Jak mravenčí kolonie dobývají znalosti

Daniel Vodák, Luboš Popelínský

Laboratoř dobývání znalostí
Fakulta informatiky, Masarykova univerzita v Brně,
Botanická 68a, 602 00, Brno
{xvodak, popel}@fi.muni.cz

Abstrakt Mezi nová řešení klasifikačních úloh patří přístupy inspirované sociálním chováním společenstev hmyzu - včel, termitů nebo mravenců. V této práci přinášíme kritickou analýzu systému Ant-Miner a uvádíme jeho vylepšenou verzi. Uvádíme nejprve analýzu časové náročnosti původního algoritmu GUI Ant-Miner. Ukážeme, že datová sada Wisconsin Breast Cancer použitá pro experimenty ve všech předchozích publikovaných článcích není vhodná. Poté popíšeme novou verzi, Ant-Miner$^+$, včetně nového pravidla pro určení kvality naučených dat. Uvádíme výsledky experimentů s tímto systémem a porovnání s jinými učícími algoritmy.

Klíčová slova: swarm intelligence, kolonie mravenců, strojové učení, klasifikace

1 Inspirace

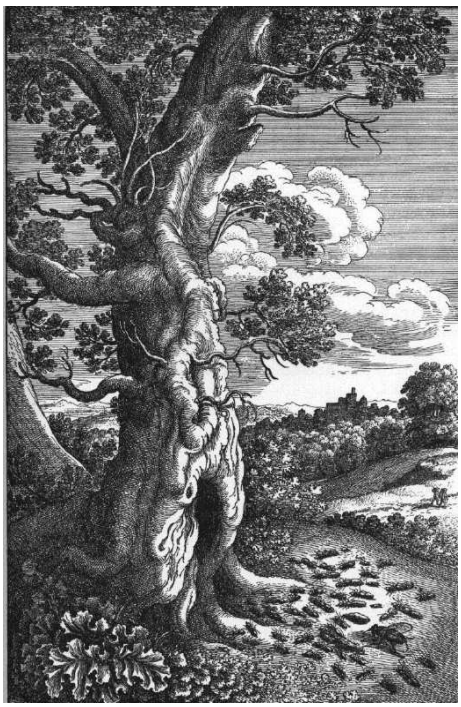
Koncept klasifikačních úloh je v oblasti dolování znalostí již dlouho znám a hojně využíván. Pomocí učící množiny obsahující příklady (databázové položky – přiřazení hodnot z pevně daných domén každému ze sady pevně daných atributů), u nichž je známa náležitost k jedné z predefinovaných tříd , je zkonstruován klasifikátor – program schopný roztřídit příklady, jejichž třída dosud známa není. Základní koncept však zanechává mnoho prostoru pro konkrétní realizaci úlohy, zejména pak pro konstrukci klasifikátoru, jeho strukturu a odpovídající způsob použití.

Mezi neklasická řešení patří přístupy inspirované biologickými systémy, mimo jiné sociálním chováním pozorovaným ve společenstvech hmyzu – včel, termitů, mravenců – a označovaným jako *swarm intelligence* [3, 7]. V principu se vždy jedná o totéž. V zásadě primitivní agenti nezávisle na sobě pracují na svém jednoduchém, dílčím úkolu. Vhodné spojení výsledků práce jednotlivých agentů však vede k vynoření se poznání na úrovni problému jako celku.

Výpočetní systémy inspirované chováním sociálního hmyzu jsou intenzivně zkoumány od poloviny 90. let [3]. V tomto textu se zabýváme metodami poučenými z chování mravenčích kolonií [4], které byly úspěšně použity především pro optimalizační úlohy a úlohy prohledávání, ale též např. pro analýzu počítačových sítí nebo dobývání znalostí.

© Václav Snášel (Ed.): Znalosti 2008, pp. 291–302, ISBN 978-80-227-2827-0.
FIIT STU Bratislava, Ústav informatiky a softvérového inžinierstva, 2008.

V dobývání znalostí se často toto paradigma využívá pro shlukování dat. Pro úlohy klasifikační byla od roku 2001, kdy se objevil první návrh algoritmu Ant-Miner [9], navržena celá rodina algoritmů z tohoto návrhu vycházejících [5, 8, 10].



V této studii jsme si dali za cíl zjištění vhodnosti tohoto algoritmu pro řešení reálných úloh. V dosud publikovaných článcích byl totiž tento algoritmus použit pouze pro klasifikaci malých souborů dat, z nichž největší obsahoval necelých 700 příkladů. Ve všech dosavadních studiích též chybí diskuse časové náročnosti. Proto jsme se zaměřili na velké datové množiny a na analýzu časové náročnosti. Navrhujeme navíc nové kriterium kvality pravidel, které u většiny testovaných datových sad zvyšuje přesnost klasifikace. V následující kapitole nejprve popíšeme hlavní myšlenku, z níž tato skupina algoritmů vychází a poté v kapitole 3 podrobněji původní algoritmus Ant-Miner. Nové kriterium kvality pravidel a další modifikace původního algoritmu a implemetace nového systému Ant-Miner$^+$ jsou popsány v kapitole 4. Kapitola 5 přináší popis datových souborů v experimentech použitých. Kapitola 6 uvádí popis experiemntů. Jejich výsledky jsou obsahem následujících kapitol. Analýza časové náročnosti implementace GUI-Ant-Miner je popsána v kapitole 7. Porovnání nové implementace Ant-Miner$^+$ s původním GUI Ant-Minerem a porovnání Ant-Mineru$^+$ s jinými učícími algoritmy přináší kapitola 8. Kapitola 9 se věnuje diskusi výsledků a závěrečná kapitola 10 shrnuje hlavní výsledky a uvádí směry další práce.

2 Umělé kolonie mravenců pro klasifikaci

Systém, jehož popisu je věnována tato práce, se inspiroval mravenčí kolonií a jejím způsobem lokalizace zdrojů - potravy nebo stavebního materiálu. V takové kolonii jsou veškeré nutné informace sdělovány doslova za pochodu, bez potřeby jakéhokoliv řídícího centra a dokonce bez přímé komunikace na úrovni jedinců. Hlavními prvky převzatými ze skutečné mravenčí kolonie jsou mravenci-agenti budující cestičky (v našem případě ve formě klasifikačních pravidel) a jejich schopnost zanechávat na těchto cestičkách během návratu do mraveniště – byla-li jejich výprava úspěšná – feromonové stopy umožňující vznik pozitivní zpětné

vazby. Ta způsobí v konečném důsledku hromadění mravenců ve frekventovaných koridorech spojujících mraveniště s nějakým výhodným zdrojem potravy nebo stavebního materiálu. V našem případě bude tímto zdrojem skupina učících příkladů jedné třídy a cesta k ní se stane součástí klasifikátoru jako jedno z klasifikačních pravidel.

Jak bylo řečeno, úspěšní mravenci při návratu do mraveniště značí svou cestu feromonem. Ti, kteří našli kratší cestu, a tedy lepší řešení, se po této cestě vydávají častěji a i jejich feromonová stopa je silnější než u jejich méně úspěšných spolubydlících. Postupně tak přibývá mravenců, kteří jdou po této silnější stopě. Je-li touto stopou cesta mezi různými termy $< atribut >=< hodnota >$, vytváří tak klasifikační pravidlo.

3 Algoritmus Ant-Miner

3.1 Vstupy a výstupy

Vstupem algoritmu (viz obr. 2) je učící množina ve formě relace, kde řádek odpovídá jednomu příkladu a sloupce uvádějí hodnoty jednotlivých atributů. Jeden z atributů označuje třídu klasifikace a příklady jsou klasifikovány do konečného počtu tříd. Ant-Miner generuje na základě učících dat klasifikátor, který je tvořen uspořádaným seznamem pravidel typu IF <podmínky> $THEN$ <třída>. Každé pravidlo má tvar

$$\text{IF } atribut_1 = hodnota_1 \wedge ... \wedge atribut_n = hodnota_n \text{ THEN <třída>} ,$$

kde n je počet atributů učících příkladů. Protože první získané pravidlo bylo naučeno na kompletní učící množině a každé další na menší a menší množině, relevance ostatních pravidel klesá společně s jejich pořadím. To se odráží ve frekvenci jejich využití, protože při klasifikaci jsou na každý příklad aplikována klasifikační pravidla v tom pořadí, v jakém byla získána. Predikovanou třídou se pak stává třída pravidla, které jako první pokrylo daný příklad. Před samotným spuštěním systému musí být pro kontrolu průběhu zmíněných hlavních cyklů určeny hodnoty následujících parametrů:

Max-počet-nepokrytých-příkladů – maximální povolený počet učících příkladů nepokrytých nalezenými pravidly;

Min-pokrytí-pravidla – minimální počet učících příkladů, jenž musí být pokryt každým kandidátním pravidlem;

Počet-mravenců – maximální počet mravenců (tj. různých kandidátních pravidel) v rámci jednoho průchodu vnějším cyklem;

Max-stejných-pravidel – maximální povolený počet mravenců, kteří dospěli ke shodnému pravidlu (v rámci jednoho průchodu vnějším cyklem).

Generický algoritmus je na obr 2. Různé implementace se potom budou lišit v kurzívou vysázených operacích.

```
UčícíMnožina = { všechny učící příklady };
Spočti množství informace v každém termu atribut=hodnota;
NaučenáPravidla = [];
WHILE (card(UčícíMnožina) ≥ Max-počet-nepokrytých-příkladů)
   i=1; /* index mravence */
   N_konverg=1; /* pro test konvergence*/
   Inicializuj cesty stejným množstvím feromonu;
   REPEAT
      Vytvoř pravidlo P_i ;
      Prořež pravidlo P_i ;
      Aktualizuj podle P_i feromonové stopy ;
      IF (P_i = P_{i-1}) THEN N_konverg++ ELSE N_konverg=1 ;
      i++
   UNTIL i ≥ Počet-mravenců OR N_konverg ≥ Max-stejných-pravidel;
   Vyber nejlepší pravidlo z P_i;
   Přidej toto pravidlo do seznamu NaučenáPravidla;
   Modifikuj učící množinu
   /* např. zruš všechny příklady pokryté tímto pravidlem */ ;
END WHILE
```

Obrázek 2. Ant-Miner (podle [9])

3.2 Učení a prořezávání pravidel

Pro všechny termy $< atribut >=< hodnota >$ je nejdříve odhadnuta jejich diskriminační schopnost jako množství informace svázané s jednotlivými hodnotami atributů učící množiny měřené pomocí entropie [6, 12]. Množství feromonu je na počátku stejné u všech hodnot všech atributů.

Kostru učicího algoritmu tvoří dvojice vnořených cyklů. Vnitřní cyklus je zodpovědný za konstrukci kandidátních pravidel, ze kterých je při každé iteraci vnějšího cyklu vybráno právě jedno pravidlo finální, jež se stává součástí klasifikátoru. Vnější cyklus probíhá tak dlouho, dokud není finálními pravidly pokryta požadovaná část učící množiny.

Ve vnitřním cyklu každý z mravenců postupně vytvoří jedno kandidátní pravidlo. Pravidla jsou budována inkrementálně (term po termu) formou pravděpodobnostního výběru ovlivněného množstvími informace a feromonu příslušejícími jednotlivým atributovým hodnotám. Práce mravence je u konce, jakmile je každému z atributů přiřazena nějaká hodnota (úplné kandidátní pravidlo) nebo klesne-li pokrytí pravidla pod stanovenou hodnotu Min-pokrytí-pravidla (částečné kandidátní pravidlo). Třídou pravidla je třída s většinovým zastoupením mezi případy učící množiny, které jsou pravidlem pokryty.

Hotové kandidátní pravidlo (ať již úplné, či neúplné) je poté prořezáno, aby se zabránilo přeučení. Prořezání pravidla je postupem opačným k jeho budování: termy jsou po jednom z upravovaného pravidla odstraňovány tak dlouho, dokud se zvyšuje jeho kvalita Q. V každém kroku prořezání je odebrán právě ten term, jehož odstranění zvýší kvalitu pravidla nejvíce. Protože vlivem odstranění některých termů může dojít ke změně třídy pravidla, je potřeba před zjištěním kvality prořezaného pravidla znovu určit jeho třídu. Výsledné prořezané kandidátní pravidlo je porovnáno s předchozím nalezeným kandidátním pravidlem, aby mohla být posouzena pravidlová konvergence.

Po vystoupení z vnitřního cyklu je z množiny nalezených kandidátních pravidel vybráno kandidátní pravidlo o nejvyšší kvalitě, přidáno do výsledného klasifikátoru a příklady jím pokryté jsou odstraněny z učící množiny. Neklesl-li počet příkladů v učící množině pod hodnotu řídící proměnné Max-počet-nepokrytých-příkladů, je po reinicializaci systému (tj. rovnoměrná redistribuce feromonové stopy a vypočtení diskriminační schopnosti pro hodnoty atributů, jenž se i nadále vyskytují v příkladech učící množiny) zahájena nová iterace vnějšího cyklu. V opačném případě je budování klasifikátoru u konce.

4 Ant-Miner$^+$

Naše implementace zachovává základní schéma algoritmu, rozšiřuje jej však o nové kriterium kvality a odstraňuje některé chyby původního návrhu.

4.1 Kriterium kvality

Původní kriterium kvality Q naučeného pravidla [9] je v systému Ant-Miner definováno takto:

$$Q = (truePos/(truePos + falseNeg)) * (trueNeg/(falsePos + trueNeg)),$$

kde truePos (falsePos) je počet příkladů pokrytých pravidlem, t.j. platí pro ně podmínka pravidla a patřících (nepatřících) do třídy predikované pravidlem, falseNeg (trueNeg) je potom počet příkladů nepokrytých pravidlem a patřících (nepatřících) do třídy predikované pravidlem. Při implementaci původního návrhu však několikrát došlo k situaci, kdy se systém choval nevhodně či přímo nekorektně. Stalo se tak sice při práci s velmi malými datovými množinami (maximálně 7 případů), výskyt podobných případům však nelze vyloučit u libovolně velkých datových množin.

Je-li hodnota entropie libovolné atributové hodnoty maximální možná, rovná se z ní odvozená hodnota pro výběr termu $< atribut >=< hodnota >$ nule. Takových termů se v učící množině může obecně vyskytnout mnoho. V krajním případě se to může týkat všech termů, což by zcela znemožnilo pokračování ve výpočtech. Tyto nulové hodnoty jsou proto v naší implementaci nahrazeny malým kladným nenulovým číslem (0.000001).

Při výpočtu kvality pravidla podle původního návrhu navíc může dojít k dělení nulou, a to pokud jsou hodnoty trueNeg a falsePos obě rovny nule. Ve zmíněných případech je vzorec kritéria kvality pravidla zjednodušen na

Q = (truePos/(truePos + falseNeg)) * (1/(falsePos + 1))

V návrhu zmíněné kritérium kvality pravidel nemusí být pro potřeby uživatele vyhovující. Zvláště hodné zamyšlení je využití extrémně nízkého počtu atributů při budování klasifikačních pravidel - často bývá v pravidle využit pouze jeden atribut. V systému je proto implementováno druhé kritérium kvality

Q2 = (1/numExamples)*(truePos/(falsePos + 1))

4.2 Konstrukce defaultního pravidla

Při vybírání třídy defaultního pravidla - tj. pravidla, které se použije, pokud nebylo použito žádné z naučených - může nastat situace, kdy je učící množina prázdná (jakkoliv je nastaven parametr Max-počet-nepokrytých-příkladů, poslední nalezené pravidlo může pokrýt všechny zbývající případy učící množiny). Za těchto podmínek nelze užít postupu, při kterém je za třídu prázdného pravidla prohlášena třída s nejširším zastoupením ve zbytku učící množiny, i tehdy ale může být prázdné pravidlo nezbytné. Problému lze předejít určením nejčastěji se vyskytující třídy před nalezením prvního klasifikačního pravidla. Pokud je i po nalezení posledního klasifikačního pravidla učící množina neprázdná, je proměnná uchovávající hodnotu nejčastěji se vyskytující třídy přepsána dle původního návrhu.

4.3 Implementace

Ant-Miner$^+$ je implementován v javě. Systém se skládá ze čtyř tříd:

`Centre` je centrální třídou řídící běh systému a pořadí všech jeho výpočtů; obsahuje veškeré proměnné s nastavitelnými prametry, metodu main, v jejímž těle jsou implementovány hlavní cykly původního návrhu, a výsledný klasifikátor.

`Ant` je třídou reprezentující jedince mravenčí kolonie a jejich kandidátní pravidla. Metody související přímo s budováním či prořezáváním klasifikačních pravidel jsou volány prostřednictvím této třídy.

Třída `DataFiller` obstarává načítání dat ze zdrojových souborů a jejich převod do vnitřních datových struktur. Všechny výpočty systému, byť volané i z jiných tříd, jsou prováděné uvnitř metod třídy `DataFiller`.

`ACSException` je třídou pro objekty typu výjimka. Vypořádává se s chybami, ke kterým může dojít v průběhu práce systému.

Při jednom spuštění systému dojde k vytvoření právě jednoho objektu třídy `Centre` a právě jednoho objektu třídy `DataFiller`. Počet objektů třídy `ACSException` (Ant System Colony Exception) je závislý na počtu chyb, ke kterým dojde během práce systému (ideálně tedy nula, jinak jedna). Počet objektů třídy `Ant` (tj. počet mravenců) podílejících se na výpočtech v rámci jednoho běhu systému je závislý na vstupních datech, hodnotách parametrů systému a v neposlední řadě také průběžných výsledcích jeho výpočtů. Jejich počet není nijak omezen.

5 Datové soubory

V experimentech byly použity následující datové soubory z UCI Repository [2]:

Breast Cancer Wisconsin	699 příkladů, 9 spojitých atributů
King-rook-vs-king-pawn	959 příkladů, 36 nominálních atributů (náhodně vybráno z celkových 3196 příkladů)
Yeast	1484 příkladů, 7 spojitých atributů
Waveform21	5000 příkladů, 19 spojitých atributů

Wisconsin Breast Cancer Data jsme zvoili proto, že stejná data byla použita ve všech předchozích studiích o Ant-Mineru. Velké datové soubory *King-rook-vs-king-pawn, Yeast, Waveform21* byly vybrány tak, aby se lišily jak v počtu příkladů (od 959 do 5000[1]) tak v počtu spojitých a nominálních atributů. Protože Ant-Miner umí pracovat jen s nominálními atributy, byly všechny spojité nejdřív diskretizovány. Doména každého spojitého atributu byla rozdělena na intervaly stejné délky (Weka [1], metoda Discretize, defaultní nastavení).

6 Experimenty

Cílem experimentů bylo zjistit

1. závislost doby učení, přesnosti klasifikace a velikosti klasifikátoru (velikosti množiny pravidel a délky těchto pravidel) na parametrech systému a
2. použitelnost Ant-Mineru pro velké datové soubory

Pro první cíl jsme použili implementaci GUI Ant-Miner, kterou vytvořili autoři původního návrhu [9] a která je volně dostupná na `https://sourceforge.net//projects/guiantminer`. Souhrn výsledků najde čtenář v kapitole 7, úplné výsledky pak v příloze.

Protože GUI Ant-Miner nedovoluje zpracovávat větší datové soubory, pro odpověď na druhou otázku jsme použili naši implementaci Ant-Miner$^+$, Výsledky a porovnání s dalšími učicími algoritmy jsou náplní další kapitoly.

Všechny výsledky byly získány 10tisložkovou křížovou validací. Kriteriem kvality byla přesnost, tj. poměr počtu správně klasifikovaných příkladů ku všem příkladům, přičemž každý příklad byl klasifikátorem klasifikován do právě jedné třídy.

7 Analýza časové náročnosti GUI Ant-Mineru

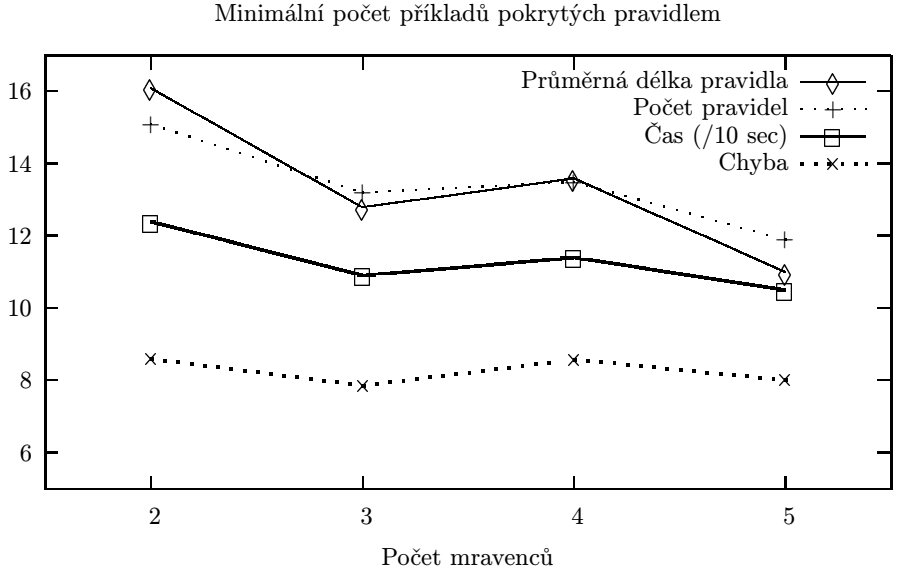
Pro tuto analýzu jsme zvolili data Wisconsin Breast Cancer, protože byla použita ve všech předchozích studiích o algoritmu Ant-Miner [5, 8–10]. Defaultní nastavení GUI Ant-Mineru bylo

[1] Horní hranici 5000 příkladů jsme zvolili kvůli velké časové náročnosti učení.

Počet-mravenců = 5
Min-pokrytí-pravidla = 5
Max-počet-nepokrytých-příkladů = 10
Max-stejných-pravidel = 10

Implementace GUI Ant-Miner se ukázala jako velice nestabilní. Pro defaultní nastavení výpočet proběhne bez potíží, při jiném však cyklí nebo havaruje. To nastalo asi v 30% běhů.

Navíc i když datová sada Wisconsin Breast Cancer byla použita pro experimenty ve všech předchozích publikovaných článcích, není rozhodně vhodná. Např. přesnost učení nezávisí na velikosti kolonie mravenců, dokonce pro dva mravence dává přijatelný výsledek.



Obrázek 3. Závislost na minimálním pokrytí

S rostoucím počtem mravenců roste délka doby učení, avšak přesnost ani počet a délka pravidel se nemění. Pro kolonii 300 mravenců přesáhla doba učení 1 hodinu, aniž došlo k zvýšení přesnosti.

Zajímavá je však závislost na pokrytí neboli síle pravidel, tj. na minimálním počtu příkladů pravidlem pokrytých (obr. 3). Především se změnou pokrytí přesnost zůstává stejná. Se zvyšujícím minimiálním pokrytím však Ant-Miner generuje jednodušší hypotézu. Klesá totiž jak počet pravidel - z 15ti na 12 - tak jejich délka - z 16ti na 11. Přitom se dokonce zkracuje doba učení.

V závislosti na Max-stejných-pravidel se stále přesnost nemění. S klesajícím parametrem však rychle klesá doba učení. Počet a složitost pravidel zůstává téměř stejná.

Nevhodnost datové sady Wisconsin Breast Cancer je nejlépe vidět na závislosti na maximálním počtu nepokrytých příkladů Přesnost učení se výrazně nemění až do hodnoty 250 nepokrytých příkladů. Přitom klesá výrazně počet a délka pravidel a doba učení.

8 Výsledky experimentů s Ant-Miner$^+$

V této kapitole popíšeme experimenty s novou verzí algoritmu s upraveným kriteriem kvality. Porovnáme též přesnost učení tohoto algoritmu s výsledky často používaných algoritmů strojového učení.

Jak je vidět z tab. 1, nové kriterium kvality naučených pravidel zvýšilo přesnost učení u tří se čtyř datových souborů, jen u Yeast došlo ke snížení. Cena,

Tabulka 1. Srovnání výsledků pro původní a nové kriterium kvality

	Původní kriterium kvality			Nové kriterium kvality		
	Přesnost	Pravidel	Čas	Přesnost	Pravidel	Čas
Wisc.BC	92.3%	12	29vt	94.0%	24	1min
KRKS	82.4%	9	15min	93.3%	33	45min
Wav21	70.0%	14	1h	74.5%	413	2 h
Yeast	41.4%	15	15vt	35.0%	70	14min

Tabulka 2. Srovnání přesnosti pro různé učicí algoritmy

	Baseline	Ant-Miner$^+$	J48	NB	AdaBoost	DTable
Wisc.BC	65.5%	94.0%	94.1%	97.3%	95.0%	95.7%
KRKS	52.2%	82.4%	98.8%	87.5%	93.1%	96.2%
Waveform21	33,9%	74.5%	76.7%	81.0%	70.4%	73.2%
Yeast	31.2%	35.0%	59.1%	59.1%	40.7%	57.4%

kterou za to zaplatíme, je větší počet naučených pravidel a z toho vyplývající větší časová náročnost.

Druhá část experimentů se týkala porovnání Ant-Mineru$^+$ s jinými učícími algoritmy. V tab. 2 sloupec Baseline značí přesnost při klasifikaci do nejpočetnější třídy. Pro srovnání jsme použili implementace algoritmu rozhodovacích stromů [12], naivní bayesovský klasifikátor (NB) [6], AdaBoost a rozhodovací tabulky

(DTable), vše v implementaci Weka [1]. Ant-Miner$^+$ nikde nedosáhl nejvyšší přesnosti a jen u Waveform21 neskončil jako poslední.

9 Diskuse

Velkou výhodou algoritmů z rodiny Ant-Mineru je snadné ovládání. Všechny čtyři parametry – velikost kolonie mravenců, minimální počet příkladů pokrytých pravidlem, minimální počet mravenců pro přijetí pravidla a maximální počet příkladů nepokrytých žádným pravidlem – jsou velmi intuitivní. Nepominutelný je též psychologický efekt. Představa kolonie velmi jednoduchých agentů, kteří jsou schopni řešit složitý problém, aniž známe přesně jejich chování, je vzrušující.

Na druhou stranu má tento přístup – navzdory šesti letům od své první implementace – řadu nedostatků. Tím hlavním je časová náročnost. Pro data waveform21 trvalo učení (10tisložková křížová validace) 2 hodiny. Více jsme se tomuto problému věnovali v kapitole 7. Na velkých učících množinách se neprokázalo tvrzení z [9] o tom, že přesnost učení je vyšší než u jiných učících algoritmů.

10 Závěr

Nové kriterium kvality vede ke zvýšení přesnosti učení za cenu vyšší časové náročnosti. Bez ohledu na tento výsledek však ani algoritmus Ant-Miner$^+$ není kompetitivní na větších datových souborech s jinými učícími algoritmy, a to především kvůli dlouhé době učení. Jak však vyplývá z analýzy v kapitole 7, tuto dobu je možné při vhodném nastavení parametrů výrazně zkrátit bez zhoršení přesnosti učení. Tento proces je možné automatizovat použitím metalearningu. Algoritmus se dá snadno modifikovat na any-time algoritmus a především jeho výpočet distribuovat. V [5] je popsána paralelizovaná varianta tohoto algoritmu. Kromě tohoto přístupu je ovšem možné distribuovat kolonii mravenců, distribuovat data horizontálně nebo vertikálně a využít hlasovacích metod (např. comittee-based learning).

Poděkování

Autoři děkují členům Semináře z dobývání znalostí na Fakultě informatiky MU a recenzentům za podnětné připomínky. Tato práce byla částečně podpořena Fakultou informatiky MU Brno.

Reference

1. Weka `http://www.cs.waikato.ac.nz/~ml/weka/`.
2. C. L. Blake and C. J. Merz: UCI Repository of machine learning databases, `http://www.ics.uci.edu/ mlearn/MLRepository.html`. University of California, Irvine, Dept. of Information and Computer Sciences.

3. E. Bonabeau, M. Dorigo, G. Theraulaz: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press 1999.
4. M. Dorigo et al. (eds.): Proceedings of 5th Intl. Workshop on Ants Colony Optimization and Swarm Intelligence, ANTS 2006. LNCS 4150, Springer Verlag 2006.
5. Y. Chen, L. Chen, and L. Tu: Parallel Ant Colony Algorithm for Mining Classification Rules. IEEE International Conference on Granular Computing (GrC06), 2006.
6. T. M. Mitchell: Machine Learning. McGraw Hill 1997.
7. Návrat P. et al.: Vyhľadávanie informácií pomocou včiel. Sborník konference Znalosti 2007, Ostrava 2007, str. 63-74.
8. B. Liu, H. A. Abass, B. McKay: Classification Rule DIscovery with Ant Colony Optimization. IEEE Computational Intelligence Bulletin, Vol.3 No.1, February 2004.
9. R.S.Parpinelli, H.S.Lopes, A.A.Freitas: An Ant Colony Based System for Data Mining: Application to Medical Data. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 791–797, 2001.
10. R.S. Parpinelli, H.S. Lopes, and A.A. Freitas: Classification-rule discovery with an ant colony algorithm. In M. Khosrow-Pour, editor, Encyclopedia of Information Science and Technology, pp. 420-424. Idea Group, Hershey, January 2005.
11. L. Qin, Y. Chen, L. Chen, and Y. Yuan: A New Optimization Algorithm Based on the Ant Colony System with Density Control Strategy. Third International Symposium on Neural Networks (ISNN 2006), 2006.
12. J. R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann 1993.

Přiloha: Výsledky GUI Ant-Mineru pro data Wisconsin Breast Cancer

1. Počet mravenců

No. ants	Accuracy Rate	No. Rules	No. Conditions	Time
2	92,56% +/- 0,92%	12,0 +/- 0,21	11,1 +/- 0,28	71
3	91,56% +/- 0,81%	12,5 +/- 0,17	11,6 +/- 0,16	93
4	92,42% +/- 1,11%	12,5 +/- 0,17	11,5 +/- 0,17	113
5	91,99% +/- 1,3%	11,9 +/- 0,18	11,0 +/- 0,15	105
6	92,28% +/- 1,23%	12,3 +/- 0,26	11,4 +/- 0,34	128
7	92,13% +/- 1,05%	12,0 +/- 0,21	11,1 +/- 0,28	117
8	91,84% +/- 0,78%	11,7 +/- 0,3	10,8 +/- 0,29	127
9	92,56% +/- 1,08%	12,0 +/- 0,21	11,3 +/- 0,21	146
10	92,13% +/- 0,93%	12,3 +/- 0,26	11,6 +/- 0,34	147

2. Minimální počet příkladů pravidlem pokrytých

Min.cases	Accuracy Rate	No. Rules	No. Conditions	Time
2	91,41% +/- 0,65%	15,1 +/- 0,43	16,1 +/- 0,81	124
3	92,13% +/- 0,57%	13,2 +/- 0,39	12,8 +/- 0,61	109
4	91,42% +/- 1,38%	13,5 +/- 0,34	13,6 +/- 0,64	114
5	91,99% +/- 1,3%	11,9 +/- 0,18	11,0 +/- 0,15	105

3. Maximální počet stejných pravidel

Rules f.c	Accuracy Rate	No. Rules	No. Conditions	Time
2	92,99% +/- 0,75%	12,2 +/- 0,33	11,2 +/- 0,33	26
4	91,71% +/- 1,02%	12,1 +/- 0,18	11,2 +/- 0,2	63
6	91,69% +/- 1,37%	12,4 +/- 0,22	11,7 +/- 0,3	75
8	92,13% +/- 0,57%	12 +/- 0,15	11,1 +/- 0,23	95
10	91,99% +/- 1,3%	11,9 +/- 0,18	11,0 +/- 0,15	105

4. Maximální počet nepokrytých příkladů

Uncovered	Accuracy Rate	No. Rules	No. Conditions	Time
10	91,99% +/- 1,3%	11,9 +/- 0,18	11,0 +/- 0,15	105
12	92,56% +/- 1%	11,7 +/- 0,26	11,0 +/- 0,37	103
14	92,56% +/- 0,97%	11,6 +/- 0,22	10,7 +/- 0,26	104
16	91,7% +/- 0,9%	11 +/- 0,21	10,1 +/- 0,28	130
18	92,13% +/- 1,13%	10,9 +/- 0,28	10,3 +/- 0,45	121
20	92,27% +/- 0,94%	10,8 +/- 0,13	10,0 +/- 0,15	107
30	92,12% +/- 1,24%	9,2 +/- 0,13	8,3 +/- 0,15	93
50	92,13% +/- 1,69%	7,3 +/- 0,15	6,4 +/- 0,16	84
100	93,42% +/- 1,05%	5,0 +/- 0	4,2 +/- 0,13	68
200	93,13% +/- 0,76%	4,0 +/- 0	3,2 +/- 0,13	58
250	93,27% +/- 0,93%	3,0 +/- 0	2,2 +/- 0,13	54
300	87,55% +/- 1,57%	2,0 +/- 0	1,1 +/- 0,1	52

Annotation:

How An Ant Colony Mines Data

We discuss several drawbacks of Ant-Miner, a learning system that was inspired by behavior of social insects. First we bring experimental analysis of time complexity of the original algorithm. We show that Wisconsin Breast Cancer data that was used in most of experiments is not appropriate. Then we describe a new version, Ant-Miner$^+$ and also a new quality criterion. We compare performance of Ant-Miner with Ant-Miner$^+$ and Ant-Miner$^+$ with other learning algorithms.