Navrhování skupin klasifikátorů pomocí opakovaného výběru trénovacích dat

Jednou z často využívaných možností při návrhu klasifikátoru jsou *metody opakovaného výběru* trénovacích dat tak, aby byl dosažen co nejlepší odhad statistiky pro předpověď vlastností klasifikace. Výraznou roli zde hraje skutečnost, že budou klasifikována nějaká budoucí data, která samozřejmě nemohla být použita v okamžiku návrhu klasifikátoru pomocí trénovacích příkladů. Potíže při návrhu jsou také způsobeny neznámým rozložením hodnot klasifikovaných vzorů.

Výběrem vzorků dat se všeobecně zabývá statistika, zde se zmíníme o několika vybraných typických metodách používaných ve strojovém učení za situace, kdy je nutno vycházet z reálných disponibilních dat.

Existují metody opakovaného výběru vzorků, které jsou obecně zaměřeny na techniky trénování klasifikátorů.

Bagging

Metoda *bagging* ("rozdělování do sáčků") patří k nejjednodušším postupům. Název je odvozen ze slov ***b**ootstrap **agg**regation*, kde se využívá více trénovacích množin, z nichž každá je vytvořena výběrem $n' < n$ příkladů ze základní trénovací množiny D. Každá z takto vzniklých podmnožin $d_i \subset D$ je použita k natrénování jednoho z více klasifikátorů. Výsledná klasifikace neznámého datového vzorku je pak určena "hlasováním" všech takto vytvořených klasifikátorů, tj. vzorek je zařazen do třídy určené většinou. Obvykle jsou všechny tyto klasifikátory stejného typu (např. rozhodovací stromy nebo umělé neuronové sítě)—to, že mohou rozhodovat různě, souvisí s natrénováním pomocí různých trénovacích podmnožin d_i.

Strojové učení—boosting (průsvitky z přednášek; literatura: R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, 2nd Edition, John Wiley & Sons, 2001)

Nestabilita klasifikačního algoritmu je vlastnost, kdy vlivem malých změn trénovacích dat dochází ke vzniku podstatně různých klasifikátorů s relativně velkými rozdíly v klasifikační přesnosti (tj. podílu množství správně klasifikovaných vzorků k celkovému množství). K tomu dochází mj. i u rozhodovacích stromů. *Bagging* obecně zlepšuje vlastnosti nestabilních klasifikátorů vzhledem k tomu, že se bere do úvahy průměr z názorů jednotlivých klasifikátorů, avšak teoretický důkaz tohoto zlepšení pro libovolný případ neexistuje—metoda vychází z experimentálních zkušeností a není záruka, že bude fungovat vždy, i když velmi často ke zlepšení vede.

Boosting

Cílem metody *boosting* je zlepšení klasifikační přesnosti libovolného algoritmu strojového učení. I zde je základem vytvoření více klasifikátorů pomocí výběru vzorků ze základní trénovací množiny D.

Boosting vychází z vytvoření prvního klasifikátoru, jehož klasifikační přesnost je lepší než 50%. Dále jsou přidávány další klasifikátory mající stejnou klasifikační vlastnost, takže je vygenerován soubor klasifikátorů, jehož celková klasifikační přesnost je libovolně vysoká vzhledem ke vzorkům v trénovací množině—klasifikace byla zesílena (*boosted*).

Příklad: nechť má skupina klasifikátorů tři členy a klasifikace je do dvou tříd. Prvně se náhodně vybere (bez vracení) $n_1 < n$ trénovacích prvků z množiny D (n je počet prvků v D) a vznikne podmnožina D_1, na níž je natrénován první klasifikátor C_1. Stačí, když C_1 je pouze tzv. *slabý žák (weak learner)*, tj. může být pouze o málo lepší v klasifikaci než kdyby hádal náhodně—přesnost jen o málo lepší než 50% je pouze minimální požadavek; na trénovacích datech může také být přesnost samozřejmě velmi vysoká.

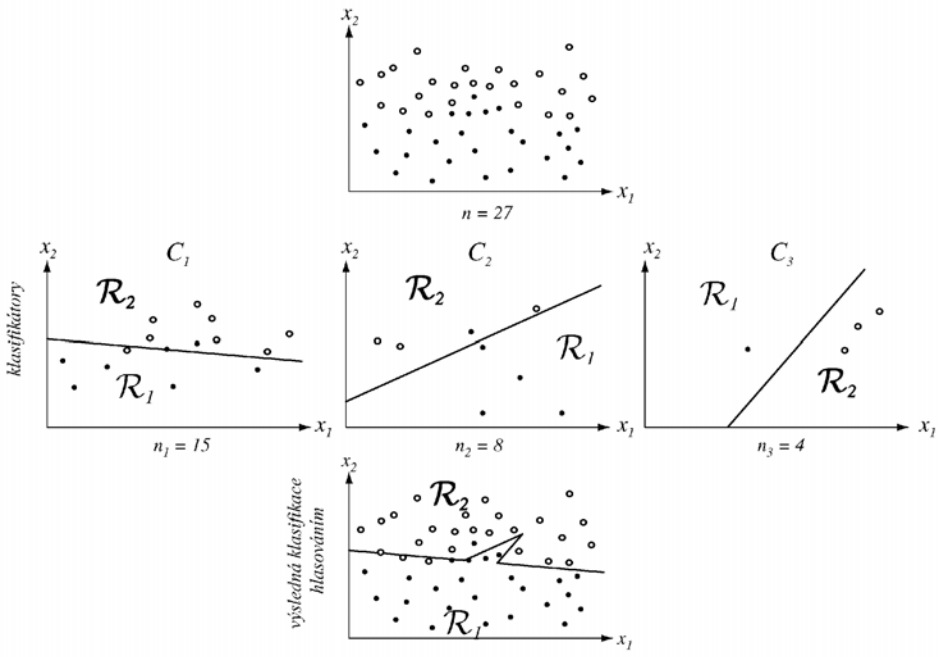
V dalším kroku se vytvoří trénovací množina D_2 (n_2 trénovacích prvků) vzhledem k již existujícímu C_1 tak, aby C_1 správně klasifikoval pouze 50% z D_2 (ze zbylých prvků v D se vyhledá postupně požadované množství správně a nesprávně klasifikovaných). Nyní se natrénuje druhý klasifikátor C_2 pomocí podmnožiny D_2, na níž C_1 selhává.

Nakonec se obdobně vytvoří D_3 (n_3 trénovacích prvků) vzhledem k již existujícím C_1 a C_2 a natrénuje se C_3. Zde hraje roli to, zda C_1 a C_2 se shodují či neshodují v klasifikaci vzorku—podle toho se vybírají prvky do D_3 tak, aby již vytvořené klasifikátory na nich při společné klasifikaci selhávaly.

Neznámý prvek **x** je pak při klasifikaci zařazen do třídy, na níž se shodne většina z vytvořených klasifikátorů. Např. když se C_1 a C_2 shodují, určují výsledek. Neshodují-li se, je výsledek dán pomocí rozhodnutí C_3 (který se zde tedy shodne buď s C_1 nebo C_2). Skupinové rozhodování dává lepší přesnost než libovolný individuální člen.

Pro stanovení hodnot n_i neexistuje jednoznačný algoritmus. Obecný požadavek je, aby se trénování účastnily všechny vzorky z D, a dále aby počet trénovacích vzorků pro každý klasifikátor byl přibližně stejný vzhledem k tomu, že klasifikátory mají rovnocenný hlas při hlasování—měly by tedy být i rovnocenně vytvořené. Lze stanovit, aby ve výše uvedeném příkladu přibližně $n_i = n/3$. V praxi mohou ale vzniknout potíže: snadnější klasifikační problémy mohou vést k tomu, že C_1 prostě funguje na většině dat, takže n_2 (i n_3) pak bude značně menší než $n_1 = n/3$ (aby se vyhovělo podmínkám vytváření podmnožin) a všechny prvky z D nebudou využity. Na druhé straně může pro obtížné klasifikační problémy dojít k tomu, že C_1 správně klasifikuje pouze velmi málo trénovacích vzorků, takže $n_1 << n/3$ a n_2 bude velmi vysoké. Řešením je (vzhledem k náhodnosti výběru n_i) opakovat celou *boosting* proceduru několikrát, aby se zajistila co nejlepší hodnota pro n_1, přibližně rovnoměrné rozdělení počtu prvků do podmnožin a pokud možno i využití *všech* trénovacích vzorků. K tomu se používají různé heuristiky snažící se dodržet výše uvedené požadavky.

Strojové učení—boosting (průsvitky z přednášek; literatura: R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, 2nd Edition, John Wiley & Sons, 2001)



Obrázek ukazuje případ klasifikace dvourozměrného problému, kde jsou dvě klasifikační třídy (symbolizované znaky ● pro třídu $\mathcal{R}_1$ a ○ pro $\mathcal{R}_2$) a tři lineární klasifikátory (zde—což není podstatné—typu *Widrow-Hoff,* tj. gradientní hledání lineární hranice oddělující třídy vzhledem k nejmenšímu čtverci chyb). Hlasující trojice má v tomto případě vždy lepší klasifikační přesnost než jednotlivé klasifikátory trénované na podmnožinách. Přesnost je také vyšší než kdyby byl vygenerován jediný klasifikátor na všech trénovacích vzorcích.

AdaBoost

AdaBoost (***ada****ptive* ***boost****ing*) je v současnosti nejpoužívanější variantou metody *boosting*—umožňuje při návrhu přidávat slabé žáky tak dlouho, dokud není dosažena určitá požadovaná nízká hodnota klasifikační chyby skupiny klasifikátorů.

Každý trénovací vzorek dostane přidělenu váhu, která určuje pravděpodobnost jeho výběru do trénovací podmnožiny pro jednotlivé klasifikátory z vytvářené skupiny.

Je-li vzorek klasifikován přesně, jeho šance na opětovný výběr pro následující klasifikátor klesá; v opačném případě roste. *AdaBoost* se takto soustředí na obtížné vzorky.

Na počátku dostanou vzorky stejné hodnoty vah. V každém iteračním kroku k je výběrem tvořena trénovací množina vzhledem k těmto váhám (které ovlivňují pravděpodobnost výběru). Na této množině je natrénován klasifikátor C_k.

V následujícím kroku jsou zvýšeny váhy vzorků klasifikovaných chybně a sníženy váhy vzorků klasifikovaných správně pomocí C_k. Vzorky jsou pak na základě nového rozdělení pravděpodobnosti výběru použity při vytváření dalšího klasifikátoru pomocí C_{k+1} a celý proces iterativně pokračuje.

Nechť vzorky (vektory hodnot atributů) $\mathbf{x}^i$ v celkové trénovací množině D mají klasifikace y_i.

Nechť $W_k(i)$ je k-té diskrétní rozdělení vah přes všechny trénovací vzorky.

Dále nechť α_k je vypočtená váha klasifikátoru C_k podle jeho přesnosti a Z_k je normalizační konstanta.

Proceduru *AdaBoost* pak lze zapsat pomocí následujícího pseudokódu:

---

1 **procedure** AdaBoost
2 **begin initialize** $D = \{\mathbf{x}^1, y_1, ..., \mathbf{x}^n, y_n\}$, k_{max}, $W_1(i) = 1/n$, $i = 1, ..., n$
3 $k \leftarrow 0$
4 **do** $k \leftarrow k + 1$
5 trénuj slabého žáka C_k pomocí vzorků z D vzhledem k $W_k(i)$
6 $E_k \leftarrow$ trénovací chyba C_k měřená na D vzhledem k $W_k(i)$
7 $\alpha_k \leftarrow \frac{1}{2}\ln\left(\frac{1-E_k}{E_k}\right)$
8 $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{pro } h_k(\mathbf{x}^i) = y_i \text{ (správná klasifikace)} \\ e^{\alpha_k} & \text{pro } h_k(\mathbf{x}^i) \neq y_i \text{ (chybná klasifikace)} \end{cases}$
9 **until** $k = k_{max}$
10 **return** C_k a α_k pro $k = 1, ..., k_{max}$ (vrací klasifikátory a jejich váhy)
11 **end**

Poznámky:

Řádek 5: Chyba klasifikátoru C_k je stanovena vzhledem k rozdělení $W_k(i)$ nad D, pomocí níž je trénován.

Řádek 8: Z_k je normalizační konstanta stanovená tak, aby $W_k(i)$ představovalo skutečné rozdělení, a $h_k(\mathbf{x}^i)$ je označení kategorie přidělené vzorku $\mathbf{x}^i$ klasifikátorem C_k (který je jedním z "hlasující" skupiny klasifikátorů).

Řádek 9: Ukončení cyklu **do—until** může být také založeno na dosažení dostatečně nízké chyby celé skupiny klasifikátorů.

Výsledné klasifikační rozhodnutí pro *testovací* vzorek **x** je založeno na diskriminační funkci, která je dána váhovaným součtem výsledků jednotlivých klasifikátorů:

$$g(\mathbf{x}) = \left[\sum_{k=1}^{k_{max}} \alpha_k h_k(\mathbf{x})\right]$$

I když mohou existovat případy dat, pro něž uvedený postup nemusí poskytnout očekávané výsledky, tak skupina slabých trénovaných "žáků" je schopna docílit libovolně nízké klasifikační chyby (pro *trénovací* data) pro dostatečně velký počet klasifikátorů k_{max}.

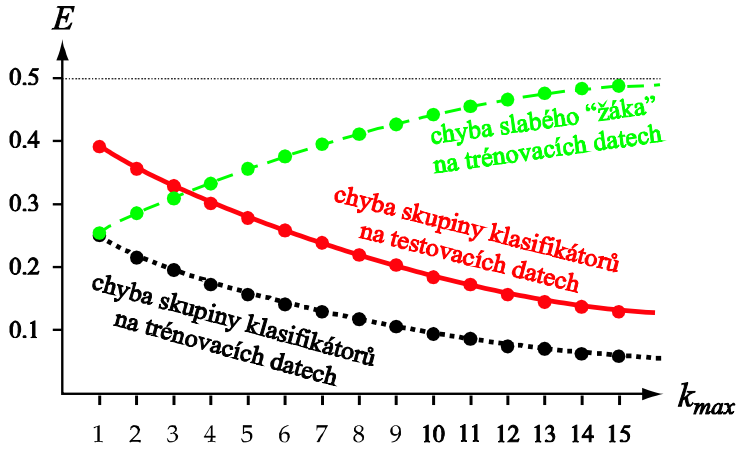
Trénovací chyba slabého žáka C_k je:

$$E_k = \frac{1}{2} - G_k$$

pro nějakou kladnou hodnotu G_k (tj. $E_k < 50\%$, žák dokáže správně zařadit o něco více než ½ trénovacích příkladů do příslušných kategorií). Chyba skupiny klasifikátorů pak je:

$$E = \prod_{k=1}^{k_{max}} \left[2\sqrt{E_k(1-E_k)}\right] = \prod_{k=1}^{k_{max}} \sqrt{1-4G_k^2} \le e^{-2\sum_{k=1}^{k_{max}} G_k^2},$$

což ilustruje následující obrázek:



AdaBoost redukuje trénovací chybu exponenciálně v závislosti na rostoucím počtu klasifikátorů k_{max} v "hlasující" skupině. Algoritmus *AdaBoost* je zaměřen na ty příklady, které způsobují klasifikační potíže, takže každý další přidaný klasifikátor má obecně větší chybu na trénovacích datech než kterýkoliv z předchozích (viz křivku *chyba slabého žáka na trénovacích datech*). Dokud každý z klasifikátorů ve skupině dává lepší než náhodný výsledek (tj. např. pro dvě kategorie je chyba menší než 0.5), váhované rozhodnutí celé skupiny zaručuje pokles chyby na trénovacích datech. Velmi často dochází k podobnému poklesu i na testovacích datech.

Zvyšování počtu klasifikátorů ve skupině ale může vést k tzv. *přetrénování* (ztrátě schopnosti generalizovat vlivem přílišného zaměření klasifikátorů na rozeznávání pouze konkrétních trénovacích dat). Simulační experimenty však ukázaly, že k tomu dochází relativně zřídka i pro extrémně vysoké hodnoty k_{max}. Pro praktickou aplikaci *AdaBoost* platí základní pravidlo:

> *Boosting* zlepšuje klasifikaci pouze tehdy, pokud klasifikátory ve skupině poskytují lepší než jenom náhodné výsledky—to však nelze zaručit předem.

Navzdory neexistenci záruky *a priori*, uvedená metoda poskytuje v mnoha reálných problémech velmi dobré výsledky.