodnotu t v danom roku $x = 1980$. Najhodnovernejšia hodnota v roku 1980 by bola 10.02 sekúnd (pre Gaussovo rozdelenie najhodnovernejší - najvyšší bod zodpovedá priemeru). Na obrázku máme 3 príklady časov - A, B, C. Z nich B je najhodnovernejší, a C najmenej hodnoverný.

Skutočný nameraný čas v roku 1980 je C (10.25 sekúnd). Hustota pravdepodobnosti $p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ vyhodnotená v $t_n = 10.25$ je dôležitá veličina, ktorá je známa ako **dôveryhodnosť (hodnovernosť)** n-tého dátového bodu. Nemôžeme zmeniť $t_n = 10.25$ (máme to v dátach), ale môžeme zmeniť $\mathbf{w}$ a σ^2 a skúšať, posúvať a hľadať čo najväčšiu dôveryhodnosť.

Idea hľadania parametrov, ktoré maximalizujú dôveryhodnosť týmto spôsobom, je kľúčový koncept v strojovom učení.



Obr. 6.1: Graf pre 1980, x-ová os t, y-ová os $p(t|x)$

6.3 Hodnovernosť datasetu

V skutočnosti sa sa nezaujímame o dôveryhodnosť jedného dátového bodu, ale o dôveryhodnosť celého datasetu. Označme $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, $\mathbf{t} = [t_1, \ldots, t_N]$. Keď máme N bodov, zaujíma nás podmienená pravdepodobnosť

$$p(t_1, \ldots, p_N | \mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{w}, \sigma^2) = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2).$$

Vyhodnotenie tejto hustoty v pozorovaných dátových bodoch dáva jednu hodnotu dôveryhodnosti pre celý dataset, ktorú môžeme optimalizovať zmenou $\mathbf{w}$ a σ^2.

Predpoklad, že šum v každom dátovom bode je nezávislá veličina, znamená $p(\epsilon_1, \ldots, \epsilon_N) = \Pi_n p(\epsilon_n)$ a umožňuje nám použiť zaujímavé odvodenia. Táto spojená (joint) podmienená hustota môže byť rozdelená do N separatných výrazov, jeden pre každý dátový objekt:

$$L = p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \Pi_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \sigma^2) = \Pi_{n=1}^{N} \mathcal{N}(\mathbf{w}^T \mathbf{x}_n, \sigma^2) \qquad (6.7)$$

Nepovedali sme, že t_n hodnoty sú navzájom úplne nezávislé. Sú v priemere v čase klesajúce, a teda ponúkajú jasnú štatistickú závislosť. Ak by boli úplne nezávislé, nemohli by sme vytvárať model. V skutočnosti sú **podmienene nezávislé** – vzhľadom k daným hodnotám $\mathbf{w}$ (deterministická časť modelu) t_n sú nezávislé (model ponúka výpočet v ľubovoľnom bode); bez nich nie sú celkom nezávislé.

Predstavme si, že nám chýbajú dáta z jedného roku, napríklad 1960. Ostatné roky sú v poriadku. $\mathbf{X}, \mathbf{t}$ neobsahujú údaje o roku 1960. Z daných dát by sme chceli vedieť niečo o roku 1960. Teda nás zaujíma

$$p(t_{1960}|\mathbf{x}_{1960}, \mathbf{X}, \mathbf{t})$$

Z definície podmienenej pravdepodobnosti dostaneme

$$p(t_{1960}|\mathbf{x}_{1960}, \mathbf{X}, \mathbf{t}) = \frac{p(t_{1960}, \mathbf{t}|\mathbf{x}_{1960}, \mathbf{X})}{p(\mathbf{t}|\mathbf{X})}$$

Za prepokladu, že $\mathbf{t}$ sú nezávislé, výsledky v t_{1960} závisia len od $\mathbf{x}_{1960}$:

$$p(t_{1960}|\mathbf{x}_{1960}, \mathbf{X}, \mathbf{t}) = \frac{p(t_{1960}|\mathbf{x}_{1960})\Pi_n p(t_n|\mathbf{x}_n)}{\Pi_n p(t_n|\mathbf{x}_n} = p(t_{1960}|\mathbf{x}_{1960})$$

Avšak pre náš model toto nie je použiteľné, t_{1960} musí byť v nejakom zmysle závisle od ostatných dát. Táto závislosť je zapúzdrená v parametroch $\mathbf{w}$. Ak poznáme $\mathbf{w}$, tak to, čo zostáva neznáme, je chyba medzi pozorovanými dátami a $\mathbf{w}^T\mathbf{x}_n$. Predpokladáme, že tieto chyby sú nezávislé. Bez modelu pozorovania sú nezávislé, ale keď berieme do úvahy model ($\mathbf{w}$), pretože je tu chyba, pozorovania nie sú nezávislé.

6.4 Maximálna dôveryhodnosť

Budeme hľadať parametre modelu $\mathbf{w}$ a σ^2 také, ktoré urobia náš model najdôveryhodnejší. Z analytických dôvodov budeme maximalizovať prirodzený algoritmus dôveryhodnosti L (logaritmus je rastúca funkcia a teda je to možné urobiť).

$$\log\, L = \Sigma_{n=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{1}{2\sigma^2}(t_n - f(\mathbf{x}_n; \mathbf{w}))^2\}\right)$$

$$= \sum_{n=1}^{N}\left(-\frac{1}{2}\log(2\pi) - \log\sigma - \frac{1}{2\sigma^2}(t_n - f(\mathbf{x}_n; \mathbf{w}))^2\right)$$

$$= -\frac{N}{2}\log 2\pi - N\log\sigma - \frac{1}{2\sigma^2}\Sigma_{n=1}^{N}(t_n - f(\mathbf{x}_n; \mathbf{w}))^2$$

Substitúciou $f(\mathbf{x}_n; \mathbf{w}) = \mathbf{w}^T\mathbf{x}_n$ dostaneme

$$L = -\frac{N}{2}\log 2\pi - N\log\sigma - \frac{1}{2\sigma^2}\Sigma_{n=1}^{N}(t_n - \mathbf{w}^T\mathbf{x}_n)^2 \tag{6.8}$$

Parciálne derivácie podľa $\mathbf{w}$

$$\frac{\partial \log L}{\partial \mathbf{w}} = \frac{1}{\sigma^2}\Sigma_{n=1}^{N}\mathbf{x}_n(t_n - \mathbf{w}^T\mathbf{x}_n)$$

$$= \frac{1}{\sigma^2}\Sigma_{n=1}^{N}\left(\mathbf{x}_n t_n - \mathbf{x}_n\mathbf{x}_n^T\mathbf{w}\right) = 0$$

Označme

$$\mathbf{X} = [\mathbf{x}_1^T\mathbf{x}_2^T \ldots \mathbf{x}_N^T]^T, \mathbf{t} = [t_1 t_2 \ldots t_N]^T \tag{6.9}$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}$$

Maximálna dôveryhodnosť z hľadiska $\mathbf{w}$ je v maticovom tvare

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t} \tag{6.10}$$

Pre σ^2 dostávame

$$\frac{\partial L}{\partial \sigma} = -\frac{N}{\sigma} + \frac{1}{\sigma^3}\Sigma_{n=1}^{N}(t_n - \mathbf{x}_n^T\hat{\mathbf{w}})^2 = 0 \tag{6.11}$$

$$\hat{\sigma}^2 = \frac{1}{N}\Sigma_{n=1}^{N}(t_n - \mathbf{x}_n^T\hat{\mathbf{w}})^2 \tag{6.12}$$

Tento výraz je perfektný - variancia je jednoducho priemer kvadratickej chyby. V maticovom označení

$$\hat{\sigma}^2 = \frac{1}{N}(\mathbf{t} - \mathbf{X}\hat{\mathbf{w}})^T(\mathbf{t} - \mathbf{X}\hat{\mathbf{w}}) = \frac{1}{N}(\mathbf{t}^T\mathbf{t} - 2\mathbf{t}\mathbf{X}\hat{\mathbf{w}} + \hat{\mathbf{w}}^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{w}}) \tag{6.13}$$

Toto zjednodušíme substitúciou (6.10), navyše platí $\hat{\mathbf{w}}^T = \mathbf{t}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}$ a $(\mathbf{X}^T\mathbf{X})^{-1}$ je symetrická, a preto sa rovná svojej transponovanej matici.

Pre rekordy v behu dostávame

$$\hat{\mathbf{w}} = [36.4165, -0.0133]^T, \sigma^2 = 0.0503$$

$\mathbf{w}$ je to isté ako pri pri lineárnej regresii, σ^2 predstavuje varianciu Gaussovho šumu, ktorý sme predpokladali v dátach.

6.5 Charakteristika max. dôveryhodnosti

Aby sme dokázali, že je to maximum, mali by sme vypočítať druhú deriváciu vo vypočítanom bode. Hodnota $\log L$ v maxime je

$$\log L = -\frac{N}{2}\log 2\pi - N\log\hat{\sigma^2} - \frac{1}{2\hat{\sigma^2}}N\hat{\sigma^2} \tag{6.14}$$

$$= -\frac{N}{2}(1 + \log 2\pi) - \frac{N}{2}\log\hat{\sigma^2}$$

L bude rásť, keď $\hat{\sigma^2}$ bude klesať. Toto ale ovplyní $\mathbf{w}$, teda deterministický model (výber komplikovanejšieho modelu).

Vzťah medzi zovšeobecnením a pretrénovaním sa niekedy popisuje ako bias – variancia tradeoff.

Ak by sme mali prístup k rozdeleniu, podľa ktorého tréningová vzorka vznikla, $p(x|t)$, mohli by sme vypočítať očakávanú hodnotu kvadratickej chyby medzi odhadnutými a skutočnými hodnotami.

Chceme, aby táto hodnota $\bar{\mathcal{M}}$ bola čo najmenšia. Môže byť dekomponovaná do dvoch výrazov **bias**, $\mathcal{B}$, a **variancia**, $\mathcal{V}$:

$$\bar{\mathcal{M}} = \mathcal{B}^2 + \mathcal{V}$$

Bias opisuje systematické chyby medzi naším modelom a generovanými dátami.

Ak je model príliš jednoduchý, bias je veľký (podtrénovanie). Ak vezmeme zložitejší model, bias vieme znížiť. Avšak zložitejšie modely majú vysokú varianciu.

Dôležité je nájsť správne vyváženie oboch.

6.6 Úlohy

1. Predpokladajme, že dataset N binárnych hodnôt $x_1, \ldots, x_N$ bol generovaný Bernoulliho rozdelením. Vypočítajte odhad maximálnej dôveryhodnosti pre Bernoulliho parameter.

   Bernoulliho rozdelenie: $P(X = x) = q^x(1 - x)^{1-x}$

2. Analyzujte skoky do diaľky u žien použitím šumu. Vygenerujte podobnú tréningovú vzorku.

Kapitola 7

Klasifikácia

7.1 Úvod

Dve najčastejšie objavujúce sa úlohy v strojovom učení sú klasifikácia a predikcia. Hlavný rozdiel spočíva v type cieľového atribútu, ktorého hodnotu predpovedáme. Klasifikácia je skupina metód strojového učenia, ktoré slúžia na predpovedanie hodnôt nominálnych atribútov (tried), napr. pridelenie alebo nepridelenie úveru.

Predikcia je skupina metód strojového učenia, ktoré slúžia na predpovedanie hodnôt numerických atribútov, napr. predikcia spotreby vody na určitom území.

Niektorí autori používajú namiesto pojmu predikcia pojem regresia a následne spoločným pojmom predikcia označujú klasifikáciu a regresiu. My budeme používať pôvodné označenie.

Klasifikácia – predpovedanie hodnôt nominálnych atribútov (tried).

1. fáza: vybudovať (naučiť sa) model správania dát na základe tréningovej množiny – obsahuje záznamy o objektoch, ktorých je hodnota cieľového atribútu už známa.
2. fáza: zostavený model sa použije na predikciu hodnoty cieľového atribútu u nových objektov (na základe hodnôt ostatných atributútov, ktoré sú o objekte k dispozícii).

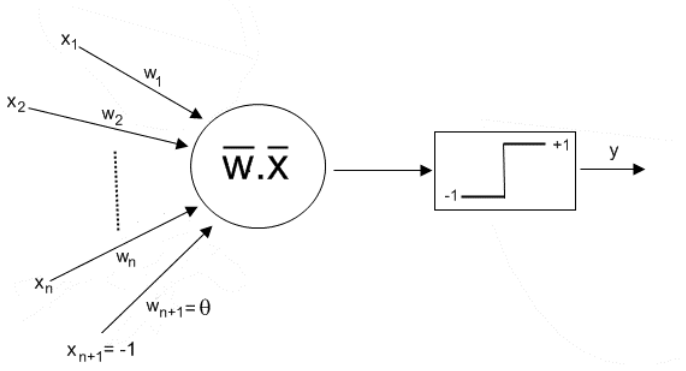
Príklad cieľových atribútov: pohlavie, áno/nie, typ školy a podobne.

Príklad: pridelenie úveru – zaradenie nových žiadateľov do kategórie prideliť/neprideliť na základe podobných údajov minulých žiadateľov o úver.

7.2 Klasifikácia do dvoch tried

Perceptrón

Perceptrón je základný prvok umelej neurónovej siete rovnako ako je neurón základným prvkom organických neurónových sietí. Existuje viac modelov perceptrónu, ktoré sa navzájom líšia svojimi funkciami a tiež tým, do akej miery sa blížia ku funkcii skutočného neurónu. V tejto kapitole sa budeme zaoberať modelom perceptrónu s bipolárnou binárnou aktivačnou funkciou, ktorý navrhol americký psychológ Frank Rosenblatt v roku 1958. Model je znázornený na obrázku 7.2. Na vstupe perceptrónu je vstupný vektor $\mathbf{x} = (x_1, x_2, \ldots, x_N)$. Nastaviteľnými parametrami sú váhy a prah perceprónu, t. j. vektor $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ a θ.



Obr. 7.1: Model perceptrónu

Funkciu f budeme nazývať *aktivačnou funkciou*.

$$f(x) = \begin{cases} 1 & \text{ak } x \geq 1, \\ -1 & \text{inak.} \end{cases} \tag{7.1}$$

Pre vstupný príklad $\mathbf{x}$ perceptrón vypočíta hodnotu

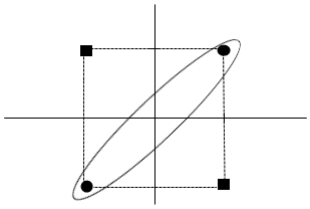
$$y = f(\mathbf{w} \cdot \mathbf{x} - \theta) = f(\sum_{n=1}^{N+1} w_n x_n) = f(\sum_{n=1}^{N} w_n x_n - \theta), \tag{7.2}$$

kde $w_{N+1} = -1, x_{N+1} = \theta$.

Jediné čo perceptrón dokáže, je rozdelenie priestoru vzorov na dva polpriestory (označíme si tieto polpriestory PL1 a PL2), ktoré sú oddelené lineárnou deliacou hranicou - nadrovinou. Deliaca hranica v N-rozmernom priestore je vyjadrená rovnicou (7.3)

$$w_1 x_1 + w_2 x_2 + \cdots + w_N x_N - \theta = 0 \tag{7.3}$$

Ak budeme uvažovať 2-rozmerný priestor, t. j. perceptrón s dvoma vstupmi, tak ten nedokáže riešť situáciu na obrázku 7.2, pretože neexistuje nadrovina (priamka), ktorá oddelí štvorčekov'e body od krúžkových.



Obr. 7.2: Usporiadanie bodov v rovine, ktoré perceptrón nedokáže oddeliť priamkou.

Riešime nasledujúci problém

Formulácia problému

Je daná tréningová vzorka

$$\overline{s} = ((\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \dots (\mathbf{x_M}, y_M)),$$

kde $\mathbf{x_i} = (x_{i,1}, x_{i,2}, \dots x_{i,N})$ a $y_i \in \{-1, +1\}$. Je potrebné nájsť nastavenie váh a prahu tak, aby klasifikácia do dvoch tried podľa perceptrónu bola konzistentná s tréningovou vzorkou.

Rossenblatt navrhol základný učiaci algoritmus pre učenie perceptrónu, avšak tu budeme hľadať riešenie iným spôsobom.

Pre p-tý príklad, $p = 1, \dots, M$ platí, ak je správne klasifikovaný

$$\mathbf{w} \cdot \mathbf{x_p} - \theta > 0, \quad \text{ak } y_p = +1$$

$$\mathbf{w} \cdot \mathbf{x_p} - \theta < 0, \quad \text{ak } y_p = -1$$

Uvedené nerovnosti je možné zapísať pomocou jednej, ak použijeme $-y_p$, teda

$$-y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta) < 0 \tag{7.4}$$

Výraz $\max\{0, -y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta\})$ nadobúda hodnotu 0, ak je príklad $\mathbf{x_p}$ správne klasifikovaný a nadobúda kladnú hodnotu, ak je klasifikovaný nesprávne.

Chceme, aby všetky príklady boli klasifikované správne, preto uvažujeme nasledujúcu funkciu

$$f_1(\mathbf{w}, \theta) = \sum_{p=1}^{N} \max\{0, -y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta\}) \tag{7.5}$$

a tiež chceme, aby chyba klasifikácie bola minimálna, preto riešime problém

$$\min_{\mathbf{w},\theta} \sum_{p=1}^{N} \max\{0, -y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta)\}, \tag{7.6}$$

ktorý nájde optimálne parametre pre separujúcu nadrovinu. Avšak sú tu dva technické problémy, s ktorými sa treba vysporiadať pri hľadaní riešenia:

- jedným z optimálnych riešení je $\theta = 0, \mathbf{w} = (0, 0, \ldots, 0)$; toto riešenie je potrebné ignorovať vo väšine prípadov;
- funkcia f_1 je spojitá, ale nie je všade diferencovateľná, preto sa tu nedá použiť metóda poklesu gradientu.

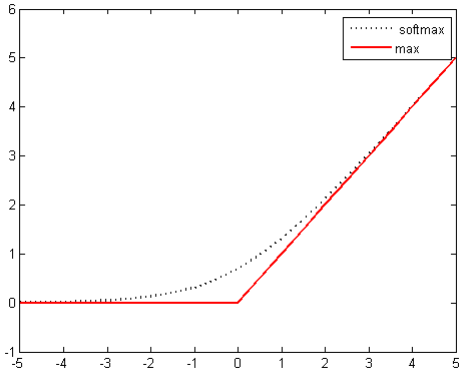
Logistická funkcia

Jedným z používaných riešení je nahradenie nediferencovateľnej funkcie max vo výraze (7.5), kde sa jedná o maximum z dvoch hodnôt, funkciou

$$softmax(e_1, e_2) = \log(e^{e_1} + e^{e_2}); \tag{7.7}$$

V našom prípade

$$softmax(e_1, e_2) = softmax(0, -y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta) = \log(1 + e^{-y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta)}) \tag{7.8}$$



Obr. 7.3: Porovnanie priebehov funkcií max a $softmax$.

Funkcia $softmax$ je dobrou aproximáciou funkcie max a je diferencovateľná. Takže funkciu f_1 nahradíme funkciou f_2 (7.9), ktorú budeme nazývať *softmax cenová funkcia*.

$$f_2(\mathbf{w}, \theta) = \sum_{p=1}^{N} \log(1 + e^{-y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta)}) \tag{7.9}$$

Táto funkcia už nemá vyšie uvedené dva problémy funkcie $f1$. Formálne sa náš problém redukoval na problém (7.10).

$$\min_{\mathbf{w}, \theta} \sum_{p=1}^{N} \log(1 + e^{-y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta)}) \tag{7.10}$$

Použijeme gradientovú metódu na hľadanie minima funkcie $f_2(\mathbf{w}, \theta)$. Zaveďme označenie $\tilde{\mathbf{x_p}} = \begin{bmatrix} \mathbf{x_p} \\ -1 \end{bmatrix}$ a $\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ \theta \end{bmatrix}$ Funkcia f_2 sa zmení na

$$f_2(\tilde{\mathbf{w}}) = \sum_{\mathbf{p=1}}^{\mathbf{N}} \log(\mathbf{1} + \mathbf{e}^{-\mathbf{y_p}(\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x_p}})}) \tag{7.11}$$

Deriváciou funkcie f_2 dostaneme vzťah (7.12).

$$\nabla f_2(\tilde{\mathbf{w}}) = \sum_{\mathbf{p=1}}^{\mathbf{N}} \sigma(-\mathbf{y_p}\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x_p}}) \left(\mathbf{1} - \sigma(-\mathbf{y_p}\tilde{\mathbf{w}} \cdot \tilde{\mathbf{x_p}})\right) \tilde{\mathbf{x_p}} \cdot \tilde{\mathbf{x_p}} \tag{7.12}$$

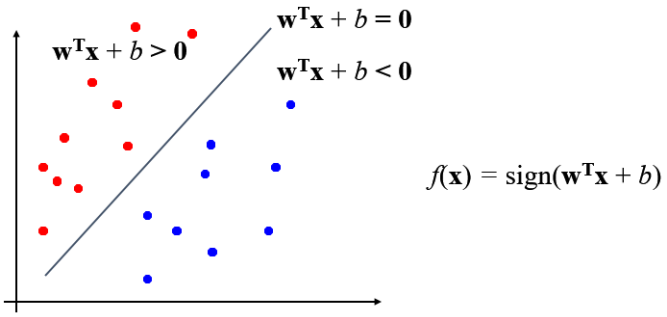
kde $\sigma(-t) = \frac{1}{1+e^t}$ je logistická sigmoidálna funkcia. Riešenie pre nastavenie váh získame, keď položíme $\nabla f_2(\tilde{\mathbf{w}}) = \mathbf{0}$.

7.3 Pásový perceptrón

Nadrovina, ktorá oddeľuje body dvoch tried, je daná rovnicou, ktorú zapíšeme vo vektorovom tvare $\mathbf{w} \cdot \mathbf{x} = 0$. Ak nechceme, aby v blízkosti nadroviny sa nachádzali nejaké body, môžeme stanoviť nasledujúce podmienky

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x_p} - \theta &\geq d, \text{ ak } y_p = +1 \\ \mathbf{w} \cdot \mathbf{x_p} - \theta &\leq d, \text{ ak } y_p = -1, \end{aligned} \tag{7.13}$$

kde d určuje nadrovinu posunutú do kladného alebo záporného polpriestoru.



Obr. 7.4: Lineárne separovateľné dáta je možné oddeliť nadrovinou.

Analogickým spôsobom ako pri obyčajnom perceptróne vieme určiť funkciu, ktorú budeme minimalizovať (7.14).

$$\min_{\mathbf{w},\theta} \sum_{p=1}^{M} \max\left(0, d - y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta)\right) \tag{7.14}$$

Aj v tomto prípade je potrebné riešiť to, že funkcia (7.14) je v aspoň jednom bode nediferencovateľná. Je tu tiež použiteľná funkcia $softmax$. Podobne je potom možné použiť gradientovú metódu, ktorá vedie k nastaveniu vhodných váh a prahu.

Presnosť naučeného klasifikátora

Zavedieme základnú funkciu, ktorá presne počíta počet bodov z tréningovej vzorky, ktoré boli klasifikované nekorektne.

$$f_0(\mathbf{w}, \theta) = \sum_{p=1}^{M} \max\left(0, sign(-y_p(\mathbf{w} \cdot \mathbf{x_p} - \theta))\right) \tag{7.15}$$

Funkcia je rastúca pri každom nesprávne klasifikovanom príklade. Funkcia môže byť použitá pri určovaní presnosti klasifikátora, ktorú je možné vyjadriť vzťahom

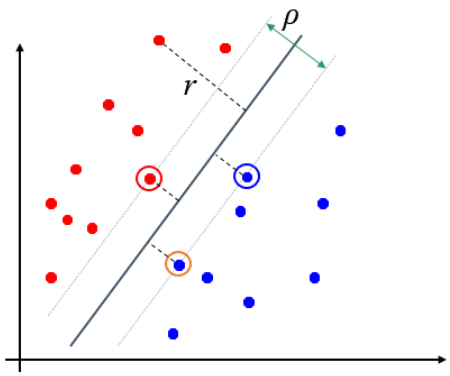
$$presnosť = 1 - \frac{f_0(\mathbf{w}^*, \theta^*)}{M}. \tag{7.16}$$

kde $\mathbf{w}^*, \theta^*$ predstavujú naučené hodnoty parametrov $\mathbf{w}$ a θ.

Táto metrika nadobúda hodnoty $\langle 0, 1\rangle$ a môže byť použiá aj na percentuálne vyjadrenie.

7.4 Klasifikácia s podpornými vektormi - SVM (Support Vector Machines)

Ideu pásového perceptróna budeme ďalej rozvíjať v tom smere, že budeme hľadať najväčšie možné d (najširší pás), ktoré bude triedy oddeľovať.



Obr. 7.5: Hranice pásu oddeľujúceho obe triedy je možné vypočítať pomocou vzdialenosti dvoch bodov normály na separujúcu priamku, $\frac{2}{\|\mathbf{w}\|}$.

Klasifikácia s maximálnym ρ

- Implikuje, že len podporné vektory sú dôležité; ostatné tréningové príklady sú ignorovateľné.

- Intuícia nám hovorí, že maximálne ρ je dobré a je to tiež v súlade s PAC teóriou.

Základný SVM problém

$$\begin{aligned} &minimalizovať_{b,\mathbf{w}} \quad \|\mathbf{w}\|^2 \\ &vzhľadom\ na \quad \max\left(0, 1 - y_p(b + \mathbf{x}_p^T\mathbf{w})\right) = 0, p = 1, \ldots N \end{aligned}$$

7.5 Nie celkom presná klasifikácia

Stará formulácia

- Nájsť $\mathbf{w}$ a b také, že
  $\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\|\mathbf{w}\|^T\|\mathbf{w}\|$ je minimálne
  a pre všetky $(\mathbf{x}_i, y_i), i = 1, \ldots n$ platí $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Modifikovaná formulácia obsahuje voľné premenné

- Nájsť $\mathbf{w}$ a b také, že
  $\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^T\|\mathbf{w}\| + C\sum_{i=1}^N \xi_i$ je minimálne
  a pre všetky $(\mathbf{x}_i, y_i), i = 1, \ldots n$ platí $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0$

Použitie nie celkom presných hraníc nám dáva voľný parameter C, ktorý musí byť fixovaný. Na parameter C sa môžeme dívať ako na prostriedok kontrolujúci "pretrénovanie".

C zohľadňuje maximálny rozsah separátora a tiež fitovanie trénovacích dát.

Podobnými úpravami ako v presnej klasifikácii dostaneme nasledujúci problém kvadratického programovania:

$$argmax_{\mathbf{w}} \quad \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i,j=1}^N \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j, \tag{7.17}$$

s podmienkami $\sum_{i=1}^N \alpha_i y_i = 0$ a $0 \leq \alpha_i \leq C$ pre všetky i.

Hodnoty α_i v tomto prípade majú ohraničenie zhora C.

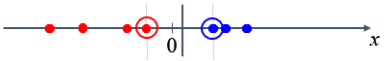
Vplyv každého tréningového bodu v našej rozhodovacej funkcii je úmerný α_i.

C tento vplyv ohraničí.

Ak C klesá, maximálny vplyv každého tréningového bodu je znížený, a tak viac z nich sa stáva aktívnymi v rozhodovacej funkcii. C by malo byť fixované.

7.6 Nelineárne SVM

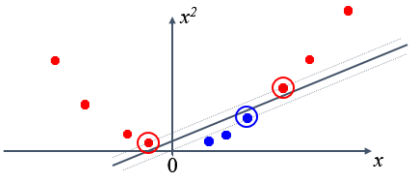
V nasledujúcich dvoch obrázkoch vidíme dve situácie rozloženia bodov na priamke. V prvom prípade sú separovateľné v druhom nie sú.



Obr. 7.6: Separovateľné body na priamke pomocou SVM.

Obr. 7.7: Neseparovateľné body na priamke pomocou SVM.

Keby sme tieto body pretransformovali pomocou kvadratickej funkcie do roviny, mohli by sme už lineárny separátor nájsť tak, ako to je na obrázku 7.6.



Obr. 7.8: Transformácia dát do roviny umožňuje nájsť lineárny separátor.

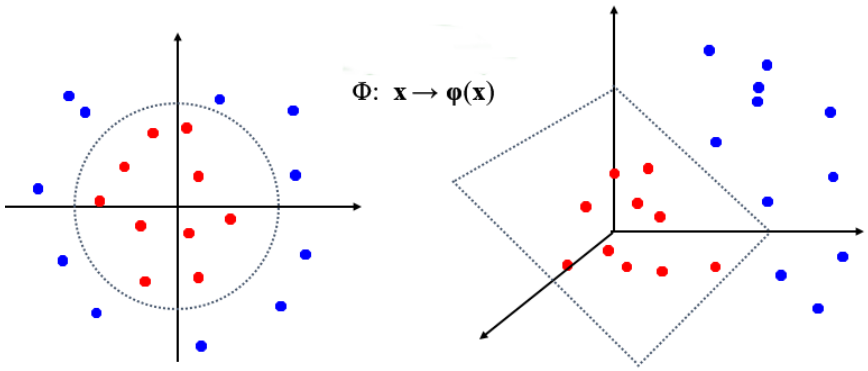
Namapujeme teda dáta do priestoru s vyššou dimenziou. V priestore s vyššou dimenziou je jednoduchšie nájsť oddeľujúcu nadrovinu. Napríklad, transformácia z roviny do 3-rozmerného priestoru je na obrázku 7.6.

Trik na použitie jadrovej funkcie pri transformácii dát do priestoru a vyššou dimenziou - Kernel Trick

- Lineárne separátory pracujú so skalárnym súčinom medzi vektormi $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Ak každý dátový bod je namapovaný do priestoru vyššej dimenzie pomocou transformácie $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$, dostávame skalárny súčin

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- **Jadrová funkcia** je funkcia, ktorá je ekvivalentná skalárnemu súčinu v základnom priestore.



Obr. 7.9: Transformácia z roviny do 3-rozmerného priestoru.

Príklad na použitie jadrovej funkcie:

- Uvažujme 2- dimenzionálny priestor, vektory sú v tvare $\mathbf{x} = [x_1, x_2]^T$; Nech jadrová fukcia má tvar $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
- $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$
  $= (1+x_{i1}x_{j1}+x_{i2}x_{j2})^2 = 1+x_{i1}^2x_{j1}^2+2x_{i1}x_{i2}x_{j1}x_{j2}+x_{i2}^2x_{j2}^2+2x_{i1}x_{j1}+2x_{i2}x_{j2}$
- Uvažujme transformáciu
  $\phi(\mathbf{x})^T = [1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]$
- Potrebujeme dokázať $K(\mathbf{x}_i\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$
- $\phi(x_i)^T\phi(x_j) =$
  $[1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^T * [1, x_{j1}^2, \sqrt{2}x_{j1}x_{j2}, x_{j2}^2, \sqrt{2}x_{j1},$
  $\sqrt{2}x_{j2}] = 1 + x_{i1}^2x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$
- Požadovaná rovnosť je dokázaná. Teda jadrová funkcia implicitne mapuje dáta do priestoru vyššej dimenzie.

K jadrovej funkcii $K(\mathbf{x}_i, \mathbf{x}_j)$ a dátam je možné vytvoriť maticu v tvare:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

Poznámka: Ak vytvorená matica je semi-pozitívne definitná, tak funkcia $K(\mathbf{x}_i, \mathbf{x}_j)$ je použiteľná ako jadrová funkcia (Mercerova veta).

Semi-pozitívne definitná symetrická matica zodpovedá semi-pozitívne definitnej symetrickej Gramovej matici.

Príklady jadrových (kernel) funkcií

- Lineárna jadrová funkcia

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- Polynomiálna jadrová funkcia

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$$

- Gaussova jadrová funkcia

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$$

- Sigmoidálna jadrová funkcia

$$K(\mathbf{x}_i, \mathbf{x}_j) = tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

Keď pracujeme s jadrovými funkciami máme optimalizačný problém v nasledujúcom tvare:

$$argmax_w \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{vzhľadom na} \quad \sum_{i=1}^{N} \alpha_i y_i = 0,\ 0 \leq \alpha_n \leq C, i = 1, \ldots N$$

$$y_{nov} = sign\left(\sum_{i=1}^{N} \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}_{nov}) + b \right)$$

7.7 Úlohy

1. Pre niektorú vyššie uvedenú jadrovú funkciu ukážte, že je jadrová.
2. Navrhnite množinu dát v dvojrozmernom priestore, ktorá nie je lineárne separovateľná, aplikujte na ňu použitie niektorej jadrovej funkcie.

Kapitola 8

Rozhodovacie stromy

Rozhodovacie stromy predstavujú jednu z často používaných metód na riešenie klasifikačných úloh. Ich výhody spočívajú v prehľadnosti a dobrej interpretácii. Pripomeňme si základné pojmy, ktoré využijeme pri popise rozhodovacích stromov.

Príklady (pozorovania, objekty) budeme označovať ako n-tice

$$\boldsymbol{x}_i \in \mathcal{R}^n, \quad i \in \{1, 2, \ldots, m\},$$

pričom n je prirodzené číslo.

Pre hodnoty cieľového atribútu y_i pre $i \in \{1, 2, \ldots, m\}$ platí, že

$$y_i \in C, \quad C = \{1, 2, \ldots, c\},$$

ak cieľový atribút môže nadobúdať c rôznych hodnôt.

Tréningová množina predstavuje množinu dvojíc

$$S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_k, y_k)\} \subseteq \mathcal{R}^n \times \{1, 2, \ldots, c\}, \quad k < m,$$

Testovacia množina má nasledujúci tvar

$$T = \{\boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_m\} \subseteq \mathcal{R}^n.$$

Klasifikátorom budeme nazývať funkciu K, ktorá každému príkladu z testovacej množiny priradí hodnotu, triedu, cieľového atribútu, formálne

$$K : T \to C.$$

Klasifikačná metóda rozhodovacích stromov je založená na rekurzívnom delení tréningovej množiny na dve alebo viac častí – vytvára sa strom. V každom kroku sa použije jedna vysvetľujúca premenná, teda vyberie

sa atribút, pomocou ktorého vetvíme rozhodovací strom. Deliaci bod sa určí tak, aby sa čo najviac zvýšila homogenita vzniknutných podmnožín príkladov (pomocou entropie, Giniho indexu alebo iné kritériá, napríklad korelačné koeficienty, či štatistické testy). Vytváranie stromu je ukončené, ak v koncových vrcholoch (listoch) stromu ostali len pozorovania z rovnakej triedy alebo je splnená nejaká obmedzujúca podmienka na minimálny počet pozorovaní v listovom vrchole, hĺbku stromu a podobne. Každému listovému vrcholu (ktorý vlastne zodpovedá zreťazeniu logických podmienok) sa priradí trieda, ktorá sa najčastejšie vyskytuje v pozorovaniach tréningovej množiny zodpovedajúcich danému listovému vrcholu.

Niektoré z kritérií pre výber vysvetľujúcich atribútov na delenie rozhodovacieho stromu si predstavíme v nasledujúcej podkapitole.

8.1 Miery výberu atribútov

Miera výberu atribútov slúži na určenie deliaceho kritéria, ktoré oddelí danú množinu príkladov do tried. Ak sme rozdelili množinu príkladov na základe istého deliaceho kritéria, každá podmnožina by mala byť v ideálnom prípade homogénna, teda obsahuje iba príklady jednej triedy. Najlepšie deliace kritérium je teda také, ktoré nám zabezpečí takýto výsledok.

Miery výberu atribútov sa zvyknú nazývať aj **deliace pravidlá**, pretože určujú, akým spôsobom sú príklady v danom uzle rozhodovacieho stromu rozdelené. Tieto miery poskytujú zoradenie atribútov vzhľadom na tzv. kvalitu rozdelenia uvažovanej množiny príkladov do tried. Atribút, ktorý dosiahne najvyššie skóre, je vybraný za deliaci atribút. Ak deliaci atribút obsahuje spojité hodnoty a uvažujeme binárne stromy, je potrebné určiť ako deliace kritérium tzv. deliaci bod, resp. deliacu podmnožinu.

Ak je k uzlu stromu pre množinu príkladov S priradené deliace kritérium, hrany vychádzajúce z tohto uzla zodpovedajú jednotlivým výstupom tohto kritéria.

Entropia

Entropia je miera homogenity v skupine príkladov, miera pomocou ktorej vetvíme rozhodovací strom. Hodnoty nadobúda z jednotkového intervalu. Čím vyššia hodnota, tým je homogenita nižšia.

Nech $x_i(A) = a$ vyjadruje hodnotu a atribútu A, ktorá prislúcha príkladu x_i pre $i \in \{1, 2, \ldots, k\}$, pričom k je prirodzené číslo, ktoré vyjadruje počet príkladov v tréningovej množine.

Entropiu v jednej hrane uzla rozhodovacieho stromu môžeme vypočítať nasledovne:

$$H\Big(x_i(A) = a\Big) = -\sum_{y_j \in \{1,\dots,c\}} \mathrm{p}(y_j) \cdot \log_2(\mathrm{p}(y_j)),$$

pričom $\mathrm{p}(y_j)$ je pravdepodobnosť, že príklad patriaci hrane $x_i(A) = a$ bude klasifikovaný do triedy y_j.

Celková entropia v uzle má tvar

$$H(A) = \sum_{a \in A} \mathrm{p}\Big(x_i(A) = a\Big) \cdot H\Big(x_i(A) = a\Big).$$

Informačný zisk

Informačný zisk je definovaný ako rozdiel medzi entropiou v jednej hrane uzla a celkovou entropiou v uzle. Informačný zisk vyjadrujeme

$$I\Big(x_i(A) = a\Big) = H\Big(x_i(A) = a\Big) - H(A),$$

pričom $H\Big(x_i(A) = a\Big)$ je entropia v jednej hrane uzle a $H(A)$ je celková entropia v uzle.

Informačný zisk sa pri výbere atribútu snažíme maximalizovať. Táto miera výberu atribútov má nevýhodu, že väčšinou uprednostní také atribúty, ktoré majú viac vetiev. Uvedený nedostatok je možné odstrániť normalizovaním informačného zisku, zavedením tzv. **pomerového informačného zisku** pomocou tzv. pomerovej entropie.

Pomerovú entropiu definujeme

$$H_P(A) = -\sum_{a \in A} \mathrm{p}\Big(x_i(A) = a\Big) \cdot \log_2\left(\mathrm{p}\Big(x_i(A) = a\Big)\right),$$

pričom $\mathrm{p}\Big(x_i(A) = a\Big)$ je pravdepodobnosť, že príklad x_i nadobúda v atribúte A hodnotu a.

Následne môžeme pomocou pomerovej entropie definovať pomerový informačný zisk nasledujúcim spôsobom.

Pomerový informačný zisk

Pomerový informačný zisk môžeme zapísať v tvare

$$I_P\Big(x_i(A) = a\Big) = \frac{I\Big(x_i(A) = a\Big)}{H_P(A)},$$

pričom $I\Big(x_i(A) = a\Big)$ je informačný zisk.

Giniho index

Giniho index je možné použiť pre kategoriálne atribúty, resp. numerické atribúty po ich diskretizácii. Nižšie hodnoty Giniho indexu indikujú lepšiu schopnosť atribútu rozdeliť príklady do tried.

Giniho index v jednej hrane uzla môžeme vyjadriť v tvare

$$G\Big(x_i(A) = a\Big) = 1 - \sum_{y_j \in \{1,\ldots,c\}} \mathrm{p}(y_j) * p(y_j),$$

pričom $\mathrm{p}(y_j)$ je pravdepodobnosť, že príklad patriaci hrane $x_i(A) = a$ bude klasifikovaný do triedy y_j.

Celkový Giniho index v uzle vypočítame

$$G(A) = \sum_{a \in A} \mathrm{p}\Big(x_i(A) = a\Big) \cdot G\Big(x_i(A) = a\Big).$$

V tejto podkapitole sme predstavili niekoľko mier na výber atribútov pre delenie rozhodovacieho stromu. Bolo navrhnutých viacero ďalších mier založených napríklad na chí-kvadrát testovaní nezávislosti. Niektoré postupy používajú aj viacatribútové delenie, pri ktorom je delenie stromu založené na kombinácii atribútov. Napriek viacerým štúdiám nebolo preukázané, aby niektorá z týchto mier bola nadradená nad všetkými ostatnými. Z tohto dôvodu závisí výber vhodnej miery od použitej aplikačnej domény, typu atribútov a ďalších faktorov.

8.2 Algoritmus pre generovanie rozhodovacích stromov

Ak už poznáme spôsob, akým ohodnotiť kvalitu rozdelenia rozhodovacieho stromu v jednotlivých uzloch, môžeme definovať všeobecný algoritmus na generovanie rozhodovacieho stromu:

Algoritmus na generovanie rozhodovacieho stromu

Algoritmus 8.1 *Konštrukcia rozhodovacieho stromu*

KonštruujRS(*tréningová množina S, minPodiel* $\in [0,1]$)

ak minPodiel objektov z S patrí do triedy $y_j \in C$

potom vytvor listový uzol zaradzujúci objekty do y_j;

inak

pre každý atribút A

pre každé rozdelenie hodnôt $a \in A$

ohodnoť kvalitu rozdelenia;

vykonaj najlepšie rozdelenie;

vzniknú množiny $S_1, S_2, \ldots, S_p$*;*

KonštruujRS$(S_1,$ *minPodiel);*

KonštruujRS$(S_2,$ *minPodiel);*

$\vdots$

KonštruujRS$(S_p,$ *minPodiel).*

Ak sa pozrieme na konkrétne implementácie tohto všeobecného algoritmu zistíme, že algoritmus **ID3**, ktorého autorom je Quinlan, využíva **entropiu** a od nej odvodený **informačný zisk** ako kritérium delenia rozhodovacieho stromu. Quinlan navrhol aj algoritmus C4.5, ktorý je vylepšeným a doplneným algoritmom ID3, využíva **entropiu** a od nej odvodený **pomerový informačný zisk**. Algoritmus **CART** využíva **Giniho index**.

8.3 Preučenie rozhodovacieho stromu a jeho prerezávanie

Preučenie (tzv. overfitting) rozhodovacieho stromu znamená, že rozhodovací strom klasifikuje správne príklady v tréningovej údajovej sade, ale pri klasifikácii príkladov z testovacej údajovej sady poskytuje nesprávne výsledky.

Prerezávanie rozhodovacieho stromu

Po vytvorení rozhodovacieho stromu môžu byť niektoré vetvy ovplyvnené anomáliami v tréningovej sade údajov z dôvodu extrémnych hodnôt alebo chybných údajov. **Metódy prerezávania rozhodovacích stromov** slúžia na riešenie takýchto situácií. Zvyčajne sa používajú štatistické metódy na odstránenie tých najmenej spoľahlivých vetiev stromu. Orezané stromy majú tendenciu byť jednoduchšie, a teda aj ľahšie interpretovateľné.

Existujú dva základné prístupy k prerezávaniu rozhodovacích stromov:

- prerezávanie počas generovania rozhodovacieho stromu (tzv. prepruning),
- prerezávanie po vygenerovaní rozhodovacieho stromu (tzv. postpruning),

Pri prvom prístupe sa rozhodovací strom prerezáva predčasným zastavením jeho konštrukcie (napríklad rozhodnutím nedeliť ďalej podmnožinu tréningových n-tíc v danom uzle. Po zastavení sa uzol stane listom.

Príkladom v liste priradíme triedu, ktorá zodpovedá najčastejšej hodnote medzi podmnožinou n-tíc alebo na základe rozdelenia pravdepodobnosti týchto n-tíc do jednotlivých tried.

Na posúdenie správnosti rozdelenia v uzle používame pri konštrukcii rozhodovacieho stromu miery výberu atribútov, akými sú napríklad informačný zisk, Giniho index, štatistická významnosť a iné miery. Ak pri rozdeľovaní príkladov v uzle by vzniklo rozdelenie, ktorého kvalita klesne pod vopred stanovenú hranicu, rozdelenie danej podmnožiny zastavíme. Niekedy je však náročné určiť primeranú hranicu, teda vhodný prah pre zastavenie delenia. Vysoké prahy môžu mať za následok príliš zjednodušené stromy, zatiaľ čo nízke prahové hodnoty by mohli viesť k veľmi malému zjednodušeniu.

Druhým a bežnejším prístupom je prerezávanie po vygenerovaní rozhodovacieho stromu, pri ktorom sa odstraňujú podstromy z úplne vytvoreného rozhodovacieho stromu. Podstrom v danom uzle sa orezáva odstránením jeho vetiev a jeho nahradením listom. Príkladom v liste je priradená trieda, ktorá je najčastejšia v príslušnom podstrome.

Bagging a boosting

Na zvyšovanie presnosti klasifikácie s použitím rozhodovacích stromov sa používa aj metóda tzv. **baggingu**, pri ktorej sa postupne vytvára určitý počet rozhodovacích stromov (odporúča sa veľký počet), pričom ich vytváranie je nezávislé (môže byť realizované paralelne). Každý rozhodovací strom sa tak vytvára nad inou náhodnou vzorkou príkladov z tréningovej množiny. Náhoda sa zapája aj do výberu vysvetľujúcej premennej pri každom delení, pričom sa nevyberá zo všetkých premenných, ale len z menšej náhodne vybranej podmnožiny premenných. Výsledná klasifikácia sa získa takým spôsobom, že sa vytvoria všetky rozhodovacie stromy, pričom trieda pre konkrétny príklad sa určí ako modus zo všetkých tried, ktoré boli pridelené danému príkladu v rámci všetkých vygenerovaných rozhodovacích stromov. V takomto prípade hovoríme o tzv. **náhodných lesoch**. Je možné matematicky ukázať, že zvyšovaním počtu rozhodovacích stromov nedochádza k preučeniu, teda je vhodné vytvoriť čo najviac rozhodovacích stromov, pokiaľ to dovoľuje výpočtový výkon. Praktické skúsenosti z mnohých oblastí ukazujú, že náhodné lesy dokázali mnohokrát výrazným spôsobom zvýšiť presnosť klasifikácie v porovnaní s jedným rozhodovacím stromom.

Ďalším spôsobom zvyšovania presnosti klasifikácie pri použití rozhodovacích stromov je metóda tzv. **boostingu**. Na rozdiel od náhodných lesov pri boostingu nevytvárame jednotlivé rozhodovacie stromy nezávisle, ale postupne ich budujeme tak, aby každý ďalší rozhodovací strom čo najviac prispel k znižovaniu chyby klasifikácie pomocou doteraz vytvorených rozhodovacích stromov. Pri veľkom počte rozhodovacích stromov alebo pri nevhodnej veľkosti rozhodovacích stromov môže dôjsť k preučeniu, preto je pri tejto metóde náročnejšia inicializácia vstupných hyperparametrov.

Kapitola 9

Zhlukovanie

Zhlukovanie (zhluková analýza, klastering) je metóda, ktorej cieľom je zoskupovanie množiny navzájom podobných objektov do tried, tzv. zhlukov.

Zhluk je množina navzájom podobných objektov, ktoré sa od objektov nepatriacich do zhluku odlišujú. Podobnosť objektov popisujeme na základe hodnôt ich atribútov vzhľadom na použité metrické miery (tzv. metriky).

Aj klasifikácia je metódou, ktorá rozdeľuje množinu objektov do tried alebo skupín, ale pri zhlukovaní nie sú cieľové triedy vopred dané, označenie cieľových tried vopred nepoznáme. Klasifikácia vyžaduje poznať označenia cieľových tried pre objekty v tréningovej množine, na základe čoho sa vybuduje klasifikátor na modelovanie každej skupiny. Zhlukovanie predstavuje v podstate opačný proces. Najprv rozdelíme množinu objektov do skupín na základe ich podobnosti a následne priradíme označenie pre relatívne malý počet zhlukov, ktoré takýmto spôsobom vzniknú.

Zhlukovanie má dôležité postavenie v strojovom učení, rozpoznávaní vzorov, analýze obrazu, dolovaní údajov, bioinformatike a iných odvetviach.

Jednotlivé algoritmy zhlukovania sa líšia spôsobom hľadania zhlukov, efektivitou vyhľadávania a obsahom jednotlivých zhlukov. Môžeme uvažovať:

- **klasický prístup**: každý objekt patrí do určitého zhluku, alebo do neho nepatrí,
- **fuzzy prístup**: každý objekt patrí do každého zhluku s určitým stupňom príslušnosti.

Vo všeobecnosti, rozlišujeme niekoľko typov techník zhlukovania:

Typy zhlukovania

- segmentačné metódy,
- hierarchické metódy,
- metódy založené na hustote,

- metódy založené na mriežke,
- metódy založené na modeloch.

V nasledujúcich častiach si predstavíme niektoré príklady týchto metód.

9.1 Typy údajov pri zhlukovaní

Predpokladajme, že množina údajov obsahuje n objektov, ktoré môžu reprezentovať osoby, krajiny, dokumenty a iné príklady. Pri zhlukovaní najčastejšie používame údajové štruktúry **matice údajov** (štruktúra typu objekt-atribút) a **matice podobnosti, resp. odlišnosti** (štruktúra typu objekt-objekt).

Matica údajov

Matica údajov X je matica v tvare $n \times p$, pomocou ktorej je možné reprezentovať hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$:

$$\begin{pmatrix} x_{11} & \ldots & x_{1j} & \ldots & x_{1p} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{i1} & \ldots & x_{ij} & \ldots & x_{ip} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{n1} & \ldots & x_{nj} & \ldots & x_{np} \end{pmatrix} \tag{9.1}$$

Matica podobnosti

Matica podobnosti, resp. odlišnosti reprezentuje vzťahy medzi dvojicami objektov pre všetkých n objektov. Štruktúra je vo forme matice typu $n \times n$, pričom hodnota $d_{i,l}$ reprezentuje zistenú mieru podobnosti, resp. odlišnosti medzi objektmi i a j pre $i \in \{1, 2, \ldots, n\}$ a $l \in \{1, 2, \ldots, n\}$:

$$\begin{pmatrix} 1 & & & & \\ \mathrm{d}(2,1) & 1 & & & \\ \mathrm{d}(3,1) & \mathrm{d}(3,2) & 1 & & \\ \vdots & \vdots & \vdots & & \\ \mathrm{d}(n,1) & \mathrm{d}(n,2) & \ldots & \ldots & 1 \end{pmatrix} \tag{9.2}$$

V matici podobnosti alebo matici odlišnosti platí $d(i, l) = d(l, i)$ pre $i \in \{1, 2, \ldots, n\}$ a $l \in \{1, 2, \ldots, n\}$. Z matice údajov je možné vytvoriť maticu podobnosti, resp. maticu odlišnosti podľa podmienok použitého algoritmu zhlukovania.

V prípade intervalových premenných (napríklad výška osoby, hmotnosť osoby, teplota) používame **štandardizáciu hodnôt atribútu**.

Pri štandardizácii transformujeme hodnoty daného atribútu na hodnoty bez jednotiek (bez ohľadu na to, či meriame v metroch alebo palcoch). Napríklad pre atribút a_j, $j \in \{1, 2, \ldots, p\}$ je postup nasledujúci:

1. Nech $x_{1j}, x_{2j}, \ldots, x_{nj}$ sú hodnoty všetkých objektov v atribúte a_j pre $j \in \{1, 2, \ldots, p\}$. Nech m_j je aritmetický priemer hodnôt v atribúte a_j pre $j \in \{1, 2, \ldots, p\}$. Potom priemernú absolútnu odchýlku s_{a_j} vypočítame:

$$s_{a_j} = \frac{1}{n}(|x_{1j} - m_j| + |x_{2j} - m_j| + \ldots + |x_{nj} - m_j|).$$

2. Z-skóre (štandardizované skóre) vypočítame ako:

$$z_{ij} = \frac{x_{ij} - m_j}{s_j}$$

pre $i \in \{1, 2, \ldots,$n$\}$ a $j \in \{1, 2, \ldots, p\}$. Z-skóre môže byť užitočne použité na detekciu anomálií v množine údajov.

Po štandardizácii údajov alebo v prípade možnosti aj bez nej nasleduje vytvorenie matice odlišnosti, resp. matice podobnosti.

Odlišnosť, resp. podobnosť medzi objektmi, ktoré sú vyjadrené intervalovými premennými definujeme ako vzdialenosť medzi objektmi. Používa sa napríklad **Euklidovská vzdialenosť**

$$d(i, l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \ldots + (x_{ip} - x_{lp}^2)},$$

pričom x_{ij} a x_{lj} predstavuje hodnoty objektov o_i a o_l v atribúte a_j pre $i \in \{1, 2, \ldots, n\}$, $l \in \{1, 2, \ldots, n\}$ a $j \in \{1, 2, \ldots, p\}$.

Vo všeobecnosti, každému atribútu môžeme priradiť váhu.

Nech $\{w_1, w_2, \ldots, w_p\}$ sú váhy jednotlivých atribútov pre $j \in \{1, 2, \ldots, p\}$. V takom prípade môžeme vyjadriť **váženú Euklidovskú vzdialenosť** nasledujúcim spôsobom:

$$d(i, k) = \sqrt{w_1(x_{i1} - x_{l1})^2 + w_2(x_{i2} - x_{l2})^2 + \ldots + w_p(x_{ip} - x_{lp}^2)},$$

pričom x_{ij} a x_{lj} predstavuje hodnoty objektov o_i a o_l v atribúte a_j pre $i \in \{1, 2, \ldots, n\}$, $l \in \{1, 2, \ldots, n\}$ a $j \in \{1, 2, \ldots, p\}$.

Iným príkladom vzdialenostnej metriky je **Manhattanská vzdialenosť**, ktorá je definovaná nasledujúcim spôsobom:

$$d(i, j) = |x_{i1} - x_{l1}| + |x_{i2} - x_{l2}| + \ldots + |x_{ip} - x_{lp}|,$$

pričom x_{ij} a x_{lj} predstavuje hodnoty objektov o_i a o_l v atribúte a_j pre $i \in \{1, 2, \ldots, n\}$, $l \in \{1, 2, \ldots, n\}$ a $j \in \{1, 2, \ldots, p\}$.

Minkowského vzdialenosť je zovšeobecnením Euklidovej vzdialenosti alebo Manhattanskej vzdialenosti. Je definovaná takýmto spôsobom:

$$d(i,j) = \left(|x_{i1} - x_{l1}|^m + |x_{i2} - x_{l2}|^m + \ldots + |x_{ip} - x_{lp}|^m \right)^{\frac{1}{m}},$$

pričom m je prirodzené číslo väčšie ako nula, x_{ij} a x_{kj} predstavuje hodnoty objektov o_i a o_k v atribúte a_j pre $i \in \{1, 2, \ldots, n\}$, $k \in \{1, 2, \ldots, n\}$ a $j \in \{1, 2, \ldots, p\}$. Takáto vzdialenosť sa označuje aj ako L_m norma, L_1 predstavuje Manhattanskú vzdialenosť, L_2 predstavuje Euklidovskú vzdialenosť.

V prípade binárnych atribútov môžeme použiť tabuľku početností, v ktorej budeme pre každú dvojicu objektov reprezentovať, v koľkých prípadoch sa hodnoty ich atribútov zhodujú. Podobný princíp môžeme zvoliť aj pri kategoriálnych premenných s rôznym počtom hodnôt. Pri ordinálnych premenných môžeme navyše reprezentovať poradie hodnôt atribútov a maticu podobnosti generovať na základe zachovávania, resp. nezachovávania poradia hodnôt medzi dvojicou objektov.

9.2 Segmentačné metódy zhlukovania

Tieto metódy zhlukovania sú založené na vytvorení vopred známeho počtu zhlukov, pričom každý zhluk musí obsahovať aspoň jeden objekt a každý objekt musí patriť práve do jedného zhluku. V niektorých fuzzy algoritmoch môže byť druhá podmienka zjemnená. Tieto metódy zhlukovania sú založené na iteratívnom prístupe, pričom našim cieľom je zlepšiť zhlukovanie postupným presúvaním objektov medzi zhlukmi.

Najznámejšími a najpoužívanejšími segmentačnými metódami zhlukovania je **k-means zhlukovanie, zhlukovanie k-medoidov** a ich rôzne varianty.

Nech X je matica údajov reprezentujúca hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$, napríklad x_{ij} vyjadruje hodnotu objektu o_i v atribúte a_j pre $i \in \{1, 2, \ldots, n\}$ a $j \in \{1, 2, \ldots, p\}$. Nech k je počet zhlukov ($k \leq n$).

K-means zhlukovanie

Metóda **k-means zhlukovania** má nasledujúci postup:

1. Náhodne vyberieme k objektov, každý z nich reprezentuje tzv. **centrum zhluku**, ktoré označíme $C_1, C_2, \ldots, C_k$.

2. Každý objekt priradíme do zhluku, ktorý mu je najpodobnejší vzhľadom na vzdialenosť medzi objektom a centrom zhluku.

3. Ak je každý objekt priradený k nejakému zhluku, vypočítame nové centrá zhlukov tak, že vzdialenosť medzi všetkými objektmi zhluku a novým centrom zhluku bude minimalizovaná, spriemerovaná. Napríklad ak budeme uvažovať v 2-rozmernom priestore zhluk s dvoma objektmi o_1, o_2, tak centrum zhluku C_1 bude nadobúdať hodnoty

$$x_{C_1 1} = \frac{x_{11} + x_{21}}{2} \qquad \text{a} \qquad x_{C_1 2} = \frac{x_{12} + x_{22}}{2}.$$

4. Ak po prepočítaní centier zhlukov nastala zmena aspoň v jednom centre zhluku, pokračujeme krokom 2.

Na určenie toho, či vytvorené zhluky sú kompaktné a kvalitne oddelené, môžeme použiť **kritérium štvorcovej chyby:**

$$E = \sum_{i=1}^{k} \left(\sum_{o_l \in C_i} |x_{l1} - x_{\mathrm{C_i}1}|^2 + |x_{l2} - x_{\mathrm{C_i}2}|^2 + \ldots + |x_{lp} - x_{\mathrm{C_i}p}|^2 \right),$$

pričom l označuje indexy objektov patriacich do zhluku C_i pre $i \in \{1, 2, \ldots, k\}$. Čím je hodnota kritéria štvorcovej chyby menšia, tým sú zhluky kompaktnejšie a kvalitnejšie oddelené.

Algoritmus pre k-means zhlukovanie môžeme vo všeobecnosti zapísať nasledovne:

Algoritmus 9.1 *K-means zhlukovanie*

Nech X je matica údajov, ktorá reprezentuje hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$. Nech k je počet zhlukov, $k \leq n$.

begin

náhodne vyber k objektov z množiny objektov $\{o_1, o_2, \ldots, o_n\}$;

vybrané objekty označ za centrá zhlukov C_i pre $i \in \{1, 2, \ldots, k\}$;

repeat

pre každý objekt $\{o_1, o_2, \ldots, o_n\}$ nájdi najbližšie centrum zhluku C_i pre $i \in \{1, 2, \ldots, k\}$;

pre každé centrum zhluku C_i, $i \in \{1, 2, \ldots, k\}$

aktualizuj $x_{C_i 1}, \ldots, x_{C_i p}$ tak, aby vzdialenosť od objektov zhluku bola minimálna;

until *centrá zhlukov C_i, $i \in \{1, 2, \ldots, k\}$ sú po aktualizácii nezmenené;*

end

Metóda k-means zhlukovania môže byť použitá len v prípade, že je definované centrum zhluku pomocou priemerných hodnôt. To nie je možné v prípade kategoriálnych premenných. Potreba vopred určiť počet zhlukov môže byť tiež nevýhodou. Existuje niekoľko ďalších variánt tejto metódy, ktoré sa líšia v spôsobe výberu počiatočných zhlukov, výpočte podobnosti medzi objektmi, a v spôsobe aktualizácie centier zhlukov.

Metóda k-means je citlivá na extrémne hodnoty, keďže používame aritmetické priemery a kritérium štvorcovej chyby. Tieto nevýhody môžu byť odstránené použitím metódy zhlukovania k-medoidov, ktorú si predstavíme v nasledujúcej časti.

Metóda k-medoidov zhlukovania používa namiesto výpočtu priemerných hodnôt pri aktualizácii centier zhlukov ako centrá zhlukov samotné objekty z matice údajov. Konkrétne za centrum zhluku vyberieme objekt, ktorý je k stredu zhluku najbližšie. Rozdelenie objektov do príslušných zhlukov je založené na princípe minimalizácie súčtu odlišností medzi objektmi a centrami zhlukov.

Kritérium absolútnej chyby pri tejto metóde je definované

$$E = \sum_{i=1}^{k} \left(\sum_{o_l \in C_i:} |x_{l1} - x_{\mathrm{C_i}1}| + |x_{l2} - x_{\mathrm{C_i}2}| + \ldots + |x_{lp} - x_{\mathrm{C_i}p}| \right),$$

pričom l označuje indexy objektov patriacich do zhluku C_i pre $i \in \{1, 2, \ldots, k\}$.

Kritérium absolútnej chyby môže byť použité na vyjadrenie tzv. **nákladovej funkcie**, ktorá určuje hodnotu, o ktorú sa zmenila absolútna chyba po nahradení aktuálneho centra zhluku iným objektom. Celková nákladová funkcia nahradenia predstavuje sumu nákladov za všetky objekty, ktoré nie sú priradené k centrám zhlukov. Ak sú celkové náklady záporné (absolútna chyba klesla), tak aktuálne centrum zhluku je nahradené príslušným objektom. Ak sú celkové náklady pozitívne, centrum zhluku sa nemení.

Algoritmus zhlukovania k-medoidov môžeme vo všeobecnosti zapísať nasledovne:

Algoritmus 9.2 *Zhlukovanie k-medoidov*

Nech X je matica údajov, ktorá reprezentuje hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$. Nech k je počet zhlukov, $k \leq n$.

begin

náhodne vyber k objektov z množiny objektov $\{o_1, o_2, \ldots, o_n\}$;

vybrané objekty označ za centrá zhlukov C_i pre $i \in \{1, 2, \ldots, k\}$;

repeat

pre každý objekt $\{o_1, o_2, \ldots, o_n\}$ nájdi najbližšie centrum zhluku C_i pre $i \in \{1, 2, \ldots, k\}$;

vypočítaj chybu E;

pre každé centrum zhluku C_i, $i \in \{1, 2, \ldots, k\}$

náhodne vyber objekt o_l z množiny $\{o_1, o_2, \ldots, o_n\}$;

vypočítaj zmenu chyby, ak aktuálne centrum zhluku C_i nahradíme o_l;

ak chyba klesla, nahraď centrum zhluku C_i objektom o_l;

until *centrá zhlukov C_i, $i \in \{1, 2, \ldots, k\}$ sú po aktualizácii nezmenené;*

end

Metóda zhlukovania k-medoidov je robustnejšia ako metóda k-means zhlukovania s ohľadom na zašumené a anomálne údaje, pretože reperezentácia pomocou medoidov (reprezentujúcich objektov) je menej ovplyvnená extrémnymi hodnotami ako metóda k-means s centrami počítanými pomocou priemerných hodnôt. Spracovanie metódou k-medoidov je však výpočtovo náročnejšie. Obe metódy vyžadujú, aby používateľ určil počet zhlukov.

V prípade veľkého množstva objektov môžeme pre výber vhodných medoidov vybrať namiesto všetkých údajov len ich časť. Na takejto myšlienke sú založené napríklad algoritmy CLARANS (zhlukovanie vo veľkých aplikáciách založené na náhodnom vyhľadávaní).

Okrem použitia priemeru alebo medoidu ako miery stredu zhluku, iné varianty segmentačných metód využívajú pri rozdelení objektov do zhlukov medián alebo modus. Výsledkom pri takomto postupe sú **metódy zhlukovania k-mediánov alebo metóda k-modusov**.

9.3 Hierarchické metódy zhlukovania

Hierarchické metódy zhlukovania sú založené na zoskupovaní objektov do stromu zhlukov. Hierarchická dekompozícia môže byť formovaná buď zdola nahol (aglomeratívne hierarchické zhlukovanie) alebo zhora nadol (rozkladajúce hierarchické zhlukovanie). Integrácia hierarchických a segmentačných metód zhlukovania sa ukázala ako vhodné riešenie pri mnohých aplikáciách.

Aglomeratívne hierarchické zhlukovanie je založené na spájaní, pričom v nultej iterácii je každému objektu priradený vlastný zhluk. Tieto jednoprvkové zhluky sú následne spájané do stále väčších zhlukov, až kým nie je splnená podmienka ukončenia, napríklad počet zhlukov (v teoretickom prípade môže nastať situáciu, že po ukončení zhlukovania vznikne jeden zhluk, ktorý bude obsahovať všetky objekty). Väčšina metód hierarchického zhlukovania patrí do tejto skupiny, pričom sa líšia len v definícii a spôsobe určovania podobnosti zhlukov.

Rozkladajúce hierarchické zhlukovanie je založené na rozdeľovaní, pričom v nultej iterácii patria všetky objekty jednému zhluku. Tento zhluk je ďalej rozdeľovaný do stále menších zhlukov, až kým nie je splnená podmienka ukončenia (v teoretickom prípade môže nastať aj taká situácia, že po ukončení zhlukovania vzniknú jednoprvkové zhluky). Väčšina metód hierarchického zhlukovania sú rozkladajúce, líšia sa definíciou a spôsobom určovania podobnosti zhlukov.

Dendrogram je stromová štruktúra, ktorú môžeme použiť na znázornenie hierarchického zhlukovania. Ilustruje, akým spôsobom sú objekty v jednotlivých iteráciách spájané, resp. rozdeľované.

Nech C_i a C_j sú zhluky pre $i \in \{1, 2, \ldots, k\}$, $j \in \{1, 2, \ldots, k\}$, $i \neq j$ a nech $o \in C_i$ a $o' \in C_j$. Nech d vyjadruje funkciu vzdialenosti medzi dvoma objektmi a funkcia d^* vyjadruje funkciu vzdialenosti medzi dvoma zhlukmi.

Pri spájaní, resp. rozdeľovaní zhlukov sa najčastejšie používajú nasledujúce miery na určovanie vzdialenosti medzi zhlukmi:

1. **Minimálna vzdialenosť** – vzdialenosť medzi dvoma objektmi z dvoch rôznych zhlukov, ktorá je najmenšia:

$$d^*_{\min}(C_i, C_j) = \min\{d(o, o') : o \in C_i, o' \in C_j\}.$$

   V tomto prípade hovoríme o zhlukovacom algoritme **najbližších susedov**. Ak je navyše daná podmienka ukončenia zhlukovania, ak vzdialenosť medzi najbližšími zhlukmi presiahne určitý prah, hovoríme o **algoritme jednoduchého spojenia.** Ak si predstavíme objekty ako vrcholy v grafe a hrany, ktoré predstavujú cestu medzi vrcholmi v zhluku, potom spojenie dvoch zhlukov C_i a C_j zodpovedá pridaniu hrany medzi najbližšou dvojicou vrcholov v C_i a C_j. Výsledný graf tvorí strom – aglomeratívne hierarchické zhlukovanie, ktoré používa minimálnu vzdialenosť, nazývame aj **algoritmus minimálnej kostry grafu**.

2. **Maximálna vzdialenosť** – vzdialenosť medzi dvoma objektmi z dvoch rôznych zhlukov, ktorá je najväčšia:

$$d^*_{\max}(C_i, C_j) = \max\{d(o, o') : o \in C_i, o' \in C_j\}.$$

   V tomto prípade hovoríme o zhlukovacom algoritme **najvzdialenejších susedov**. Ak je navyše daná podmienka ukončenia zhlukovania, ak vzdialenosť medzi najbližšími zhlukmi presiahne určitý prah, hovoríme o **algoritme úplného spojenia.** Ak si predstavíme objekty ako vrcholy v grafe a hrany, ktoré spájajú vrcholy, tak každý zhluk predstavuje úplný podgraf, teda hrany spájajú všetky vrcholy v zhluku. Ak sú zhluky približne rovnakej veľkosti, metóda produkuje zhluky vysokej kvality.

3. **Priemerná vzdialenosť** – priemerná vzdialenosť medzi všetkými dvojicami objektov z dvoch rôznych zhlukov:

$$d^*_{\text{average}}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{o \in C_i} \sum_{o' \in C_j} d(o, o'),$$

   pričom n_i a n_j predstavujú počet objektov v zhluku C_i a C_j pre $i \in \{1, 2, \ldots, k\}$, $j \in \{1, 2, \ldots, k\}$.

4. **Vzdialenosť priemerov** – vzdialenosť medzi centrami dvoch zhlukov.

   Miery minimálnej a maximálnej vzdialenosti sú citlivé na extrémne hodnoty. V takýchto prípadoch je vhodné používať priemernú vzdialenosť alebo vzdialenosť priemerov. Kým vzdialenosť priemerov má jednoduchý výpočet, priemerná vzdialenosť dokáže spracovať aj kategoriálne a numerické údaje.

Algoritmus aglomeratívneho hierarchického zhlukovania pre jednoduché spojenie môžeme vo všeobecnosti zapísať nasledovne:

Algoritmus 9.3 *Aglomeratívne hierarchické zhlukovanie pre jednoduché spojenie*

Nech X je matica údajov, ktorá reprezentuje hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$. Nech θ je prah spájania zhlukov.

begin

pre každý objekt o_i, $i \in \{1, 2, \ldots, n\}$ vytvor zhluk C_i, $i \in \{1, 2, \ldots, n\}$;

k:= n;

repeat

$minVzdialenost := min\{d^*_{min}(C_i, C_j) : i \in \{1, 2, \ldots, k\}, j \in \{1, 2, \ldots, k\}, i \neq j\}$;

ak $d^*_{textrmmin}(C_i, C_j) = minVzdialenost$

$C_i := C_i \cup C_j$;

zmeň označenia zhlukov na $C_1, C_2, \ldots, C_{k-1}$;

k:=k-1;

until *$minVzdialenost > \theta$;*

end

Ak v jednej iterácii dôjde k spojeniu dvoch zhlukov, nasledujúci krok už pracuje iba s jedným novým zhlukom, ktorý už nie je možné znovu rozdeliť. Túto nevýhodu je možné odstrániť použitím kombinovaných metód zhlukovania.

9.4 Metódy zhlukovania založené na hustote

Metódy zhlukovania založené na hustote definujú zhluk ako maximálnu množinu husto spojených objektov. Táto metóda si vyžaduje použitie nasledujúcich pojmov:

- **ε-okolie objektu** je okolie objektu s polomerom ε.
- **Vnútorný objekt** je objekt, ktorého ε-okolie obsahuje aspoň m objektov.
- Objekt o je **priamo dosiahnuteľný vzhľadom na hustotu** z objektu o', ak o je v ε-okolí o' a o' je vnútorným objektom.
- Objekt o je **dosiahnuteľný vzhľadom na hustotu** z objektu o' pre dané ε a m, ak existuje postupnosť objektov $o_1, o_2, \ldots, o_t$ taká, že $o_1 = o'$ a $o_t = o$, pričom o_{i+1} je priamo dosiahnuteľný vzhľadom na hustotu z objektu o_i pre $i \in \{1, 2, \ldots, t\}$, ε a m.
- Objekt o je **husto spojený** s objektom o' s ohľadom na ε a m, ak existuje objekt o^* taký, že o a o' sú dosiahnuteľné vzhľadom na hustotu s ohľadom na ε a m.

Iba vnútorné objekty sú navzájom dosiahnuteľné vzhľadom na hustotu. Husté spojenie je symetrická relácia. **Zhluk založený na hustote** je množina husto spojených objektov, ktoré sú maximálne z pohľadu dosiahnuteľnosti vzhľadom na hustotu.

Algoritmus zhlukovania založený na hustote môžeme vo všeobecnosti zapísať nasledovne:

Algoritmus 9.4 *Algoritmus zhlukovania založený na hustote*

Nech X je matica údajov, ktorá reprezentuje hodnoty n objektov $\{o_1, o_2, \ldots, o_n\}$ v p atribútoch $\{a_1, a_2, \ldots, a_p\}$. Nech ε je okolie objektov a m je minimálny počet objektov v ε okolí vnútorného objektu.

begin

pre každý objekt o_i, $i \in \{1, 2, \ldots, n\}$:

m_i:= počet objektov v ε-okolí objektu o_i;

ak $m_i > m$:

vytvor nový zhluk s vnútorným objektom o_i;

repeat

pre každý vnútorný objekt o_i

pridaj do zhluku C_i všetky priamo dosiahnuteľné objekty o z o_i;

until *žiaden objekt o nie je pridaný do niektorého zhluku;*

end

Vhodným zvolením okolia ε a minimálneho počtu objektov v okolí vnútorného bodu m, je možné pomocou metód zhlukovania založených na hustote nájsť tvary zhlukov, ktoré sú odlišné od tvarov zhlukov generovaných segmentačnými a hierarchickými metód. Prezentovaný algoritmus je možné nájsť v literatúre aj pod názvom DBSCAN (Density Based Spatial Clustering of Applications with Noise). Výhodou algoritmu je aj schopnosť pracovať so zašumenými údajmi.

9.5 Požiadavky na zhlukovanie

Potenciálne aplikácie zhlukovania v praxi si vyžadujú špeciálne požiadavky. Napríklad v oblasti strojové učenia a dolovania údajov môžeme uvažovať:

- **Schopnosť pracovať s rôznym typom atribútov:** Mnoho algoritmov je navrhnutých na zhlukovanie objektov s numerickými atribútmi. Aplikácie môžu vyžadovať vytváranie zhlukov aj pre iné typy údajov, akými sú napríklad binárne, kategorické (nominálne) a ordinálne údaje, alebo kombinácia týchto typov údajov.
- **Škálovateľnosť:** Väčšina algoritmov zhlukovania má dobré výsledky na malých množinách údajov s niekoľkými stovkami objektov. Veľká databáza môže obsahovať milióny objektov. Zhlukovanie na vzorke daného veľkého súboru údajov tak môže viesť k skresleným výsledkom. Je potrebné teda využívať vysoko škálovateľné algoritmy zhlukovania.
- **Znalosť aplikačnej domény na určenie vstupných parametrov:** Niektoré algoritmy zhlukovania vyžadujú, aby používatelia zadali určité vstupné parametre zhlukovania (napríklad počet požadovaných zhlukov). Výsledky zhlukovania môžu byť citlivé na vstupné parametre. Parametre je často ťažké určiť, najmä pre množiny údajov obsahujúce vysokodimenzionálne objekty.
- **Nájdenie zhlukov ľubovoľného tvaru:** Viacero algoritmov zhlukovania je založených na euklidovských alebo manhattanských mierach. Metódy s meraním vzdialenosti majú tendenciu nájsť zhluky s podobnou veľkosťou a hustotou. Zhluk však môže mať akýkoľvek tvar, preto je vhodné poznať aj algoritmy na detekciu zhlukov ľubovoľného tvaru.
- **Schopnosť práce so zašumenými údajmi:** Databázy v reálnom svete obsahuje anomálne hodnoty alebo chýbajúce, neznáme, či chybné údaje. Niektoré algoritmy zhlukovania sú na tieto údaje citlivé a môžu viesť k zoskupeniam nízkej kvality.
- **Dimenzionalita údajov:** Databáza alebo dátový sklad môže obsahovať niekoľko dimenzií. Mnoho algoritmov je vhodných pri spracovávaní nízkodimenzionálnych údajov, ktoré obsahujú dva až tri rozmery. Ľudské oči dokážu posúdiť kvalitu zhlukovania až do troch dimenzií. Hľadanie zhlukov pre objekty vo vysoko rozmernom priestore je náročné, prípadne takýchto objektov môže byť málo a výsledok môže byť skreslený.
- **Zhlukovanie založené na obmedzeniach:** Je možné, že aplikácie v reálnom svete budú musieť vykonať zhlukovanie vzhľadom na rôzne obmedzenia. Napríklad, ak úlohou je výber vhodných miest pre daný počet nových bankomatov v meste, pri riešení môžeme zoskupovať domácnosti vzhľadom na rieky, diaľničné siete, typ a počet zákazníkov pre jeden zhluk. Nájsť skupiny s požadovaným správaním v zhlukoch, ktoré vyhovujú týmto obmedzeniam, môže byť náročnou úlohou.

- **Interpretovateľnosť a použiteľnosť:** Používatelia očakávajú, že ich zhluky budú interpretovateľné, zrozumiteľné a a použiteľné. To znamená, že vytváranie zhlukov bude potrebné spájať s konkrétnym významom interpretácie a aplikácie.

- **Prírastkové zhlukovanie, či necitlivosť na poradie vstupných údajov:** Niektoré zhlukovacie algoritmy nemôžu obsahovať novo-vložené údaje (t. j. aktualizáciu databázy) do existujúcich výslekov. Je potrebné použiť nové zhlukovanie od začiatku. Niektoré algoritmy zhlukovania sú citlivé na poradie vstupných údajov. Vzhľadom na vstupnú množinu objektov môže takýto algoritmus určiť rôzne zoskupenia v závislosti od poradia prezentácie vstupných objektov. Je dôležité zaoberať sa aj prírastkovými algoritmami zhlukovania a algoritmami, ktoré sú necitlivé na poradie vstupu.

Kapitola 10

Bayesovský prístup ku strojovému učeniu

10.1 Úvod

- Pridanie šumu do dát umožňuje dosiahnuť lepšie výsledky v predikcii.
- Sme schopní kvantifikovať neurčitosť, ktorá je prítomná v odhadoch našich parametrov, a z toho vyplývajúcej predikcie.
- Od idey, že máme neurčitosť v odhadoch našich parametrov, je len krok k uvažovaniu daných parametrov ako náhodných premenných.
- Bayesovské metódy sa tu stávajú veľmi dôležitými. Urobíme úvod do ich použitia podľa [6].

10.2 Hra Hod mincou

Idete cez trh a narazíte na stánok, kde sa zákazníci zúčastňujú hry *Hod mincou*. Pravidlá hry sú jednoduché:

- Zákazník hádže mincou 10 krát
- Ak padne hlava 6x alebo menej krát, zákazník dostáva naspäť dvojnásobok vložených peňazí (1 EUR stávka). Inak zákazník o peniaze príde.

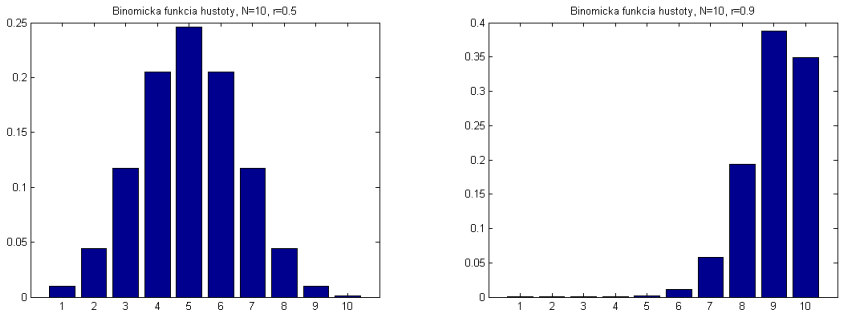
Úvahy o možnosti vyhrať:

Predpokladajme, že pravdepodobnosť, že padne hlava je r. Potom, pravdepodobnosť, že hlava padne y krát z N hodov, je

$$P(Y = y) = \binom{N}{y} r^y (1-r)^{N-y} \tag{10.1}$$

Pre $P(Y \leq 6)$ dostávame

$$P(Y \leq 6) = 1 - P(Y > 6) = 1 - (P(Y = 7) + P(Y = 8) + P(Y = 9) + P(Y = 10))$$



Obr. 10.1: Prvý panel predstavuje funkciu hustoty, ak minca je spravodlivá, t. j. $r = 0.5$, druhý panel predstavuje prípad, keď viac padá hlava, t. j. $r = 0.9$ (pri 10 hodoch).

$= 1 - (0.1172 + 0.0439 + 0.0098 + 0.001) = 0.8281$

čo vyzerá celkom zaujímavo. Je možné tiež vypočítať očakávanú finančnú sumu z hry. Očakávaná hodnota funkcie $f(X)$ náhodnej premennej X je počítaná

$$\mathbf{E}_{P(x)}\{f(X)\} = \sum_x f(x)P(x)$$

suma je cez všetky možné hodnoty náhodnej premennej X.

Nech X bude náhodná premenná, ktorá nadobúda hodnotu 1 pri výhre (padne hlava 6 a menej krát) a 0 inak: $P(X = 1) = P(Y \leq 6)$.

Ak $X = 1$, je to výhra a zákazník dostáva 2 EUR, teda $f(1) = 2$. Ak prehrá, nedostáva nič, teda $f(0) = 0$.

Očakávaný zisk je

$$f(1)P(X = 1) + f(0)P(X = 0) = 2 * P(Y \leq 6) + 0 * P(Y > 6) = 1.6562$$

Teda v priemere vyhráva 1.6562-1=0.6562 centov na hre.

Zdá sa, že hrať je OK. Ale je v tom háčik. V skutočnosti výhier nie je až tak veľa. Ďalším problémovým parametrom v hre môže byť r.

Pozorovanie

Možno predpoklady podliehajúce výpočtom sú nesprávne. Ide o nasledujúce:

- Počet hláv môže byť modelovaný ako náhodná premenná s binomickým rozdelením a pravdepodobnosť, že padne hlava v ľubovoľnom hode je r.
- Minca je spravodlivá - pravdepodobnosť oboch strán je $r = 0.5$.

Mohli by sme r uvažovať ako parameter a použiť ho pri fitovaní dát.

Úvahy o postupnosti hodov

Sú traja ľudia, ktorí hrajú túto hru. Predpokladajme, že prvý hráč dostane postupnosť: H, O, H, H, H, H, H, H, H, H.

Je možné vypočítať maximálnu hodnotu vierohodnosti (likelihood) na základe binomického rozdelenia pre r takto:

$$P(Y = y|r, N) = \binom{N}{y} r^y (1-r)^{N-y} \tag{10.2}$$

Zlogaritmovaním

$$L = log(P(Y = y|r, N)) = log\binom{N}{y} + y\,log\,r + (N-y)log\,(1-r) \tag{10.3}$$

Zderivujeme podľa r

$$\frac{\partial L}{\partial r} = \frac{y}{r} - \frac{N-y}{1-r} = 0$$

$$r = \frac{y}{N}$$

Ak $y = 9$ a $N = 10$ dostávame $r = 0.9$ a $P(Y \leq 6) = 0.0128$.

Očakávaný zisk $2 * P(Y \leq 6) + 0 * P(Y > 6) = 0.0256, 0.0256 - 1 = -0.9744$ na jednu hru. Teda strata 97 centov.

$P(Y \leq 6) = 0.0128$ znamená, že len jeden zákazník zo 100 vyhrá. Toto by mohlo odradiť ďalších hráčov.

10.3 Bayesovský prístup k hre

10.3.1 Bayesova veta - všeobecný tvar

Predpokladajme, že náhodné javy $B_i, 1 \leq i \leq k$ sú navzájom nezavislé javy a v každom pokuse nastáva práve jeden z nich, takže súčet ich pravdepodobností je rovný 1, t. j. $\sum_{i=1}^{k} P(B_i) = 1$

Ak poznáme podmienené pravdepodobnosti $P(A|B_i)$ javu A pri podmienke B_i pre všetky i a *apriórne* pravdepodobnosti javov $P(B_i)$, potom je možné pravdepodobnosť javu $P(B_j|A)$ vypočítať podľa Bayesovho vzorca:

$$P(B_j|A) = \frac{P(A|B_j) * P(B_j)}{P(A)} = \frac{P(A|B_j) * P(B_j)}{\sum_{i=1}^{k} P(A|B_i) * P(B_i)} \tag{10.4}$$

- $P(B_j|A)$ je posteriórna pravdepodobnosť javu B_j;
- $P(A|B_j)$ je dôveryhodnosť javu B_j;
- $P(B_j)$ je apriórna pravdepodobnosť javu B_j;
- $\sum_{i=1}^{k} P(A|B_i) * P(B_i)$ je marginálna pravdepodobnosť dát.

V prípade spojitých hodnôt premenných pravdepodobosť je nahradená funkciou hustoty pravdepodobnosti. Potom pre hypotézu B_j dostávame

$$f(B_j|A) = \frac{f(A|B_j) * f(B_j)}{P(A)} = \frac{f(A|B_j) * f(B_j)}{\int_{B_i} f(A|B_i) * f(B_i)\, dB_i} \tag{10.5}$$

- $f(B_j|A)$ je posteriórna hustota pravdepodobnosti pre B_j;
- $f(A|B_j)$ je dôveryhodnosť hypotézy B_j;
- $f(B_j)$ je apriórna hustota pravdepodobnosti pre B_j;
- $\int_{B_i} f(A|B_i) * f(B_i)\, dB_i$ je marginálna pravdepodobnosť dát.

10.3.2 Bayesova veta v hre s mincou

Hodnoty pravdepodobnosti r (že padne hlava) sa menia v čase a mohli by sme sa na ňu dívať ako na náhodnú premennú R.

Nezávisle od dĺžky postupnosti hodov vždy bude nejaká neurčitosť v r - budeme ju brať ako náhodnú premennú s asociovaným rozdelením. Toto nám pomôže merať a porozumieť túto neurčitosť.

Nech Y_N je náhodná premenná vyjadrujúca počet padnutých hláv v N hodoch. Chceme poznať pravdepodobnostné rozdelenie pre r podmienené hodnotami Y_N, teda $p(r|y_N)$.

Pri danom rozdelení bude možné vypočítať očakávanú pravdepodobnosť výhry tak, že vezmeme očakávania $P(Y_{nov} \leq 6|r)$ vzhľadom na $p(r|y_N)$:

$$P(Y_{nov} \leq 6|y_N) = \int P(Y_{nov} \leq 6|r)p(r|y_N)dr,$$

kde Y_{nov} je náhodná premenná opisujúca počet padnutých hláv v budúcej množine desiatich hodov.

Podľa Bayesovho pravidla dostaneme:

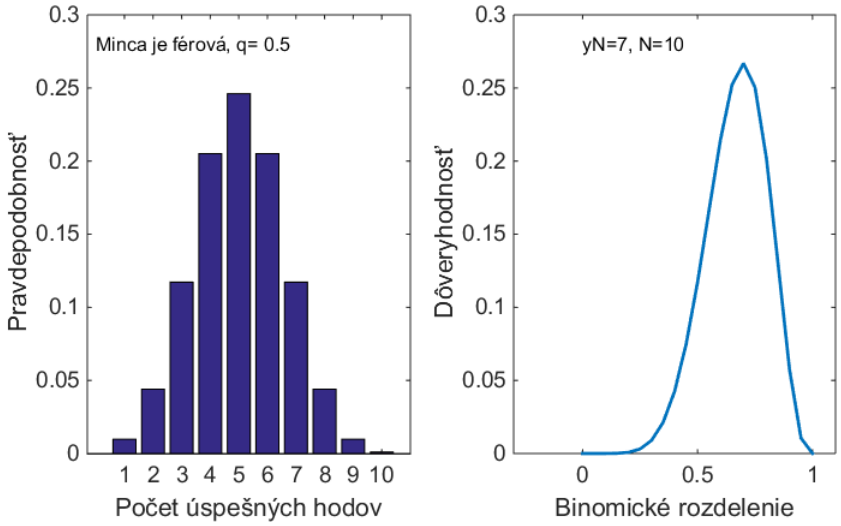
$$p(r|y_N) = \frac{P(y_N|r)p(r)}{P(y_N}.$$

$p(y_N|r)$ – pravdepodobnostné rozdelenie nad počtom hláv v N nezávislých hodoch za predpokladu, že pravdepodobnosť, že padne hlava v jednom hode, je r.

Príklad: Binomické rozdelenie pravdepodobnosti a dôveryhodnosť:

10.3.3 Pravdepodobnosť vs. dôveryhodnosť (likelihood)

- Binomické rozdelenie pravdepodobnosti je diskrétne (11 možných experimentálnych výsledkov podľa počtu hodených hláv).
- Dôveryhodnosť je spojitá, parameter r nadobúda hodnoty medzi 0 a 1.



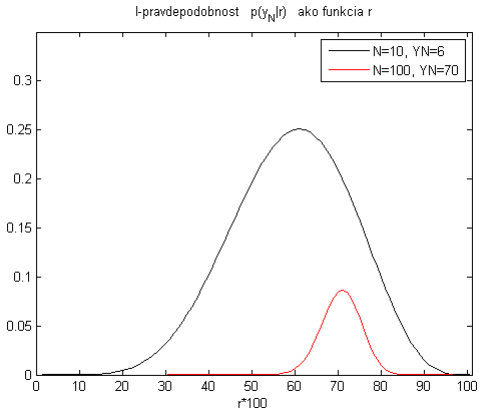
Obr. 10.2: Pri férovej minci a 7 úspešných hodoch z 10.

- Pravdepodobnosti v prvom paneli dávajú súčet 1.
- Dôveryhodnosť po zintegrovaní bude menej ako 1.

Dôveryhodnosť (likelihood), význam $P(y_N|r)$ - ako je dôveryhodné, že by sme mohli pozorovať naše dáta (y_N) pri určitej hodnote r (náš model).

V našom prípade to je binomické rozdelenie. Táto hodnota bude vysoká, ak r by mohlo nadobudnúť dosiahnuteľný výsledok y_N a nízka, ak je výsledok veľmi nepravdepodobný.

Dva scenáre pre rôzne nastavenia N a y_N



Obr. 10.3: Príklad dvoch scenárov pre vyjadrenie $p(y_N|r)$ ako funkcie r.

Dve dôležité vlastnosti dôveryhodnosti:

- Nie je to hustota pravdepodobnosti. Ak by bola, obsah plochy pod krivkou by bol rovný 1.
- Uvedené dva príklady sa líšia v tom, koľko nám povedia o r. V prvom príklade dôveryhodnosť je nenulová pre veľký interval pre r. V druhom príklade je tento interval redukovaný. Máme viac dát, a teda viac vieme (presnejšie) o r.

Apriórne rozdelenie (prior distribution), $p(r)$ - umožňuje vyjadriť nejakú vieru v r predtým, než niečo vieme o dátach.

Uvažujme nasledujúce 3 prípady:

1. Nevieme nič o hádzanej minci ani o vlastníkovi stánku.
2. Myslíme si, že minca je fér.
3. Myslíme si, že minca je vytvorená tak, že viac padá hlava.

Každý z týchto predpokladov vieme zakódovať iným apriórnym rozdelením.

r môže nadobúdať hodnoty od 0 po 1, preto musí byť modelované ako spojitá náhodná premenná.

Nevyberieme žiadny scenár, jednoducho budeme zatiaľ uvažovať $p(r|y_N)$.

Pre tento prípad je vhodná **beta funkcia hustoty** definovaná pre náhodnú premennú R ako:

$$p(r) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1} \tag{10.6}$$

Γ je známa funkcia, ktorá zaručí, že rozdelenie je normalizované. Parametre $\alpha, \beta \geq 0$ riadia tvar výsledného rozdelenia.

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\,dt, \quad \text{pre } x > 1$$

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} = \int_{r=0}^{r=1} r^{\alpha-1}(1-r)^{\beta-1}\,dr \tag{10.7}$$

zaručuje, že

$$\int_{r=0}^{r=1} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1}\,dr = 1$$

V uvedených troch prípadoch zvolíme:

- Nevieme nič: $\alpha = 1, \beta = 1$
- Férová minca: $\alpha = 50, \beta = 50$
- Neférová minca: $\alpha = 5, \beta = 1$

Marginálne rozdelenie (marginal distribution) pre $y_N, P(y_N)$ – pôsobí ako normalizujúca konštanta, ktorá zaručí, že $p(r|y_N)$ je vlastne definovaná hustota. Počíta sa ako združená hustota $p(y_N|r)$ cez r:

$$P(y_N) = \int_{r=0}^{r=1} p(y_N, r)\, dr \tag{10.8}$$

čo je možné vyjadriť

$$P(y_N) = \int_{r=0}^{r=1} P(y_N|r)p(r)\, dr \tag{10.9}$$

$P(y_N)$ je známa ako marginálna dôveryhodnosť dát, y_N, spriemernená cez všetky hodnoty parametrov.

Posteriórne rozdelenie (posterior distribution), $p(r|y_N)$ - o toto sa zaujímame.

Je to výsledok úpravy apriórnej pravdepodobnosti $p(r)$ vo svetle nových poznatkov o y_N.

- Tvar rozdelenia je zaujímavý - vyjadruje koľko informácie máme o r po kombinácii s tým, čo sme vedeli predtým a čo sme videli (dôveryhodnosť).
- Je to rozdelenie, preto nám poskytuje indikáciu o úrovni neurčitosti, stále máme v r informáciu o nejakých pozorovaných dátach.
- Posteriórne rozdelenie je možné použiť na výpočet očakávaní (výhry).

Posteriórnu hustotu vieme použiť na výpočet očakávanej hodnoty. Napríklad

$$\mathbf{E}_{p(r|y_N)}\{P(Y_{10} \leq 6)\} = \int_{r=0}^{r=1} P(Y_{10} \leq 6|r)\, p(r|y_N)\, dr$$

je očakávaná hodnota pravdepodobnosti, že vyhráme. Je potrebné brať do úvahy pozorované dáta, naša apriórna viera a neurčitosť zostáva. Treba sa rozhodnúť, či budeme hrať.

10.3.4 Dvojice dôveryhodnosť-apriórna pravdepodobnosť

Konjugované dvojice **apriórna – dôveryhodnosť** umožňujú použiť posteriórne rozdelenie v rovnakom tvare ako apriórne.

Apriórna pravdepodobnosť – dôveryhodnosť, niektoré konjugované dvojice:

- Gauss – Gauss
- Beta – Binomické
- Gamma – Gauss
- Dirichlet – Multinomial

10.3.5 Exaktné posteriórne rozdelenie

Beta rozdelenie je vhodný výber pre apriórnu pravdepodobnosť, ak dôveryhodnosť je binomické rozdelenie. Vypočítame ho presne.

Beta rozdelenie je známe ako konjugované rozdelenie k binomickej dôveryhodnosti. Použitie konjugovaného vzťahu uľahčí výpočty.

Ak v Bayesovom vzťahu vynecháme $P(y_N)$, dostaneme

$$p(r|y_N) \propto P(y_N|r)p(r)$$

Nahradením výrazov na pravej strane binomickým a beta rozd.

$$p(r|y_N) \propto [\binom{N}{y_N} r^{y_N}(1-r)^{N-y_N}] \times [\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} r^{\alpha-1}(1-r)^{\beta-1}] \qquad (10.10)$$

Pretože apriórne a beta rozd. sú konjugované, vieme, že $p(r|y_N)$ má byť beta hustota. Beta hustota s parametrami δ a γ a konštanou K má nasledujúci tvar

$$p(r) = Kr^{\delta-1}(1-r)^{\gamma-1}$$

Vhodnými úpravami dostaneme

$$p(r|y_N) \propto [\binom{N}{y_N} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}] \times [r^{y_N} r^{\alpha-1}(1-r)^{N-y_N}(1-r)^{\beta-1}]$$

$$p(r|y_N) \propto r^{y_N+\alpha-1}(1-r)^{N-y_N+\beta-1}$$

$$p(r|y_N) \propto r^{\delta-1}(1-r)^{\gamma-1}, \quad \delta = y_N+\alpha, \gamma = N-y_N+\beta$$

Preto

$$p(r|y_N) = \frac{\Gamma(\alpha+\beta+N)}{\Gamma(\alpha+y_N)\Gamma(\beta+N-y_N)} r^{\alpha+y_N-1}(1-r)^{\beta+N-y_N} \qquad (10.11)$$

Dostali sme posteriórnu hustotu pre r na základe apriórnej $p(r)$ a dát y_N.

Pripomeňme si, ako boli vypočítané - sčítaním počtov hodených hláv (y_n) do prvého apriórneho parametra α a počtov opaku mince $(N - y_N)$ do β.

Toto nám umožní získať intuíciu ohľadom α a β. Ak uvažujeme scenár korektnej mince, zo 100 hodov je 50 hlava a 50 opak mince. Mohli by sme použiť $\alpha = \beta = 50$,

Poznámka k trom scenárom

Dôslednú analýzu všetkých scenárov je možné nájsť v [6].

Chceme tu pripomenúť, že vo všetkých troch prípadoch dostaneme iné hodnoty očakávanej pravdepodobnosti pre výhru. Naše úvahy sú potvrdené tým, že najväčšia pravdepodobnosť je pri spravodlivej minci a najmenšia, ak hlava padá častejšie.

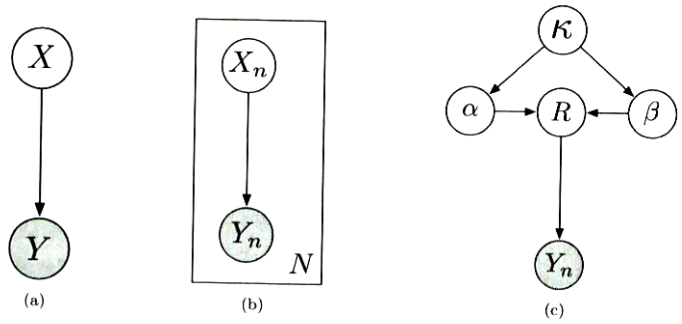
Dôležité je to, že Baeysovský prístup nám umožňuje kombinovať pozorované dáta (hody mincou) s nejakými predchádzajúcimi znalosťami (jeden zo scenárov) principiálnym spôsobom. Posteriórna hustota explicitne modeluje neurčitosť, ktorá zostáva v r v každom stave a môže byť použitá na predikcie.

Pridávanie ďalších údajov (hodov) mení scenár 1, lebo viac vieme o hodoch.

10.4 Grafické modely

Grafický model je graf, v ktorom vrcholy zodpovedajú náhodným premenným a orientované hrany závislostiam medzi náhodnými premennými.

Napríklad, jedna náhodná premenná reprezentuje hod mincou (X) a druhá predstavuje to, čo je výsledkom hodu (Y). Model je definovaný pomocou podmieneného rozdelenia $P(Y = y|X = x)$. Podmienená premenná je v šedom vrchole a k nej smeruje orientácia hrany. Nasledujúci obr. (a). Šedý vrchol vyjadruje pozorovanú premennú.



Obr. 10.4: Grafické modely vyjadrujúce vzťahy medzi náhodnými premennými, [6]

Ak si predstavíme, že hod mincou sa opakuje N-krát, máme $2N$ náhodných premenných $X_1, \ldots, X_N$ a $Y_1, \ldots, Y_N$. Toto zakreslíme pomocou obdĺžnika, v ktorom sa nachádza to, čo sa opakuje N-krát, na obr. (b). A je evidentné, že Y_n je podmienená od X_n.

Grafická reprezentácia nášho modelu pre hod mincou je na obr. (c). Má jednu pozorovanú náhodnú premennú, ktorá reprezentuje počet hodených hláv v N hodoch y_N. Toto je podmienené náhodnou premnnou R, ktorá závisí od náhodných premenných α a β. A nakoniec α a β závisia od nejakých hyperparametrov κ.

10.5 Bayesov prístup k problému predikcie rekordov na OH

Cieľ je predikovať výsledok na nasledujúcich OH z daných dát. K tomu budeme potrebovať:

- definovať apriórnu pravdepodobnosť a dôveryhodnosť
- vypočítať posteriórnu hustotu nad parametrami nášho modelu (r v prípade mince)
- keď máme posteriórnu hustotu, vieme predikovať.

Bayesov prístup - Model

Budeme používať polynomiálny model s Gaussovým šumom:

$$t_n = w_0 + w_1 x_n + w_2 x_n^2 + \cdots + w_K x_n^K + \epsilon_n, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2).$$

Vo vektorovom tvare

$t_n = \mathbf{w}^T \mathbf{x}_n + \epsilon_n$, $\mathbf{w} = [w_0, \ldots, w_K]^T$, $\mathbf{x}_n = [1, x_n, x_n^2, \ldots, x_n^K]^T$.

V maticovom tvare pre všetky dané dáta

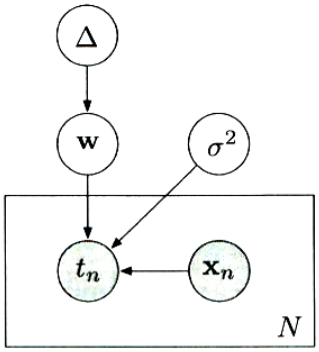
$$\mathbf{t} = \mathbf{X}\mathbf{w} + \epsilon, \quad \mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N]^T, \mathbf{t} = [t_1, t_2, \ldots, t_N]^T$$

Kvôli zjednodušeniu výpočtov budeme predpokladať, že poznáme σ^2.

Bayesovo pravidlo

$$p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2, \Delta) = \frac{p(\mathbf{t}|\mathbf{w}, \mathbf{X}, \sigma^2, \Delta) p(\mathbf{w}|\Delta)}{p(\mathbf{t}|\mathbf{X}, \sigma^2, \Delta)} = \frac{p(\mathbf{t}|\mathbf{w}, \mathbf{X}, \sigma^2) p(\mathbf{w}|\Delta)}{p(\mathbf{t}|\mathbf{X}, \sigma^2, \Delta)}$$

kde Δ predstavuje parametre potrebné pri definovaní apriórnej pravdepodobnosti pre $\mathbf{w}$ (uvedieme neskôr).



Obr. 10.5: Grafický model pre Bayesov prístup - rekordy na OH, [6].

10.5.1 Bayesov prístup k predikcii rekordov

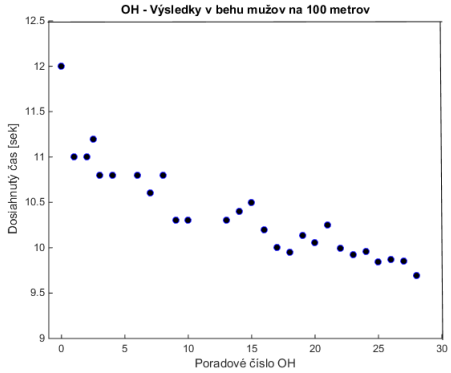
Potrebujeme určiť

- **apriórnu pravdepodobnosť** – $p(\mathbf{w}|\Delta)$

  Budeme používať Gaussovu apriórnu hustotu so zvolenými parametrami, ktoré označíme $\mu_0, \mathbf{\Sigma}_0$

  $$p(\mathbf{w}|\mu_0, \mathbf{\Sigma}_0) = \mathcal{N}(\mu_0, \mathbf{\Sigma}_0)$$

- **dôveryhodnosť** – $p(\mathbf{t}|\mathbf{w}, \mathbf{X}, \sigma^2)$



Obr. 10.6: Výsledky v behu mužov na 100 m na OH

Náš model je $\mathbf{t} = \mathbf{X}\mathbf{w} + \epsilon,$ kde $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_N)$. Teda

$$p(\mathbf{t}|\mathbf{w}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_n)$$

N-dimenzionálna Gaussova hustota s priemerom $\mathbf{X}\mathbf{w}$ a varianciou $\sigma^2\mathbf{I}_N$

- **posteriórnu pravdepodobnosť** Pretože vieme, že posteriórna pravdepodobnosť bude Gaussova, zatiaľ si nebudeme šímať marginálnu pravdepodobnosť, preto

$$p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2) \propto p(\mathbf{t}|\mathbf{w}, \mathbf{X}, \sigma^2)p(\mathbf{w}|\mu_0, \mathbf{\Sigma}_0) \tag{10.12}$$

$$= \frac{1}{(2\pi)^{N/2}|\sigma^2\mathbf{I}|^{1/2}} exp(-\frac{1}{2}(\mathbf{t} - \mathbf{X}\mathbf{w})^T(\sigma^2\mathbf{I})^{-1}(\mathbf{t} - \mathbf{X}\mathbf{w})))$$

Tiež tu použijeme Gaussovo rozdelenie

$$p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mu_\mathbf{w}, \Sigma_\mathbf{w}) \tag{10.13}$$

$$\propto exp(-\frac{1}{2}(\mathbf{w} - \mu_\mathbf{w})^T\Sigma_w^{-1}(\mathbf{w} - \mu_\mathbf{w}))$$

a na základe úprav a porovnania (10.12) a (10.13) dostaneme

$$p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mu_\mathbf{w}, \mathbf{\Sigma}_\mathbf{w}) \tag{10.14}$$

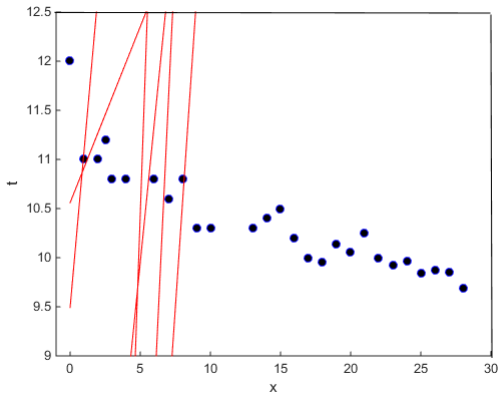
$$\mathbf{\Sigma}_\mathbf{w} = (\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_0^{-1})^{-1}, \tag{10.15}$$

$$\mu_\mathbf{w} = \mathbf{\Sigma}_\mathbf{w}(\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{t} + \mathbf{\Sigma}_0^{-1}\mu_0) \tag{10.16}$$

10.5.2 Prípad lineárneho modelu $t_n = w_0 + w_1x_n$

Predpokladajme, že časy x_n v behu na 100 m mužov na OH sú znázornené na obrázku 10.6

Vstupy sú teda $\mathbf{x}_n = [1, x_n]^T$. Predpokladáme, že reálne nič nevieme o tom, ako vybrať parametre, tak zvolíme $\mu_0 = [0,\ 0]^T$. Pre kovarianciu zvolíme $\mathbf{\Sigma}_0 =$

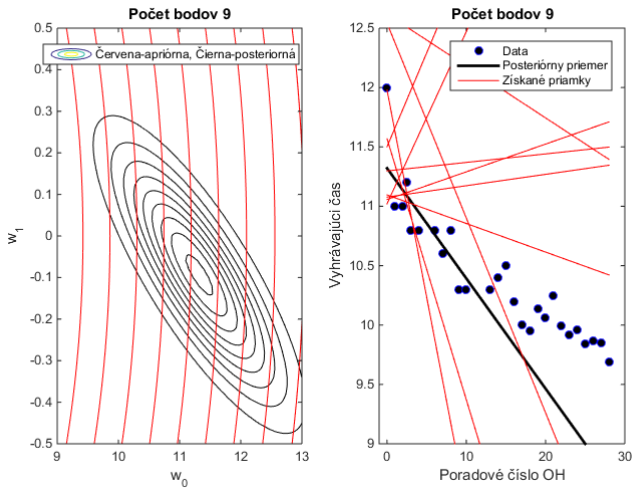


Obr. 10.7: Situácia po spracovaní prvého bodu.

$((100,\, 0), (0, 5))\,, \sigma^2 = 10$, pretože sme už videli odhad pre w_0, ktorý bol väčší ako pre w_1.

Na základe vzťahov (10.14), (10.15) a (10.16) pre $\sigma^2 = 10$ a prvý bod dát $\mathbf{X} = [1, 0], \mathbf{T} = 12$ vieme vypočítať posteriórne rozdelenie. Jeden bod nám niečo povie o hodnote w_0, ale takmer nič o hodnote w_1. Vieme vygenerovať nejaké priamky, ale tie sú od dát veľmi vzdialené, vidíme to na obrázku 10.7.

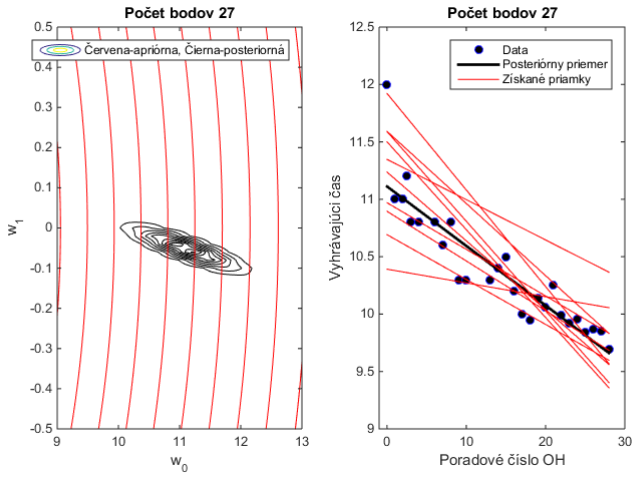
Ak vezmeme 9 bodov, situácia je lepšia tak, ako je to na obrázku .... Všetky body nám dajú výsledky znázornené na obrázku 10.8.



Obr. 10.8: Apriórna a posteriórna pravdepodobnosť po 9 OH

Pri použití všetkých bodov dostaneme výsledky znázornené na obrázku 10.9.

V obrázkoch 10.8 a 10.9 pozorujeme, že posteriórna pravdepodobnosť sa stala viac kondenzovaná, čo znamená, že máme viac poznatkov pre $\mathbf{w}$. Pozorujeme napríklad, že keď zvýšime w_0, musíme znížiť w_1. V apriórnej pravdepodobnosti sme predpokladali, že sú nezávislé.

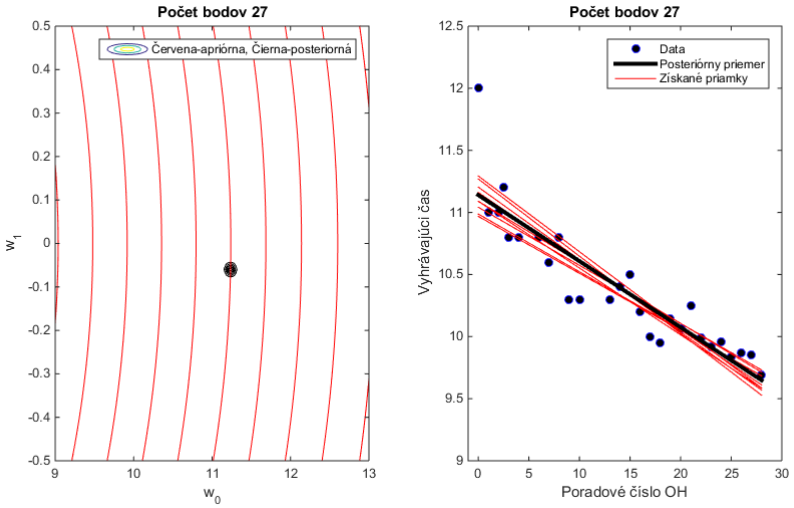


Obr. 10.9: Apriórna a posteriórna pravdepodobnosť po 27 OH

V obrázku 10.9 pozorujeme ešte dosť veľkú variabilitu, čo je spôsobené tým, že $\sigma^2 = 2$. Zmenšením σ je možné docieliť menšiu variabilitu pre w_0 a w_1.

10.5.3 Ako urobiť predikciu

Ak zvolíme $\sigma^2 = 0.05$, dostaneme výsledky znázornené na obrázku 10.10



Obr. 10.10: Apriórna a posteriórna pravdepodobnosť po 27 OH

Predpokladáme nové pozorovanie t_{nove}, teda nás zaujíma hustota

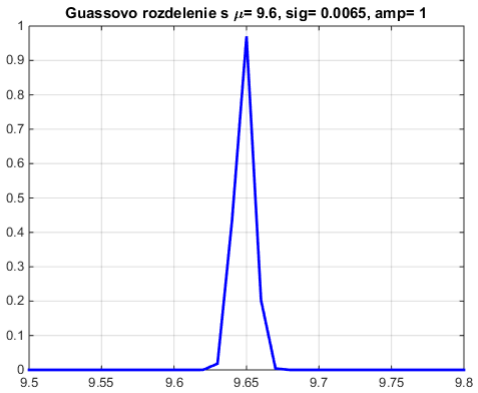
$$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \mathbf{t}, \sigma^2). \tag{10.17}$$

Výraz neobsahuje $\mathbf{w}$, ale naše očakávania sú zamerané na posteriórnu pravdepodobnosť $p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2)$. Potrebujeme vypočítať

$$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \mathbf{t}, \sigma^2) = \mathbf{E}_{p(\mathbf{w}|\mathbf{t},\mathbf{X},\sigma^2)}\{p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \sigma^2)\} \tag{10.18}$$
$$= \int p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \sigma^2) p(\mathbf{w}|\mathbf{t}, \mathbf{X}, \sigma^2)\, d\mathbf{w}.$$

$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \sigma^2)$ je v našom modeli definované ako súčin $\mathbf{x_{nove}}$ a $\mathbf{w}$ s pridaním nejakého gausssovského šumu.

$$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{x_{nove}}^T \mathbf{w}, \sigma^2).$$



Obr. 10.11: Predikcia na 28 OH

Výsledok teda očakávame v tvare Gaussovej funkcie. Teda

$$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \mathbf{t}, \sigma^2) = \mathcal{N}(\mathbf{x_{nove}^T}\mu_w, \sigma^2 + \mathbf{x_{nove}^T}\Sigma_w \mathbf{x_{nove}})$$

Pre situáciu na obrázku 10.10 dostávame

$$p(t_{nove}|\mathbf{x_{nove}}, \mathbf{X}, \mathbf{t}, \sigma^2) = \mathcal{N}(9.6484, 0.0065)$$

Prediktívne rozdelenie je na obrázku 10.11. Najviac očávaný je čas 9.65, ale je nenulová pravdepodobnosť aj horších časov.

Príklad 2-dimenzionálnej Gaussovej funkcie:

$$f(x, y) = A \exp\left(-\left(\frac{(x - x_o)^2}{2\sigma_X^2} + \frac{(y - y_o)^2}{2\sigma_Y^2}\right)\right).$$

Koeficient A je amplitúda, (x_o, y_o) je centrum.

10.5.4 Úlohy

1. Odvodili sme pre výraz pre Gaussovu posteriórnu pravdepodobnosť pre lineárny model (rekordy na OH). Ak dosadíme za $\mu_{\mathbf{0}} = [0, 0, \ldots, 0]^T$, vidíme určitú podobnosť priemerom posteriórnej pravdepodobnosti

$$\mu_{\mathbf{w}} = \frac{1}{\sigma^2}(\frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma_0}^{-1})^{-1}\mathbf{X}^T\mathbf{t} \tag{10.19}$$

   regularizovaným riešením pomocou metódy najmenších štvorcov

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + N\lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{t} \tag{10.20}$$

   Nájdite kovariančnú maticu $\mathbf{\Sigma_0}$, ktorá urobí tieto výrazy identickými, t. j. $\mathbf{\Sigma_0}$ pomocou λ.

2. V príklade OH analyzujte efekt redukcie σ^2.

Literatúra

[1] M. Anthony, N. Biggs: Computational Learning Theory, Cambridge University Press, 1991, 1997

[2] I. Goodfellow, Y. Bengio, A. Courville: Deep Learning. MIT Press book in preparation, http://www.deeplearningbook.org/version-2016-03-08/

[3] J. Han, M. Kamber, J. Pei: Data Mining – Concepts and Techniques, 3rd edition. Morgan Kaufmann Publishers, 2011

[4] T. Mitchell: Machine Learning, McGraw Hill, 1997

[5] J. D. Riggs, T. L. Lalonde: Non-Gaussian and correlated data. Handbook for applied modeling. Cambridge University Press, 2017

[6] S. Rogers, M. Girolami: A First Course in Machine learning. CRC Press, Taylor & Francis Group, 2012, 2017

[7] J. Watt, R. Borhani, A. K. Katsaggelos: Machine Learning Refined. Foundations, Algorithms, and applications. Cambridge University Press, 2016

[8] L. G. Valiant: Theory of the Learnable. Comm. of the ACM, Vol. 27, No. 1, November 1984

Strojové učenie
Vysokoškolský učebný text

Autori: doc. RNDr. Gabriela Andrejková, CSc.
RNDr. Ľubomír Antoni, PhD.

Vydavateľ: Univerzita Pavla Jozefa Šafárika v Košiciach
Vydavateľstvo ŠafárikPress

Rok vydania: 2020
Počet strán: 91
Rozsah: 4,5 AH
Vydanie: prvé

ISBN 978-80-8152-912-2 (e-publikácia)