Učenje odločitvenih pravil z evolucijsko optimizacijo

Matej Pičulin

doktorska disertacija
predana
Fakulteti za računalništvo in informatiko
kot del izpolnjevanja pogojev za pridobitev naziva
doktor znanosti
s področja
računalništva in informatike

Ljubljana, 2018

IZJAVA

Izjavljam, da sem avtor dela in da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali na drugem visokošolskem zavodu, razen v primerih, kjer so navedeni viri.

— Matej Pičulin —

september 2018

Oddajo so odobrili

dr. Igor Kononenko
redni profesor za računalništvo in informatiko
predsednik ocenjevalne komisije

dr. Marko Robnik-Šikonja
redni profesor za računalništvo in informatiko
član ocenjevalne komisije in mentor

dr. Mitja Luštrek
višji znastveni sodelavec
zunanji član ocenjevalne komisije
Institut Jožef Stefan

PREDHODNA OBJAVA

Izjavljam, da so bili rezultati obravnavane raziskave predhodno objavljeni/sprejeti za objavo v recenzirani reviji ali javno predstavljeni v naslednjih primerih:

[1] M. Pičulin, and M. Robnik-Šikonja. Handling numeric attributes with ant colony based classifier for medical decision making. Expert Systems with Applications, 41(16):7524-7535, 2014.

Potrjujem, da sem pridobil pisna dovoljenja vseh lastnikov avtorskih pravic, ki mi dovoljujejo vključitev zgoraj navedenega materiala v pričujočo disertacijo. Potrjujem, da zgoraj navedeni material opisuje rezultate raziskav, izvedenih v času mojega podiplomskega študija na Univerzi v Ljubljani.

Doktorski študij je delno sofinancirala Evropska unija, in sicer iz Evropskega socialnega sklada. Sofinanciranje se izvaja v okviru Operativnega programa razvoja človeških virov za obdobje 2007-2013, 1. razvojne prioritete Spodbujanje podjetništva in prilagodljivosti; prednostne usmeritve_1. 3: Štipendijske sheme.

University *of Ljubljana*
Faculty *of Computer and Information Science*
Matej Pičulin
Learning decision rules with evolutionary optimization

ABSTRACT

One of successful predictive and descriptive approaches in machine learning is decision rule learning. Decision rules achieve reasonable classification accuracy and are interpretable, which is important to end users, who trust predictions more if they are supported with explanations. The challenge in mining decision rules is to find a short and comprehensible rule list with high classification accuracy. This led to many different types of classification rules like crisp rules, soft rules, probabilistic rules, etc.

We developed two new methods for mining classification rules based on ant colony optimization, which is a successful discrete optimization method. In the first part of the dissertation, we present a new method called nAnt-Miner, which can, contrary to most other ant colony based approaches, handle numeric attributes. This leads to an increased search space and affects the running time and use of memory. We showed that the nAnt-Miner method is comparable to other ant colony optimization based rule learning methods, but is worse than fuzzy rules based method FURIA. The advantage of the nAnt-Miner method is that it can detect strong dependencies between attributes.

In the second part of the dissertation we present the ProAnt-Miner method, which mines probabilistic rules. We introduce a novel interpretation of pheromone values for this approach. ProAnt-Miner is faster, achieves better prediction accuracy than nAnt-Miner, and uses less memory due to a different search graph. We showed that the ProAnt-Miner classification accuracy does not statistically differ from the state-of-the-art methods like FURIA and RIPPER. The ProAnt-Miner method has new rule form, which can give the user new insights. We evaluated both methods on real and artificial datasets.

Key words ant colony optimization, evolutionary computation, machine learning, rule learning, probabilistic rules, soft rules

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*
Matej Pičulin
Učenje odločitvenih pravil z evolucijsko optimizacijo

POVZETEK

Učenje pravil je eno od uspešnih napovednih in opisnih metod strojnega učenja. Pravila dosegajo solidno klasifikacijsko točnost in so razložljiva, kar je pomembno za končne uporabnike, ki napovedim z razlago bolj zaupajo. Izziv pri iskanju odločitvenih pravil je dobiti kratke in razumljive sezname pravil z visoko klasifikacijsko točnostjo. To je vodilo do razvoja mnogih različnih oblik klasifikacijskih pravil, kot so trda pravila, mehka pravila, verjetnostna pravila itd.

Razvili smo dve metodi za iskanje odločitvenih pravil z uporabo optimizacije s kolonijo mravelj, ki je uspešna metoda za diskretno optimizacijo. V prvem delu disertacije predstavimo novo metodo imenovano nAnt-Miner, ki, za razliko od večine drugih metod, osnovanih na koloniji mravelj, obravnava tudi številske atribute. To vodi do večjega preiskovalnega prostora in vpliva na čas izvajanja ter porabo pomnilnika. Pokazali smo, da je metoda nAnt-Miner primerljiva z ostalimi metodami na osnovi kolonije mravelj, vendar je slabša od metode FURIA za iskanje mehkih pravil. Prednost metode nAnt-Miner je v tem, da lahko zazna močne odvisnosti med atributi.

V drugem delu disertacije predstavimo metodo ProAnt-Miner, ki išče verjetnostna pravila. Predstavimo novo interpretacijo feromonov za delovanje te metode. Metoda ProAnt-Miner je v primerjavi z metodo nAnt-Miner hitrejša, dosega višjo klasifikacijsko točnost in porabi manj pomnilnika, predvsem zaradi uporabe drugačnega preiskovalnega grafa. Pokazali smo, da se metoda ProAnt-Miner, glede na klasifikacijsko točnost, statistično ne razlikuje od vodilnih metod, kot sta FURIA in RIPPER. Metoda ProAnt-Miner ima novo obliko pravil, ki lahko da nov pogled na podatke. Metodi smo ovrednotili na realnih in umetnih podatkovnih množicah.

Ključne besede kolonija mravelj, evolucijsko računanje, strojno učenje, učenje pravil, verjetnosta pravila, mehka pravila

ZAHVALA

Najprej bi se rad zahvalil mojemu mentorju, rednemu profesorju Marku Robniku Šikonji, saj brez njegove pomoči in usmerjanja ter tudi potrpežljivosti to delo ne bi nastalo.

Rad bi se zahvalil vsem članom Laboratorija za kognitivno modeliranje za šproščeno vzdušje v laboratoriju in za pomoč pri delu, predvsem v prvih dveh letih.

Nenazadnje bi se zahvalil svojim staršem, ki so me podpirali na poti do disertacije.

— Matej Pičulin, Ljubljana, september 2018.

KAZALO

I

Uvod

Strojno učenje se vedno več uporablja na mnogih področjih. Marsikje se beležijo podatki o navadah ljudi in organizacij. Tako se na primer zbirajo podatki o zdravju ljudi, nakupovalnih navadah, finančnih investicijah, poteku športnih tekem, literaturi, procesih v podjetji itd. V grobem ločimo dva načina strojnega učenja, in sicer nadzorovano in nenadzorovano učenje. Pri nadzorovanem učenju želimo na označenih podatkih napovedati označbe še ne-videnih primerov, pri nenadzorovanem učenju pa iščemo zakonitosti v podatkih. Primer nadzorovanega strojnega učenja je iz nakupovalne košarice potrošnice napovedati, ali je noseča ali ne. S takimi podatki lahko podjetje potrošnikom pošilja prilagojene reklame. Primer nenadzorovanega strojnega učenja na istih podatkih je iskanje artiklov, ki se velikokrat prodajajo skupaj. To lahko podjetje izkoristi za boljšo organizacijo prodajalnih polic.

Pri prejšnjem primeru uporabnika zanima predvsem napovedna točnost algoritma strojnega učenja. V nekaterih panogah, kot na primer v medicini, pa je pomembna tudi razlaga dobljene napovedi. Ljudje velikokrat raje zaupamo odgovoru, ki ima tudi pojasnilo. Tako imajo v splošnem nevronske mreže visoko klasifikacijsko točnost, vendar jim manjka razlaga. Klasifikacijska pravila pa na drugi strani dosegajo manjšo klasifikacijsko točnost, a nudijo preprosto razlago.

Eden izmed načinov, kako se lahko naučimo klasifikacijskih pravil, je preiskovanje prostora vseh možnih seznamov pravil. Prostor vseh možnih seznamov pravil je za določeno domeno ponavadi neskončen, a ga lahko smiselno omejimo, čeprav je še prevelik, da bi lahko ocenili ali našteli vse možne sezname pravil. Za preiskovanje tega prostora seznamov uporabimo hevristike, ki nas v doglednem času vodijo do zadovoljivih rešitev. V tem delu za preiskovanje prostora odločitvenih pravil uporabljamo biološko navdihnjeno metodo kolonije mravelj, ker se je le-ta na sorodnih problemih izkazala za učinkovito. Skupina znanih metod za iskanje klasifikacijskih pravil iz podatkov na osnovi kolonije mravelj se imenuje Ant-Miner. Težave obstoječih metod Ant-Miner so pri obravnavi zveznih atributov, saj je kolonija mravelj primernejša za diskretno optimizacijo. Otero in sod. [1] so poskušali rešiti to težavo s sprotno diskretizacijo atributov. Delo je bilo povod za prvi del disertacije, v kateri smo razvili metodo nAnt-Miner, ki uporablja za preiskovanje prostora pravil kolonijo mravelj in je zmožna obravnavati tudi zvezne atribute. Cilj metode nAnt-Miner je razvoj nove metode, osnovane na metodi Ant-Miner+ [2], ki je vodilna metoda zasnovana na koloniji mravelj za iskanje trdih pravil tako, da bo odpravila pomanjkljivost metode Ant-Miner+, ki ne zna obravnavati zveznih atributov in hkrati bila še vedno statistično primerljiva glede na klasifikacijsko točnost z metodo

Ant-Miner+, pri čemer smo za diskretizacijo zveznih atributov predhodno uporabili metodo Fayyad-Irani [3].

V drugem delu disertacije smo se osredotočili na novo, verjetnostno predstavitev pravil, ki jih dobimo s kolonijo mravelj. To metodo smo poimenovali ProAnt-Miner in lahko obravnava diskretne in zvezne atribute ter sočasno gradi mehke intervale pravil. Cilj metode ProAnt-Miner je iskanje verjetnostnih pravil, ki nam lahko podajo drugačen pogled v podatke, hkrati pa želimo, da je metoda glede na klasifikacijsko točnost primerljiva z obstoječimi vodilnimi metodami, kot sta FURIA [4] in RIPPER [5].

1.1 Pregled vsebine

V poglavju 2 najprej opišemo ozadje dela, kot sta oblika klasifikacijskih pravil in mere za vrednotenje le-teh. Opišemo metode za rudarjenje pravil, ki so vodile do danes vodilnih metod za rudarjenje pravil [4, 5]. Temu sledi opis optimizacije s kolonijo mravelj, razvoj metod za iskanje pravil, osnovanih na koloniji mravelj, ter kratek pregled sorodnih raziskav. V poglavju 3 opišemo novo razvito metodo za iskanje pravil z uporabo kolonije mravelj, ki lahko obravnava diskretne in številske atribute, čeprav je kolonija mravelj osnovana za preiskovanje diskretnega prostora. V poglavju 4 predstavimo drugo razvito metodo ProAnt-Miner, ki je namenjena iskanju verjetnostnih pravil. Obe razviti metodi smo empirično ovrednotili na realnih in umetnih podatkovnih množicah. V zaključku povzamemo narejeno in predstavimo ideje za izboljšave in nadaljnje delo.

1.2 Prispevki k znanosti

V disertaciji sta predstavljena naslednja prispevka k znanosti:

- *Razvoj metode za klasifikacijo, osnovano na koloniji mravelj, ki omogoča naravno obravnavanje vseh tipov atributov.* Razvili smo metodo nAnt-Miner za učenje pravil, ki obravnava nominalne, ordinalne in številske atribute, čeprav je metoda kolonije mravelj zasnovana za diskretno optimizacijo [6]. V našem delu smo jo za učenje pravil uporabili na številskih atributih, tako da smo dinamično spreminjali graf preiskovanja. S tem dosežemo dovolj nizko prostorsko in časovno zahtevnost algoritma za praktično uporabo. Podrobnosti o metodi so v poglavju 3 in objavljene v članku [7].

- *Razvoj metode za iskanje verjetnostnih pravil z metodo kolonije mravelj.* Razvili smo metodo ProAnt-Miner, ki, tako kot metoda nAnt-Miner, obravnava številske atribute.Metoda se uči mehkih pravil Gaussovske oblike, ki jih za lažjo interpetacijo lahko spremenimo v obliko podobno trapezoidni obliki. Metoda za svoje delovanje inovativno interpretira pomen feromonov, ki hkrati usmerjajo potek preiskovanja, prestavljajo verjetnostno distribucijo vrednosti zveznih atributov in predstavljajo končni mehki interval vsakega člena pravila. Metoda je podrobneje predstavljena in ovrednotena v poglavju 4. Metoda še ni publicirana.

2

Izhodišča in sorodne raziskave

Tabela 2.1

Primer majhne podatkovne množice za klasifikacijo. Prvih pet stolpcev so atributi, razred pa predstavlja podatek, ali je oseba vrnila kredit.

Izobrazba	Zakonski stan	Spol	Otroci	Starost	Kredit
Osnovna	Samski	Moški	Ne	27	Ne
Osnovna	Samski	Moški	Da	44	Ne
Osnovna	Poročen	Moški	Ne	24	Da
Univerzitetna	Ločen	Ženski	Ne	30	Da
Univerzitetna	Poročen	Ženski	Da	33	Da
Srednja	Samski	Moški	Ne	45	Ne
Univerzitetna	Samski	Ženski	Ne	27	Ne

V strojnem učenju imamo velikokrat opravka z množico podatkov, ki so opisani z atributi in vrednostjo, ki jo želimo napovedati. Podatke najlažje predstavimo s tabelo, kjer vsak stolpec predstavlja atribut oziroma razred, vsaka vrstica pa predstavlja en učni primer, za katerega poznamo razred. Primer podatkov lahko vidimo v tabeli 2.1. Proces, ki je generiral te podatke, želimo modelirati, kar pomeni, da želimo pridobiti model, ki omogoča čim točnejše napovedovanje razreda še ne-videnih primerov. Temu procesu pravimo strojno učenje. Ločimo dve vrsti napovednih modelov, in sicer, če napovedujemo diskretno vrednost, imenujemo dobljeni model klasifikator; če pa je napovedana vrednost zvezna, ga poimenujemo regresor. V našem delu smo se osredotočili na klasifikatorje, saj razvijamo metodi, ki napovedujeta diskretne razrede.

Napovedne modele delimo tudi glede na njihov opisni jezik. V našem delu se bomo osredotočili na modele v obliki odločitvenih pravil. Drugi modeli so na primer Bayevsoki modeli, drevesni modeli ali nevronske mreže. Podrobnejša delitev modelov strojnega učenja in osnove strojnega učenja opisuje knjiga avtorjev Kononenko in Kukar [8]. Pravila ločimo na klasifikacijska in povezovalna. Za razliko od klasifikacijskih pravil pri rudarjenju povezovalnih pravil kot razred obravnavamo poljuben atribut in iščemo statistične povezave med atributi. V našem delu smo se osredotočili na klasifikacijska pravila. Področje učenja pravil je podrobno opisano v knjigi avtorjev Fürnkranz in sodelavcev [9], po katerem je zasnovano nadaljevanje tega poglavja.

V razdelku 2.1 predstavimo možne oblike klasifikacijskih pravil, v razdelku 2.2 podamo mere, ki se uporabljajo za vrednotenje pravil, v razdelku 2.3 pa predstavimo obstoječe

metode za učenje pravil. V razdelku 2.4 opišemo metodo optimizacije s kolonijo mravelj, v razdelku 2.5 pa predstavimo dosedanje metode za učenje pravil na osnovi optimizacije s kolonijo mravelj.

2.1 Oblika klasifikacijskih pravil

Klasifikacijska pravila največkrat zapišemo v treh oblikah, in sicer eksplicitni, formalni in obliki logike prvega razreda v zapisu jezika prolog.

Eksplicitna oblika: Primer eksplicitnega pravila, ki ga lahko izluščimo iz podatkov tabele 2.1 je IF Spol = Moški AND Starost > 25 THEN Kredit = Ne. To pravilo pokrije prvi, drugi in šesti primer iz tabele ter napoveduje razred Kredit z vrednostjo Ne. Takšen zapis pravila je za splošnega uporabnika preprosto razumljiv.

Formalna oblika: Formalna oblika pravil izhaja iz formalne logike. V formalni logiki lahko zapišemo primer, ki smo ga uporabili pri eksplicitni obliki kot:
Ne $\leftarrow$ Spol = Moški $\wedge$ Starost > 25.

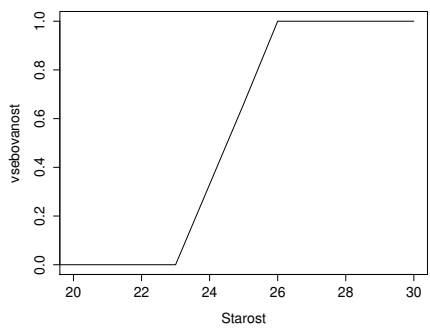
Pravila v obliki jezika prolog: Pogosto srečamo zapis, ki se uporablja v programskem jeziku prolog. V tem primeru bi bilo naše pravilo zapisano kot:
negative(X) :- Spol(X, moški), Starejsi(X, 25), kjer je predikat Starejsi(X, Y) definiran kot Starejsi(X, C) :- Starost(X, S), S > C.

Obliko pravil smo prikazali na podatkih iz tabele 2.1. Tabela vsebuje pet atributov in razred. Pravila bomo v nadaljevanju prikazovali v eksplicitni obliki, ki so sestavljena iz konjunkcije pogojev, vezanih z AND v delu pravila IF, in iz zaključka v delu pravila THEN. Ta prikaz bomo uporabljali zaradi njegove preprostosti in razumljivosti. Pravilo IF Spol = Moški AND Starost > 25 THEN Kredit = Ne pokrije tri učne primere in ima na podani učni množici 100% klasifikacijsko točnost, saj pravilno napove vse primere iz tabele 2.1, za katere velja pogojni del pravila. Če v tem pravilu člen Starost > 25 zamenjamo s členom Starost > 26, ima to pravilo ravno tako 100% točnost, saj je pri numeričnih atributih marsikdaj težko določiti ostro mejo, kjer pravilo še drži in kje preneha veljati. Taka pravila imenujemo trda pravila, ker tvorijo trde meje med razredi. Znana in uspešna sistema za rudarjenje trdih pravil sta CN2 [10] in RIPPER [5].

Omejitve trdih pravil omilimo z mehkim pravilom oziroma mehkim intervalom, kot je prikazan na sliki 2.1. Če desni interval poimenujemo Starejši, dobimo pravilo IF Spol = Moški AND Starost = Starejši THEN Kredit = Ne. To pravilo

klasificira vse moške starejše od 26 let v razred Ne, tiste s starostjo med 23 in 26 let pa bi v razred Ne klasificiralo z verjetnostjo, podano z narisano daljico. Mehki intervali so lahko omejeni z obeh strani in imajo takrat trikotno oziroma trapezoidno obliko. Ena uspešnejših metod za rudarjenje mehkih pravil je FURIA [4], njena podrobnejša analiza je objavljena v članku avtorjev Hühn in Hüllermeier [11].

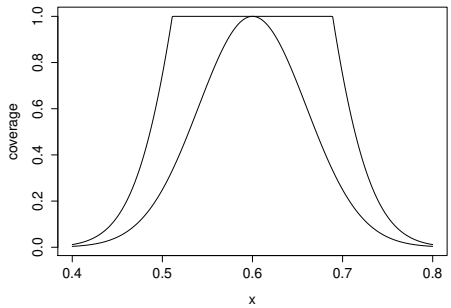
V tem delu prikažemo tudi novo obliko pravil in intervalov, osnovano na normalni distribuciji in prikazano na sliki 2.2, ki je podrobneje opisana v razdelku 4.1. Taka pravila so verjetnostna pravila, saj vsako pravilo z določeno verjetnostjo glasuje za določen razred. Intervali so podobni trapezoidnim mehkim intervalom in popolnoma pokrijejo primere blizu vrednosti 0,6, primere pod 0,5 in nad 0,7 pa le z določeno verjetnostjo.



Slika 2.1
Primer mehkega intervala.

2.2 *Mere za vrednotenje pravil*

Vsako zgrajeno pravilo pokriva določeno število primerov. Tukaj se omejimo na binarni razred oziroma na pozitivne in negativne primere. Privzamemo, da želimo s pravili pokriti pozitivne primere. Naučiti se želimo seznama pravil, ki pokrijejo celotno učno množico ter se dobro posplošijo na še ne videne primere. Da pravilno pokrijemo čim več primerov, optimiziramo naslednja kriterija:



Slika 2.2
Primer verjetnostnega intervala.

- popolnost: število pozitivnih primerov, ki jih pravila pokrijejo, mora biti čim večje;
- konsistentnost: število negativnih primerov, ki jih pravila pokrijejo, mora biti čim manjše.

Za vsako pravilo r izračunamo štiri osnove karakteristike:

- $\overline{P}$: število pozitivnih primerov, ki jih pravilo pokrije,
- $\overline{N}$: število negativnih primerov, ki jih pravilo pokrije,
- P: število vseh pozitivnih primerov,
- N: število vseh negativnih primerov.

Večino mer za vrednotenje pravil izpeljemo iz teh štirih karakteristik. Dve osnovni meri sta specifičnost in občutljivost in ju izračunamo z naslednjimi enačbami:

$$\text{Specifičnost}(r) = \frac{N - \overline{N}}{N}. \tag{2.1}$$

in

$$\text{Občutljivost}(r) = \frac{\overline{P}}{P}. \tag{2.2}$$

Specifičnost meri delež pravilno pokritih negativnih primerov in zato ne maksimizira popolnosti pravila. Občutljivost meri delež pravilno pokritih pozitivnih primerov in zato ne optimizira konsistentnosti. Ti dve meri za ocenjevanje pravil zato uporabljamo skupaj. Občutljivost trivialno maksimiziramo tako, da s pravilom pokrijemo vse primere, a s tem dobimo najslabšo možno konsistentnost.

Uporabimo lahko tudi mero podpora:

$$\text{Podpora}(r) = \frac{\overline{P}}{P + N}, \tag{2.3}$$

ki pa je podobna občutljivosti, saj se razlikuje le za normalizacijski faktor in ima zato enake slabosti kot občutljivost. Mera, ki upošteva število pravilno napovedanih pozitivnih in negativnih primerov, je klasifikacijska točnost, ki izračuna delež pravilno klasificiranih primerov in je definirana kot

$$\text{Točnost}(r) = \frac{\overline{P} + (N - \overline{N})}{P + N}, \tag{2.4}$$

Izračunamo lahko tudi delež vseh primerov, ki jih pravilo pokrije. Mero imenujemo pokritost in je uporabna, če želimo pravilo posplošiti tako, da zajame čim več primerov. Definirana je kot:

$$\text{Pokritost}(r) = \frac{\overline{P} + \overline{N}}{P + N}. \tag{2.5}$$

Uporabna mera za vrednotenje pravil je tudi natančnost:

$$\text{Natančnost}(r) = \frac{\overline{P}}{\overline{P} + \overline{N}}, \tag{2.6}$$

ki jo lahko maksimiziramo tako, da ne pokrijemo nobenega negativnega primera. Z maksimizacijo te mere dobimo pravila, ki pokrivajo le pozitivne primere. Težava te mere je, da pri njeni uporabi dobimo mnogo specifičnih pravil, ki pokrijejo majhno število primerov.

Obstajajo še druge mere za ocenjevanje kakovosti pravil, kot so entropija, gini indeks, mera F, koeficient Q, Laplace, m-Estimate ter druge, ki pa jih v nadaljevanju disertacije ne bomo uporabljali.

2.3 Algoritmi za učenje pravil

Odločitvena pravila so pomembna za področje strojnega učenja, ker so kompaktna in razumljiva. V tem razdelku opišemo pomembne metode za učenje pravil. Nekatere metode so pomembni zgodovinski mejniki, druge pa so v uporabi še danes.

2.3.1 AQ in PRISM

Metoda AQ [12] je bila prva metoda za učenje pravil, ki je uporabljala pristop loči in obvladaj (separate and conquer[1]), da je pokrila množico učnih primerov. Izraz loči in obvladaj sta pri učenju pravil prvič uporabila Bagallo in Haussler [13]. Algoritem za svoje delovanje potrebuje en naključno izbran pozitiven in en negativen učni primer. AQ zgradi pravilo, ki pokrije pozitiven primer, hkrati pa ne pokrije negativnega. Ker sta izbrana dva naključna primera, je delovanje algoritma odvisno od izbranih učnih primerov. PRISM [14] za svoje delovanje ne uporablja naključnih primerov, ampak pregleda celotno učno množico. Za preiskovanje uporablja princip preiskovanja od zgoraj navzdol. Težava obeh algoritmov je, da ne znata delati s šumnimi podatki.

2.3.2 CN2

Algoritem CN2 [10] je združil ideje algoritma AQ [12] in algoritma za učenje odločitvenih dreves ID3 [15]. Cendrowska [14] je opazil povezavo med drevesi in pravili v tem, da vsaka pot od korena do lista v drevesu predstavlja pravilo. Pravila dobljena iz drevesa so neprekrivajoča, kar je sicer omejitev algoritma. Algoritem CN2 je odpravil pomanjkljivost prejšnjih algoritmov, ker se izogiba prevelikemu prileganju učnim podatkom ter zna obravnavati podatke z več razredi in ne le binarne razrede, kot predhodni algoritmi.

2.3.3 FOIL

Algoritem FOIL [16] je prvi, ki je vpeljal možnost podajanja predhodnega znanja v sistem v obliki logike prvega reda.

[1] Podoben znan angleški izraz je 'divide and conquer' (deli in vladaj), ki je tehnika rekurzinega programiranja.

2.3.4 RIPPER

Algoritem RIPPER [5] je eden izmed vodilnih sistemov za rudarjenje pravil. Novost sistema je pri obravnavanju šumnih podatkov, s čimer se izogne prevelikemu prileganju podatkov učni množici. Sistem daje primerljive rezultate s sistemom C4.5 [17] za rudarjenje odločitvenih dreves.

2.3.5 FURIA

Algoritem FURIA [4] je eden izmed vodilnih algoritmov za rudarjenje mehkih pravil. Osnovan je na metodi RIPPER in za razliko od ostalih sistemov rudari neurejene sezname pravil, kar pomeni, da pri klasifikaciji pravil ne izvajamo po vrstnem redu, pač pa upoštevamo vsa pravila, ki pokrijejo dani primer.

2.4 Optimizacija z metodo kolonije mravelj

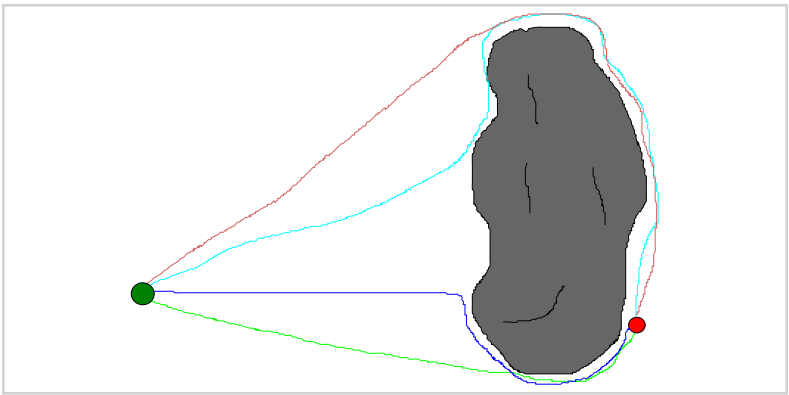
V tem razdelku 2.4.1 najprej opišemo osnovno idejo optimizacije z metodo kolonije mravelj. V razdelku 2.4.2 sledi kratek opis uporabe optimizacije s kolonijo mravelj s poudarkom na uporabi pri strojnem učenju. Razdelka 2.5.1 in 2.5.2 vsebujeta kratek opis metod za učenje pravil, na katerih je osnovano naše delo.

2.4.1 Navdih iz narave

Optimizacija z metodo kolonije mravelj je biološko navdihnjena metoda. Princip simulira proces iskanja hrane pri mravljah. V naravi mravlje med seboj posredno komunicirajo preko feromonov - kemičnih snovi, ki jih izločajo njihove žleze. Feromoni sprožajo naravni odziv drugih osebkov iste vrste. Ko mravlje *iskalci* iščejo hrano, na svoji poti spuščajo feromon, s katerim označijo pot, da se po njej lahko vrnejo v mravljišče. Druge mravlje raje sledijo potem, na katerih je več feromona. Zaradi tega in dejstva, da feromon s časom izhlapeva, se navadno ustvari samo ena močno označena pot, ki ji sledijo vse mravlje.

To biološko zakonitost izkoriščamo pri optimizaciji s simuliranjem umetnih mravelj. Recimo, da več mravelj pri iskanju hrane naleti na kamen in na njegovi drugi strani je hrana, kot je prikazano na sliki 2.3. Na sliki zelen krog predstavlja mravljišče, rdeč krog pa hrano. Če mravlje po narisanih poteh prispejo do hrane, se bo mravlja, ki je šla po zeleni in hkrati najkrajši poti, hitreje vrnila v mravljišče

in bo zato na tej poti v tem trenutku več feromona. Zaradi tega bo naslednja mravlja z večjo verjetnostjo sledila tej najkrajši poti in jo s tem še bolj ojačala. Kmalu bodo zaradi močne sledi feromona (skoraj) vse mravlje izbrale to pot, ki je optimalna z vidika optimizacije.



Slika 2.3
Prikaz različnih poti, ki jih lahko opravijo mravlje.

2.4.2 *Uporaba v računalništvu*

Simulacija mravelj pri iskanju hrane se je v računalništvu uveljavila kot optimizacijska metoda. Uporabljena sta principa feromonov in izhlapevanja feromonov. Dodatno pri simulaciji umetnih mravelj uporabljamo tudi hevristike za lažji hladni zagon.

Princip delovanja kolonije mravelj je najlažje obrazložiti na primeru trgovskega potnika, pri katerem se je metoda izkazala za učinkovito. Pri problemu trgovskega potnika iščemo najkrajšo pot, ki obišče vsa mesta in se vrne v izhodišče. Problem lahko predstavimo z grafom, v katerem vsako vozlišče predstavlja eno mesto. Povezave med mesti označimo z razdaljo med mesti. Tak graf je poln, saj dopuščamo prehode med vsemi mesti. Če prehod med dvemi mesti ni možen, razdaljo med njima označimo z ∞.

Mravlje iščejo najkrajšo pot tako, da vsaka mravlja začne v naključnem vozlišču grafa in izbere eno izmed povezav, ki vodijo iz tega vozlišča. V naslednjem vozlišču mravlja izbere naslednjo povezavo, pri čemer ne more izbrati povezav, ki vodijo v že obiskana vozlišča. To ponavlja, dokler ne obišče vseh vozlišč. Vsak obhod

predstavlja eno od možnih rešitev problema trgovskega potnika. Mravlja izbira povezave na podlagi feromonov in hevristike za zaželjenost povezav. Hevristika se s časom ne spreminja in je v primeru trgovskega potnika določena kot obratna vrednost razdalje med mesti. To pomeni, da mravlje raje izbirajo krajše poti, če so na povezavah podobne vrednosti feromonov, kar večinoma drži na začetku izvajanja algoritma. Feromoni se spreminjajo glede na prejšnje pohode mravelj, saj vsaka mravlja na svoji poti pušča sled feromona glede na kakovost zgrajene rešitve. Zaradi izhlapevanja se vrednosti feromonov na vseh poteh s časom zmanjšujejo. S tem se začetne slabe izbire s časom pozabijo. Izbira naslednje povezave je izračunana verjetnostno na podlagi vrednosti feromona in hevristik zaželjenosti, pri čemer imajo poti z večjimi vrednostmi večjo verjetnost izbire. Mravlje preiskujejo graf, dokler niso izpolnjenji ustavitveni pogoji.

Zgornji postopek opisuje inačico optimizacije s kolonijo mravelj imenovano Ant System(AS). Boljše rezultate pri učenju odločitvenih pravil daje sistem MAX-MIN Ant System [18], ki se od AS razlikuje v treh podrobnostih. V vsaki iteraciji vrednosti feromonov posodablja le mravlja z najboljšo rešitvijo. Vrednosti feromonov so omejene na interval $[\tau_{min}, \tau_{max}]$ in začetne vrednosti feromonov so nastavljene na τ_{max} namesto na 0, kot pri sistemu AS. Te spremembe povzročijo boljše iskanje pravil na začetku preiskovanja.

2.5 Učenje pravil z uporabo metod kolonije mravelj

Pri strojnem učenju pricip kolonije mravelj [19] uporabljamo predvsem za gručenje in učenje pravil. Ant-Miner [20], prva metoda za učenje klasifikacijskih pravil z uporabo metode mravelj, je bila razvita leta 2001 in objavljena leto kasneje. Glavna ideja te metode je, da se problem klasifikacije prevede na iskanje poti v grafu, pridobljene iz učnih primerov. Kasnejše manjše izboljšave te metode sta naredila Wang in Feng [21] ter Liu in sod. [22], večja izboljšava pa se imenuje Ant-Miner+, ki jo je razvili Martens [2]. Metodi Ant-Miner in Ant-Miner+ sta na kratko predstavljeni v nadaljevanju, na koncu pa omenimo še druge manjše izboljšave teh metod.

2.5.1 *Ant-Miner*

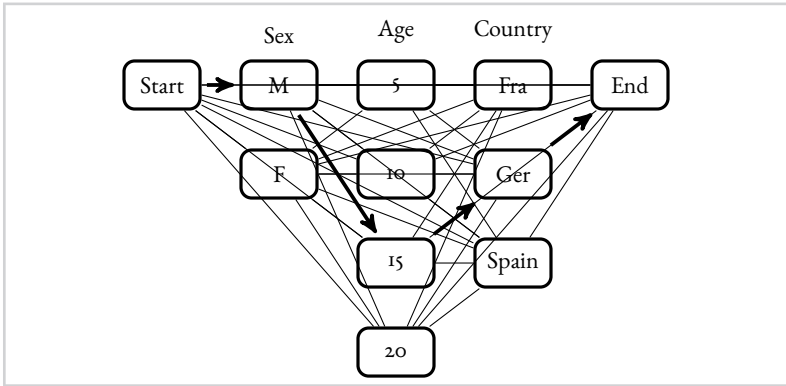
Ant-Miner [20] je prva metoda, ki išče pravila oblike IF-THEN z uporabo kolonije mravelj. Metoda lahko obravnava le nominalne atribute, uporablja prekrivni princip in ustvari urejen seznam pravil. Pravila morajo biti zato pri klasifikaciji uporabljena v istem vrstnem redu, kot so bila ustvarjena. Metoda se nauči enega pravila, nato so s tem pravilom pokriti primeri odstranjeni iz učne množice in učenje se ponovi na novi učni množici. Parpinelli in sod. [20] so metodo podrobno opisal in jo primerjal z algoritmom CN2, s katerim dobiva primerljive rezultate.

Slika 2.4 prikazuje preiskovalni graf te metode. Primer prikazuje tri atribute: Sex, Age in Country. Prvi atribut ima dve nominalni vrednosti, drugi štiri in tretji tri. Vsaka vrednost atributa je predstavljena z vozliščem. V graf sta poleg vozlišč, ki predstavljajo vrednosti atributov, dodani še vozlišči *Start* in *End*. Mravlje potujejo od vozlišča *Start* preko različnih povezav proti vozlišču *End*. Pri tej inačici so možni vsi prehodi, tako da je graf poln. Izbira naslednjega vozlišča je odvisna od hevristike η in feromonov τ, ki se hranijo v vsakem vozlišču posebej. Mravlja začne s praznim pravilom, z vsakim obiskom vozlišča pa doda člen k trenutnemu pravilu. Recimo, če mravlja izbere vozlišče *Fra*, bo dodala člen `Country = Fra` v pravilo. Ko mravlja izbere vozlišče atributa, onemogoči vse ostale vrednosti tega atributa ter tako prepreči njihovo naknadno izbiro. V našem primeru mravlja ne more več izbrati vozlišč *Ger* in *Spain*. To prepreči nesmiselna pravila, kot na primer IF Country = Fra AND Country = Ger THEN True. Mravlja dodaja člene, dokler ne izbere vseh vozlišč oziroma dokler pravilo ne doseže praga minimalnega pokritja (če torej trenutno pravilo pokrije premalo primerov, se algoritem ustavi). Ko mravlja izbere vozlišče *End*, se algoritem ustavi in doda pravilo, ki je sestavljeno iz do sedaj izbranih členov in večinskega razreda primerov, ki ga pravilo pokrije.

Ko mravlja zgradi pravilo, se le-to obreže, oziroma odstrani odvečne člene. Mravlja posodobi vrednosti feromonov glede na kakovost zgrajenega pravila. Kakovost pravila Q je definirana kot:

$$Q = \frac{\overline{P}}{P} \cdot \frac{N - \overline{N}}{N}, \tag{2.7}$$

kjer so definicije spremenljivk P, $\overline{P}$, N in $\overline{N}$ podane v razdelku 2.2. Preprosteje lahko Q predstavimo kot Občutljivost(r) $\cdot$ Specifičnost(r) pravila r, kar pomeni, da velja $0 < Q < 1$. Vrednosti feromonov obiskanih vozlišč se posodobijo po



Slika 2.4
Primer grafa za metodo Ant-Miner.

enačbi (2.8). Vrednosti feromonov normaliziramo z vsoto vseh τ_{ij}, kar simulira izhlapevanje feromonov, saj normalizacija zmanjša vrednost feromonov na vozliščih, ki niso bila posodobljena v prejšnjem koraku. Vrednost $\tau_{ij}(t)$ predstavlja vrednost feromona j-te vrednosti i-tega atributa v času t. Začetna vrednost feromona $\tau_{ij}(0)$ je določena z enačbo (2.9), kjer a predstavlja število atributov in b_i predstavlja število različnih vrednosti atributa i.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \mathrm{Q} \cdot \tau_{ij}(t) \tag{2.8}$$

$$\tau_{ij}(0) = \frac{1}{\sum_{i}^{a} b_i} \tag{2.9}$$

Hevristika η se uporablja za usmerjanje iskanja na začetku, ko so vrednosti feromonov nizke. Pri metodi Ant-Miner je hevristika izračunana na podlagi entropije.

P_{ij} je verjetnost izbire vozlišča, ki predstavlja j-to vrednost i-tega atributa. Izračunamo jo z enačbo (2.10), kjer i teče le čez atribute, ki jih mravlja še ni izbrala.

$$\mathrm{P}_{ij} = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{i}^{a} \sum_{j}^{b_i} \tau_{ij}(t) \cdot \eta_{ij}}. \tag{2.10}$$

Ta postopek se ponavlja, dokler ne dosežemo konvergence. Konvergenca pomeni, da se isto pravilo večkrat ponovi. Iskanje ustavimo tudi, če presežemo vnaprej določeno število mravelj, ki jih pošljemo skozi graf. Psevdo koda pristopa je vidna v algoritmu 1. Algoritem najprej za učenje v prvi vrstici izbere vse učne primere (TS) in ustvari prazen seznam pravil (RL). Zanka *while* med vrsticami 3 in 13 se ponavlja, dokler algoritem ne pokrije dovolj učnih primerov, kar je parameter metode. V vsakem obhodu te zanke se najprej v vrstici 4 iz učnih primerov TS zgradi graf in posodobi začetne vrednosti hevristik, feromonov in verjetnosti za izbiro vozlišč. Nato notranja zanka *while* med vrsticami 5 in 10 dobi najboljše pravilo, ki se v vrstici 11 doda v seznam dobljenih pravil RL. V vrstici 12 pa se primeri, ki jih dobljeno pravilo dobi, odstranijo iz učne množice. Notranja zanka *while* predstavlja en obhod mravlje čez graf. V vrstici 6 mravlja naredi sprehod čez graf, kar predstavlja pravilo R_i. Pravilo se nato v vrstici 7 obreže in z njim se posodobijo feromoni v grafu. Če je trenutno pravilo, glede na kakovost pravila, boljše od prejšnjih, se le-ta posodobi v vrstici 9 in shrani v R_{best}.

Algorithm 1: Ant-Miner.
Input: dataset, number of ants, convergence
Output: set of rules

```
1:  TS ← {All training examples}
2:  RL ← {}
3:  while not enough instances covered do
4:      Initialize heuristic(η), pheromones(τ) and probabilities(P)
5:      while max number of ants not reached or convergence not reached do
6:          Let an ant run from Start to End, defining rule R_i
7:          Prune Rule R_i
8:          Update pheromone values
9:          Update R_best if better rule R_i is found
10:     end while
11:     Add R_best to RL
12:     TS = TS \ {examples covered by R_best}
13: end while
```

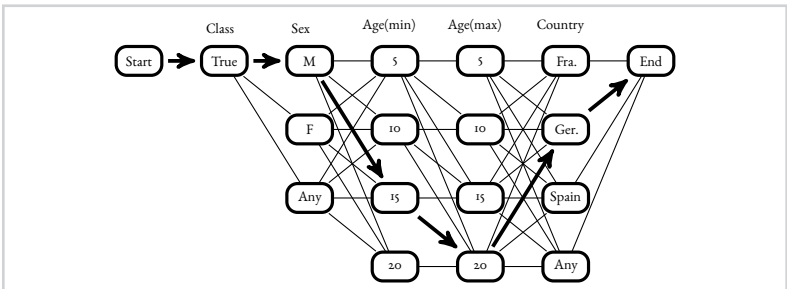
2.5.2 *Ant-Miner+*

Ant-Miner+ [2] je izboljšava algoritma Ant-Miner, ki uporablja MAX-MIN Ant System. Dve poglavitni izboljšavi sta nova predstavitev grafa preiskovanja, s čimer se zmanjša prostor preiskovanja in posledično dobi pravila z boljšo klasifikacijsko točnostjo, in razlikovanje med nominalnimi in ordinalnimi atributi, kar doprinese k razumljivosti dobljenih pravil. Metoda se ustavi, ko so vrednosti feromonov na eni poti enake τ_{max}, na vseh ostalih pa τ_{min}. Vrednosti feromonov niso shranjene v vozliščih, ampak na povezavah, kar je smiselno glede na predstavitev grafa, ki je prikazana na sliki 2.5.

Ant-Miner+ razlikuje med nominalnimi in ordinalnimi atributi in jih različno obravnava. Ordinalni atributi so predstavljeni z dvema skupinama atributov, kjer vsaka skupina vsebuje vse vrednosti atributa. Prva skupina predstavlja spodnjo mejo, druga pa zgornjo mejo vrednosti člena v pravilu. S to predstavitvijo imamo za ordinalne atribute dve meji in ne le ene kot pri metodi Ant-Miner. Slika (2.5) prikazuje obhod mravlje s pravilom :

```
IF Sex = male AND Age > 15 AND Age <= 20 AND
   Country = Germany > THEN True.
```

Na sliki je z odebeljeno črto označena pot mravlje. V danem primeru je možna izbira vrednosti razreda le *True*, ker Ant-Miner+ v graf doda vse vrednosti razreda, razen večinskega razreda. S tem skuša dobiti bolj specifična pravila in prepustiti privzetemu pravilu, da pokrije večinski razred.



Slika 2.5
Prikaz grafa metode Ant-Miner+.

Iz slike 2.5 je razvidno, da graf ni poln. To ni potrebno, ker je pri metodi Ant-Miner+ vrstni red atributov določen vnaprej, pri metodi Ant-Miner pa vrstni

red atributov določajo mravlje sproti. Zaradi vsiljenega vrstnega reda atributov se nominalnim atributom doda vozlišče *Any*, da lahko mravlja preskoči določen nominalni atribut, če je neuporaben. Metoda Ant-Miner tega vozlišča ne potrebuje, ker konča gradnjo pravila, ko obišče vozlišče *End*. Ordinalni atributi so predstavljeni z dvema skupinama atributov, ena predstavlja spodnjo mejo in druga zgornjo. Ordinalni atribut preskočimo z izbiro minimalne vrednosti v prvi skupini in maksimalne vrednosti v drugi skupini. Povezave obstajajo iz skupine g_i do skupine g_{i+1}, kjer je vsako vozliče iz skupine g_i povezano z vsakim vozliščem skupine g_{i+1}. Izjema so skupine, ki predstavljajo isti ordinalni atribut. V tem primeru je vozliče $v_{i,j}$ skupine g_i povezano z vozliščem $v_{i+1,k}$ skupine g_{i+1} le v primeru, ko velja $v_{i+1,k} > v_{i,j}$. V najslabšem primeru imamo $O(u^2)$ povezav med skupinama vozlišč, kjer u predstavlja število unikatnih vrednosti atributa.

V grafu so tudi skrita vozlišča za izbiro vrednosti parametrov α in β z vrednostmi 1, 2 ali 3. Ta vozlišča na sliki 2.5 niso vidna, so pa postavljena med vozličem *Start* in skupino za izbiro razreda. Ta dva parametra sta uporabljena pri izračunu verjetnosti za izbiro naslednjega vozlišča v grafu, tako da vrednost feromonov potenciramo z α in vrednost hevristike z β, kar pomeni, da visoke vrednosti α dajo večjo težo feromonom, visoke vrednosti β pa večjo težo začetni hevristiki.

Empirični testi so pokazali boljše delovanje metode Ant-Miner+ v primerjavi z metodo Ant-Miner. Metoda Ant-Miner+ uporablja validacijsko množico kot ustavitveni pogoj pri rudarjenju velikih učnih množic. Psevdo koda je vidna pri algoritmu 2. Algoritem najprej za učenje v prvi vrstici izbere vse učne primere (TS) in ustvari prazen seznam pravil (RL), enako kot metoda Ant-Miner. Zanka *while* med vrsticami 3 in 15 se ponavlja dokler ni dosežen ustavitveni pogoj, ki na posebni validacijski množici spremlja napako dobljenih pravil. Če se ta napaka začne večati se algoritem ustavi. V vsakem obhodu te zanke se najprej v vrstici 4 iz učnih primerov TS zgradi graf in posodobi začetne vrednosti hevristik, feromonov in verjetnosti prehodov. Nato notranja zanka *while* med vrsticami 5 in 12 dobi najboljše pravilo, ki se v vrstici 13 doda v seznam dobljenih pravil RL. V vrstici 14 pa se primeri, ki jih dobljeno pravilo dobi, odstranijo iz učne množice. Notranja zanka *while* predstavlja iskanje enega pravila, ki je najboljše pravilo pri prehodu večjega števila mravelj v vrstici 6. Vrstica 7 simulira izhlapevanje feromonov. Najboljše pravilo, dobljeno v vrstici 6, se v vrstici 8 obreže in v vrstici 9 to pravilo doda feromone na svojo pot. Ker morajo zaradi uporabe MAX-MIN

Ant-Sistema biti vrednosti feromonov omejene, se v vrstici 10 le-te po potrebi popravijo. Nazadnje se v vrstici 11 posodobijo verjetnosti prehodov.

Algorithm 2 : Ant-Miner+.
Input: dataset, number of ants, convergance, ρ
Output: set of rules

```
1:  TS ← {All training examples}
2:  RL ← {}
3:  while not early stopping do
4:      Initialize probabilities, pheromones and heuristics
5:      while not converged do
6:          Let ants run from Start node to End node
7:          Evaporate pheromone
8:          Prune rule of the best ant R_best
9:          Add pheromone to the path of R_best
10:         Adjust pheromones if needed
11:         Update probabilities of edges
12:     end while
13:     Add R_best to RL
14:     TS = TS \ {examples covered by R_best}
15: end while
```

Slaba lastnost metode Ant-Miner+ je v predhodno določenem vrstnem redu atributov. Algoritem daje pravila glede na začetno izbiro vrstnega reda atributov. Pri velikih grafih metoda včasih ne konvergira, zato je določeno maksimalno število mravelj, ki lahko preiskujejo graf. Če je to število doseženo, algoritem vrne do takrat najboljše pravilo.

2.5.3 Ostale izboljšave in sorodne metode

Ker učenje pravil z mravljami daje dobre rezultate, je mnogo raziskovalcev poskušalo obstoječe metode izboljšati. Raziskave so potekale predvsem glede oblike grafa, posodabljanja feromonov, obravnave zveznih atributov, uporabe različnih vrednotnih funkcij in oblike dobljenih pravil. V nadaljevanju na kratko predstavimo te pristope.

Večina metod na osnovi kolonije mravelj uprablja pristop loči in obvladaj [23] za gradnjo seznama pravil, kar pomeni, da najprej zgradi eno pravilo, nato pa pokrite podatke odstrani iz učne množice.

Posodabljanje feromonov v grafu je lahko globalno ali lokalno. Pri lokalnem posodabljanju nimamo informacije o feromonih na celotnem grafu, ampak le o delu tega. Globalno posodabljanje se je izkazalo za boljše, kot so opisali Dorigo in sod. [24], takšno posodabljanje uporabljamo v obeh razvitih metodah. Podobno sta avtorja Wang in Feng [21] za računanje feromonov uporabila hevristiko in s tem pohitrila algoritem na račun nekoliko slabše napovedi, hkrati pa sta tudi uporabila prilagodljiv faktor izhlapevanja, ki se spreminja skozi čas. Obstajajo tudi omejitve na vrednosti feromonov, tako so na primer pri sistemu MAX-MIN Ant System [18] le-te omejene med τ_{min} in τ_{max}, kar pa se s časom spreminja. V tem sistemu so začetni feromoni nastavljeni na maksimalno vrednost namesto na minimalno ter le izbrane najboljše mravlje posodabljajo feromone. Naša razvita metoda nAnt-Miner uporablja MAX-MIN Ant System.

Liu in sodelavci [22] so pri določanju izbire naslednjega prehoda v grafu poleg feromonov in hevristik uporabili tudi naključna števila, da so s tem razširil raziskovanje algoritma, saj je algoritem tako preveril večje število poti v grafu. Avtorja Yildirim in Çatay [25] sta vrednost feromonov spreminjala glede na čas. To je na primer uporabno pri optimizaciji problema usmerjanja vozil, ki so omejena s časovnimi okni. Te razširitve v našem delu niso uporabljene.

Nekateri avtorji so poskušali obogatiti metodo mravelj s principi drugih metod, na primer Cordón in sodelavci [26] so uvedli mutacijo feromona, osnovano na genetskih algoritmih, ter ponovne zagone določanja feromonov, da se metoda izogne lokalnim optimumom. Ti pristopi so bili testirani na problemu kvadratnega prirejanja (Quadratic Assignment Problem, QAP) in niso uporabljeni v našem delu, vendar bi bila možna uporaba ponovnega zagona določanja feromonov za razširitev metode ProAnt-Miner.

Avtorja Salama in Abdelbar [27] sta učenje pravil pri rudarjenju pravil s kolonijo mravelj razširila z uporabo logične NOT operacije. Dodala sta tudi muhaste mravlje, ki raje sledijo svoji poti, kot drugim, in mravlje z osebnostjo, kjer se nekateri parametri prilagajajo vsaki mravlji posebej. Del, ki je uporabljen tudi v našem delu, je, da so uporabljeni različni feromoni za vsak napovedan razred, ostalih lastnosti mravelj pa nismo vključili. Te razširitve so bile podrobno raziskane

in testirane v delu avtorja Liu in sod. [22]. Avtorja Chan in Freitas [28] sta optimizirala časovno zahtevnost rezanja seznama pravil, ki upočasnjuje delovanje algoritmov za rudarjenje pravil, osnovanih na metodi kolonije mravelj. V našem delu v obeh metodah na koncu seznama pravil ne režemo, vendar bi bilo to koristno pri metodi ProAnt-Miner, kot je razvidno iz rezultatov v razdelku 4.5.4.

Avtorja Medland in Otero [29] sta metode za učenje pravil s kolonijo mravelj primerjala z različnimi funkcijami za ovrednotenje kakovosti pravil, kot so občutljivost · specifičnost, Klogsenova mera, m-ocena, Jaccardova mera ter zaupanje + pokritost. V obeh naših metodah uporabljamo občutljivost · specifičnost; to oceno kakovosti pravil uporablja tudi Ant-Miner+. Primerjata tudi različne mere za kakovost celotnega seznama, kot so točnost, mikro-povprečna F-mera (micro-average f-measure), makro-povprečna F-mera (macro-average f-measure), utežena makro-povprečna F-mera (weighted macro-average f-measure) in obrnjena utežena makro-povprečna F-mera (inverse weighted macro-average f-measure). Avtor Liu in sodelavci [30] so poskušali pohitriti prvotni Ant-Miner z uporabo preprostejše hevristike. Članek avtorjev Baig in Shahzad [31] raziskuje vpliv nove hevristične funkcije, ki temelji na korelaciji med členi dobljenega pravila in rezanega seznama pravil.

Otero in sod. [1, 32] opisujejo metodo cAnt-Miner, ki je obravnavala številske atribute pri metodah s kolonijo mravelj. Atribute avtomatsko diskretizirajo glede na entropijo podatkov v trenutku, ko mravlja izbere številski atribut. V našem algoritmu nAnt-Miner to težavo odpravimo s tem, da uporabimo večje število vozlišč za predstavitev vsakega številskega atributa, pri metodi ProAnt-Miner pa so številski atributi predstavljeni z distribucijo feromonov. Zanimiva razširitev optimizacije tega algoritma je v delu avtorja Otero in sod. [33], kjer za optimizacijo uporabljajo pristop Pittsburg, ki optimizira celoten seznam pravil namesto vsakega pravila posebej. Za uporabo tega pristopa bi bilo v našem delu treba povečati graf preiskovanja.

Metode optimizacije s kolonijo mravelj so tudi širše uporabne v strojnem učenju. V nadaljevanju navajamo nekaj primerov. Iskanje Pareto optimalnih rešitev s kolonijo mravelj so opisali avtorji López-Ibáñez in Stützle [34] ter Said in sodelavci [35]. Za rudarjenje hierarhij so Otero in sodelavci [36] razvili metodo za potrebe rudarjenja ontologij genov. Ti pristopi kažejo na veliko število možnih uporab kolonije mravelj.

Kolonijo mravelj so uporabili za rudarjenje mehkih pravil s predhodno podanimi mehkimi intervali, kot so pokazali Aribarg in sodelavci [37]. Pri tem pristopu je metoda združena z idejami simuliranega ohlajanja. Pravila naše metode ProAnt-Miner, ki so sicer verjetnostna, lahko spremenimo v podobno obliko teh mehkih pravil. Prednost metode ProAnt-Miner je v tem, da sama gradi mehke intervale in jih ni treba podati vnaprej. Mehka pravila mnogokrat dosegajo boljšo klasifikacijsko točnost kot trda pravila, zato se razvijajo tudi algoritmi, ki pretvarjajo trda pravila v mehka. Eden izmed takšnih algoritmov je opisan v članku avtorjev Bounhas in sodelavcev [38]. Zaradi razumljivosti odločitvenih pravil pa velja tudi obratno, kot je pretvarjanje mehkih pravil v trda ali pretvorba rezultatov drugih metod v odločitvena pravila. Tako so na primer McGarry in sodelavci [39] opisal metodo za pretvorbo nevronskih mrež v odločitvena pravila. Verjetnostna pravila pa so razvijali tudi Tresp in sodelavci [40].

Pellegrini in sod. [41] so pri optimizaciji s kolonijo mravelj uporabili dinamično prilagajanje parametrov, kar pomeni, da so se parametri metode avtomatsko prilagajali med potekom iskanja pravil. Ta pristop bi bilo smiselno dodati metodi ProAnt-Miner in parametre prilagajati vsaki učni množici posebej.

3

Metoda nAnt-Miner

V tem poglavju predstavimo našo prvo razvito metodo imenovano nAnt-Miner (numeric Ant-Miner), ki je zasnovana na metodi Ant-Miner+ [2], vendar odpravlja pomanjkljivost te metode tako, da zna poleg nominalnih in ordinalnih atributov obravnavati tudi zvezne atribute. V razdelku 3.1 opišemo implementacijo algoritma, v razdelku 3.2 pa predstavimo empirično primerjavo z drugimi sorodnimi metodami.

3.1 nAnt-Miner

Metodo nAnt-Miner [7] smo razvili z namenom odprave slabosti algoritmov, osnovanih na koloniji mravelj, pri obravnavi številskih atributov. Obstoječe metode, kot sta Ant-Miner in Ant-Miner+, teh atributov ne obravnavajo. Za uporabo teh metod na številskih podatkih je treba zato podatke najprej diskretizirati. Metodo nAnt-Miner natančno opišemo ter razložimo, kako obravnava zvezne atribute.

Metoda nAnt-Miner je zasnovana na metodi Ant-Miner+. Glavna ideja metode je implicitno upoštevanje številskih atributov, ki jih metoda Ant-Miner+ ne more obravnavati brez predhodne diskretizacije. To lahko vodi v slabše delovanje klasifikacijskega modela, saj se pri postopku diskretizacije izgubi del informacije v podatkih. Z našo metodo to informacijo obdržimo in jo lahko učenje izkoristi. Metoda nAnt-Miner obravnava nominalne in ordinalne atribute enako kot Ant-Miner+. Za številske atribute uporablja predstavitev grafa, kar je opisano v nadaljevanju.

Algoritem 3 prikazuje psevdo kodo algoritma nAnt-Miner. V prvi vrstici naloži vse učne podatke (TS), druga pa ustvari prazen seznam pravil (RL). Vsak obhod zanke *while* v vrsticah od 3 do 15 zgradi eno pravilo in ga doda v seznam pravil. Proces dodajanja novih pravil ustavimo, ko so pokriti vsi primeri, oziroma ko na novo dodano pravilo zmanjša klasifikacijsko točnost na validacijski množici, kar preveri zanka *while* v tretji vrstici. Vrstici 4 in 5 zgradita graf preiskovanja in nastavita začetne vrednosti hevristik, feromonov in verjetnosti prehodov. Vrstici 13 in 14 dodata dobljeno najboljše pravilo R_{best}, dobljeno v zanki *while* med vrsticami 6 in 12, v seznam pravil in odstranita pokrite primere iz učne množice. Konvergenca notranje zanke *while* se preveri na dva načina, in sicer, če se pravilo dovoljkrat ponovi in s preverjanjem vrednosti feromonov številskega atributa (opisano v razdelku 3.1.6). V sedmi vrstici se zgradi pravilo s simulacijo obhoda mravlje skozi

graf. Vsaka mravlja gre skozi več skupin vozlišč in s tem dodaja člene k svojemu pravilu. Vsaka mravlja naredi pot od vozlišča *Start* do vozlišča *End* in s tem zgradi natanko eno pravilo. Osma vrstica shrani omejeno število najboljših pravil, ki jih imenujemo elitna pravila. Deveta vrstica poreže dobljena pravila in s tem odstrani nepotrebne člene iz pravila ter poveča pokritost pravil, kot je opisano v razdelku 3.1.5. Deseta vrstica shrani do sedaj najboljše dobljeno pravilo R_{best} in vrstica 11 posodobi vrednosti feromonov v grafu glede na najboljše pravilo in elitna pravila, kar je podrobneje opisano v razdelku 3.1.3.

Algorithm 3 : nAnt-Miner.
Input: dataset, number of ants, elite, w, ω, convergence, ρ
Output: set of rules

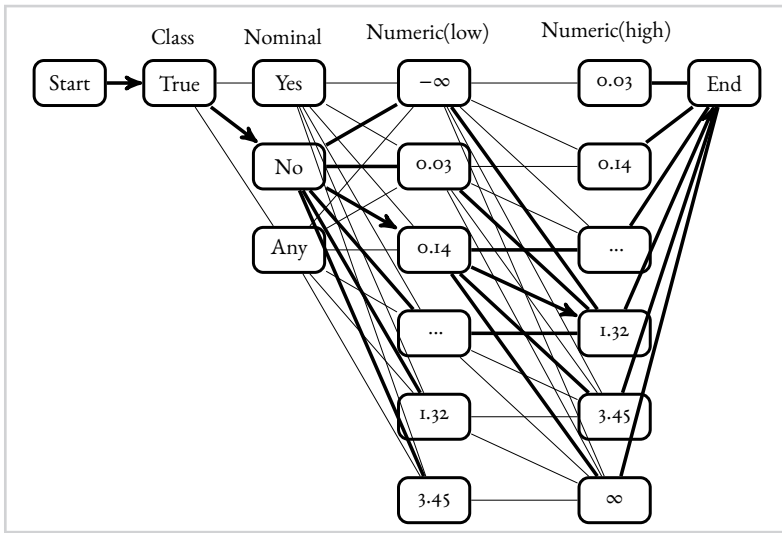
```
1:  TS ← {All training instances}
2:  RL ← {}
3:  while not stopping criterion do
4:     construct graph
5:     Initialize heuristic(η), pheromones(τ), probabilities(P)
6:     while not converged do
7:        Let ants run from Start to End nodes
8:        Keep 'elite' number of rules
9:        Prune rules
10:       Update global best rule R_best
11:       Update pheromone values on paths defined by elite rules and global best
          rule
12:    end while
13:    Add R_best to RL
14:    TS = TS \ {instances covered by R_best}
15: end while
16: return RL
```

3.1.1 Predstavitev grafa

Preiskovalni graf je pri metodi nAnt-Miner usmerjen. Vsako vozlišče grafa predstavlja eno vrednost atributa, enako kot pri metodi Ant-Miner+ [2]. Povezave v

grafu predstavljajo možne prehode med vozlišči, ki vedno vodijo iz ene skupine atributov do druge skupine in so zasnovane podobno kot pri metodi Ant-Miner+.

Graf je sestavljen iz več skupin, vsaka skupina predstavlja en atribut, razen skupin *Start*, *End* in *Class*. Skupina *Start* ima samo eno vozlišče, v katerem začnejo mravlje preiskovati graf. Skupina *End* ima ravno tako samo eno vozlišče, kjer mravlje zaključijo svoje preiskovanje. Skupina *Class* opisuje vrednosti razredov. Ta skupina vsebuje $r - 1$ vozlišč, kjer je r število možnih razredov. Dodani so vsi razredi razen večinskega. Večinski razred pokrije privzeto pravilo.



Slika 3.1
Predstavitev grafa metode nAnt-Miner.

Preostale skupine predstavljajo atribute učne množice. Vsak nominalni atribut je predstavljen z eno skupino, ordinalni in številski pa s po dvema skupinama. Nominalni atributi vsebujejo vozlišča, ki predstavljajo vrednosti nominalnega atributa ter vozlišče *Any*. Ko mravlja izbere eno od teh vozlišč, ustvari člen pravila $A = v_i$, kjer je A ime atributa in v_i i-ta vrednost tega atributa. V primeru, da mravlja izbere vozlišče *Any*, se atribut preskoči brez gradnje člena. To vozlišče mora biti na voljo, ker algoritem obišče vse atribute in bi brez vozlišča *Any* vsiljeval prisotnost vseh nominalnih atributov v vseh pravilih.

Vsak ordinalni atribut je predstavljen z dvema skupinama. Prva skupina izbere

spodnjo mejo v_i, druga skupina pa zgornjo mejo v_j člena $v_i \leq O < v_j$, kjer O predstavlja ime ordinalnega atributa, v_i in v_j pa sta vrednosti tega atributa. Povezave v grafu so ustvarjene tako, da dovoljujejo le povezave, ki zagotavljajo $v_i < v_j$.

Tudi vsak številski atribut je predstavljen z dvema skupinama. Vozlišča teh dveh skupin so sestavljena iz vseh možnih unikatnih vrednosti atributa iz učne množice. Vozlišče z vrednostjo -∞ je dodano k skupini za izbiro spodnje meje, vozlišče +∞ pa k skupini za izbiro zgornje meje. Številske atribute je možno preskočiti z izbiro vrednosti -∞ v prvi skupini in +∞ v drugi skupni. Zgrajeni člen je oblike $v_i \leq A < v_j$, kjer je A ime številskega atributa, v_i in v_j pa vrednosti atributa A. Tudi v tem primeru povezave v grafu dopuščajo le izbiro vrednosti tako, da velja $v_i < v_j$. Število vozlišč v vsaki skupni je navzgor omejeno s številom primerov v učni množici.

Povezave potekajo iz skupine g_i proti skupini g_{i+1}. Vsako vozlišče skupine g_i je povezano z vsakim vozliščem skupine g_{i+1}, razen v primeru, ko g_i in g_{i+1} pripadata istemu atributu. Kot rečeno, v tem primeru obstajajo le povezave, pri katerih je vrednost vozlišča iz skupine g_i manjša ali enaka vrednosti vozlišča v skupini g_{i+1}. Število povezav med dvema skupinama, ki sestavljata številski atribut, kvadratično narašča s številom različnih učnih primerov, kar lahko privede do velike prostorske zahtevnosti algoritma pri večjih učnih množicah. Rešitev te težave je opisana v razdelku 3.1.3.

3.1.2 *Verjetnosti na povezavah*

Verjetnost izbire povezave iz vozlišča $v_{i-1,k}$ do vozlišča $v_{i,j}$ se izračuna z enačbo (3.1)

$$p_{ikj} = \frac{[\tau_{v_{i-1,k},v_{i,j}}]^{\alpha} \cdot [\eta_{v_{i,j}}]^{\beta}}{\sum_{l=1}^{|V_i|} [\tau_{v_{i-1,k},v_{i,l}}]^{\alpha} \cdot [\eta_{v_{i,l}}]^{\beta}}, \tag{3.1}$$

kjer $v_{i,j}$ predstavlja j-to vrednost v skupini i. Vrednost $\tau_{v_{i-1,k},v_{i,j}}$ predstavlja vrednost feromona na povezavi med vozliščem $v_{i-1,k}$ ter $v_{i,j}$, vrednost $\eta_{v_{i,j}}$ pa predstavlja hevristično vrednost vozlišča $v_{i,j}$. $|V_i|$ pa je število vseh vrednosti v skupini i. Vrednost feromona je v enačbi kvadrirana, s čimer daje feromonom večji vpliv pri usmerjanju iskanja, ker s tem okrepi razlike. V algoritmu Ant-Miner+

sta za izbiro razmerja moči med feromoni in hevristiko potrebna še parametra α in β. V našem primeru sta ti dve vrednosti vedno konstantni, in sicer $\alpha = 2$ in $\beta = 1$, kar sta tudi privzeti vrednosti v algoritmu Ant-Miner+.

Po enačbi (3.1) lahko izberemo tudi pot, ki ne pokrije nobenega učnega primera. To se lahko zgodi zaradi eksponentnega števila možnih poti v grafu pri velikem številu atributov. V takem primeru lahko mravlja naredi korak nazaj in izbere vrednost, ki pokrije več primerov.

3.1.3 Posodabljanje vrednosti feromona

Metoda nAnt-Miner uporablja princip MAX-MIN Ant System [18]. Vrednosti feromonov so shranjene na povezavah in so vedno med $\tau_{min} = 0$ in $\tau_{max} = 1$. Na začetku izvajanja algoritma so nastavljene na τ_{max}. Algoritem ustavimo, ko so vrednosti feromona na eni poti skozi graf τ_{max}, na ostalih poteh pa je vrednost feromona pod $\tau_{min} + \varepsilon$, kjer je epsilon nastavljen na majhno vrednost 0,05.

Vrednosti feromonov posodabljamo v dveh korakih, in sicer z izhlapevanjem in ojačanjem. Izhlapevanje v vsaki iteraciji algoritma zmanjša vrednost feromona s faktorjem ρ (tipične vrednosti so med 0,8 in 0,99). Ojačanje poveča vrednost feromonov na poti, ki jo obišče mravlja z najboljšim pravilom ali elitna mravlja, z vrednostjo $\frac{Q_i}{t}$, kjer je t tipično 10 in Q_i kakovost pravila trenutne mravlje. Delitelj t preprečuje premočno ojačanje, sicer bi algoritem prehitro konvergiral in se verjetneje ustavil v lokalnem optimumu. Po korakih izhlapevanja in ojačanja se vrednosti popravijo tako, da prevelike vrednosti oziroma vrednosti blizu τ_{min} nastavimo na τ_{max} oziroma τ_{min}.

Kakovost pravil izračunamo z enačbo (3.2), pri kateri lahko z utežjo w nastavljamo, ali naj pravila preferirajo pokritost ali natančnost. Privzeta vrednost uteži je 0,5.

$$Q = \text{Pokritost}(r) \cdot w + \text{Natančnost}(r) \cdot (1 - w) \tag{3.2}$$

Do sedaj opisani pristop deluje, če imamo atribute s približno 15 ali manj vrednostmi in 1000 ali manj učnih primerov, kot je pokazal Martens [2]. Številski atributi imajo večinoma število različnih vrednosti enako številu primerov, pri čemer ta naivni pristop odpove. Pri taki učni množici in iz nje sestavljenem grafu algoritem ne konvergira dovolj hitro. Da se temu izognemo, predpostavimo,

da je količina feromonov na podobnih poteh podobna. Uporabimo normalno distribucijo $N(\mu, \sigma)$, s katero ocenimo kakovost poti in jih nato tudi posodobimo. Slika (3.1) prikazuje poti, ki jih posodobi ena mravlja. Izbrana pot je prikazana s puščicami, vse posodobljene poti pa so ojačane.

Vsaki skupini, ki predstavlja številski atribut A_i, dodelimo σ_i, ki predstavlja začetni razpon posodabljanja. Vrednost σ_i je izračunana kot standardna deviacija vrednosti i-tega atributa. Ko mravlja izbere vozlišče $v_{i,j}$, uporabimo to vrednost kot središče μ posodobitvene distribucije feromonov. Vrednost σ_i skupine uporabimo kot standardno deviacijo, pomnoženo s faktorjem λ, s čimer nadzorujemo širino posodobitvene distribucije feromonov. Faktor λ se v vsaki iteraciji zmanjša za faktor ω (tipične vrednosti so med 0,8 in 0,99). S tem je širina normalne porazdelitve na začetku preiskovanja široka, s čimer preprečimo prehitro konvergenco, kasneje pa se zoža, da pohitri konvergenco.

Na poti skozi vozlišča $v_{i-1,k}$, $v_{i,j}$, $v_{i+1,l}$, kjer $v_{i,j}$ predstavlja j-to vrednost številskega atributa i-te skupine, posodobimo do $2n$ povezav, kjer je n število vozlišč v skupini i. Posodobljene povezave povezujejo vozlišče z vrednostjo $v_{i-1,k}$ z vozlišči skupine i in vozlišča skupine i z vozliščem z vrednostjo $v_{i+1,l}$. Vse vmesne povezave so posodobljene z enačbo (3.3).

$$\Delta\tau_{v_{i-1,k},v_{i,l}} = \frac{0,1 \cdot Q_{best}}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{v_{i,l}-v_{i,j}}{\sigma_i}\right)^2}. \tag{3.3}$$

Kljub takšnemu posodabljanju feromonov ima v grafu večina povezav enake vrednosti. Vrednost $\Delta\tau_{v_{i-1,k},v_{i,l}}$ v zgornji enačbi se prišteje obstoječi vrednosti feromona $\tau_{v_{i-1,k},v_{i,l}}$. Vrednost Q_{best} predstavlja kakovost najboljšega pravila, saj se feromoni posodabljajo glede na njegovo kakovost. Enačba izhaja iz Gaussove krivulje, tako da bolj oddaljene vrednosti od izbrane posodobi šibkeje. Izračunajmo število povezav, ki se razlikujejo od privzete vrednosti. V vsaki iteraciji mravlja posodobi do $2n$ povezav skupine, ki predstavlja številski atribut. Te povezave označimo in jih hranimo. V najslabšem primeru bomo v n iteracijah brez izhlapevanja v grafu imeli $2n^2$ povezav različnih od privzete vrednosti. Če privzamemo faktor izhlapevanja $\rho \in (0, 1]$ ter če vrednosti feromona, ki padejo pod neko majhno vrednost ε (0,05) nastavimo na 0, zadostuje, da hranimo dosti manj povezav. Število potrebnih iteracij, da vrednost feromona pade iz 1 na ε, je $i = \log_\rho(\varepsilon)$. To pomeni, da ima vsak številski atribut največ $2in$ povezav različnih od privzete vre-

dnosti. Na primer, vrednost feromona se v i iteracijah zmanjša iz 1 na ε. Privzeta vrednost feromonov je v tej iteraciji tudi ε, ker tudi privzeta vrednost izhlapeva. Za smiselne vrednosti $\varepsilon \in [0,01,\ 0,05]$ in $\rho \in [0,85,\ 0,99]$ dobimo $i \in (30,\ 500)$, kar omogoča velik prihranek prostora pri velikih učnih množicah.

3.1.4 Hevristika kakovosti poti

Začetna vrednost hevristike v vozliščih je določena z enačbo (3.4).

$$\eta_{ij} = \frac{|\mathrm{T}_{ij} \wedge \mathrm{Class} = l_{\mathrm{ant}}|}{|\mathrm{T}_{ij}|}, \tag{3.4}$$

kjer je $|\mathrm{T}_{ij}|$ število primerov pokritih s pravilom $\mathrm{V}_i = v_{ij}$ in l_{ant} je vrednost izbranega razreda mravlje, vrednost $|\mathrm{T}_{ij} \wedge \mathrm{Class} = l_{\mathrm{ant}}|$ pa predstavlja število pokritih primerov razreda l_{ant}. Hevristika je odvisna od izbranega razreda, kar pomeni, da ima vsako vozlišče $r - 1$ hevristik, kjer je r število razredov. Takšna hevristika je uporabljena za nominalne in ordinalne atribute.

Za številske atribute je večinoma $|\mathrm{T}_{ij}| = 1$, zato je naša hevristika za številske atribute $\frac{1}{n}$, kjer je n število učnih primerov. Zaradi tega je iskanje pri številskih atributih na začetku naključno in popolnoma prepuščeno feromonom.

3.1.5 Rezanje pravil

Pravila na požrešen način porežemo, saj s tem povečamo zaupanje vanje. Pravilo "IF T_1 AND T_2 AND ... AND T_t THEN l_{ant}" z t členi iterativno režemo. Vsak člen T_i odstranimo in izračunamo zaupanje pravila brez tega člena. Če je zaupanje večje ali enako pravilu z vsemi členi, potem člen odstranimo in postopek ponavljamo. S tem postopkom se zaupanje pravila kvečjemu povečuje, hkrati pa se tudi pokrivnost pravil kvečjemu veča.

3.1.6 Ustavitveni pogoj

S preverjanjem vrednosti na povezavah grafa se odločimo, kdaj bomo algoritem ustavili. Če so vse vrednosti povezav iz skupine i do skupine $i+1$ pod ε, razen na eni sami povezavi, kjer je vrednost feromona vsaj $1-\varepsilon$, potem gradnjo ustavimo. Za nominalne in ordinalne atribute je lahko $\varepsilon = 0$. Pri številskih atributih potrebujemo ε blizu 0. Vrednosti feromonov lahko konvergirajo po $\log_\omega(\varepsilon)$ iteracijah, ker zmanjšujemo vrednost σ_i, ki predstavlja standardni odklon posodobitvene funkcije

feromonov v enačbi 3.3, s faktorjem $\omega \in [0, 1]$. Gradnjo pravila ustavimo tudi, če se isto pravilo pojavlja v več iteracijah. Od učne množice pred začetkom učenja odstranimo 10 odstotkov za množico za ovrednotenje (evalvation set). Algoritem se ustavi, če začne klasifikacijska točnost padati na množici za ovrednotenje, ki se preverja z dodajanjem vsakega novega pravila v seznam pravil.

3.2 Empirično vrednotenje metode

Najprej smo poskušali poiskati privzete parametre metode. Našo metodo s privzetimi parametri smo nato primerjali z metodami Ant-Miner+, RIPPER, FURIA in CN2. Uporabili smo učne množice, pridobljene iz repozitorija UCI [42], in umetne množice z močnimi pogojnimi odvisnostmi med atributi. Na koncu smo v razdelku 3.2.4 različne metode primerjali še na realni medicinski domeni sinkopa.

Ker ima naša metoda več parametrov, je zaželeno, da za specifično učno množico poiščemo ustrezne parametre na validacijski množici. Kljub temu smo pokazali, da metoda zadovoljivo deluje s privzetimi vrednostmi parametrov. Primerjamo klasifikacijsko točnost, velikost seznama pravil, povprečno velikost pravila ter čas izvajanja.

3.2.1 Določanje privzetih vrednosti parametrov

Analizirali smo vpliv parametrov na klasifikacijsko točnost, velikost seznama pravil, število členov na pravilo ter čas izvajanja. Vsak test smo ponovili 10-krat z 10-kratnim prečnim preverjanjem na štirih učnih množicah. Tukaj smo uporabili veliko ponovitev za vsak test, da bi varianca algoritma čim manj vplivala pri določanju privzetih parametrov. Uporabljene učne množice za določitev privzetih parametrov so *Iris (iris)*, *Acute Inflammations (dia)*, *Congressional Voting Records (vot)* in *Japanese Credit Screening (japc)*, pridobljene iz repozitorija UCI Machine Learning [42]. Učne množice, uporabljene v tem podpoglavju, niso bile uporabljene v nadaljevanju testiranja, s čimer preprečimo nepošteno primerjavo, ko bi za našo metodo vnaprej poiskali dobro delujoče parametre. Učne množice smo izbrali tako, da vsebujejo številske atribute, ker je to poglavitna prednost metode nAnt-Miner glede na ostale predhodne razvite metode Ant-Miner. Učna množica *Iris* ima le številske atribute, druge učne množice pa vsebujejo različne tipe atributov. Testiranja so bila opravljena na več različnih računalnikih, zato se lahko časi izvajanja

razlikujejo tudi za enake vrednosti parametrov in časi niso popolnoma primerljivi med različnimi tabelami. Podani časi so povprečni časi za en pregib prečnega preverjanja.

Nastavili smo pet parametrov: število mravelj na iteracijo, vrednost uteži w iz enačbe (3.2), faktor ω za zmanjševanje širine posodabljanja feromonov σ, število elitnih mravelj in mejo za konvergenco. Vsak parameter smo testirali posebej. Privzete vrednosti so nastavljene na 500 mravelj za vsako iteracijo, meja za konvergenco je 50 iteracij, faktor izhlapevanja je $\rho = 0,85$, $\omega = 0,8$ (hitrost zmanjševanja širine posodabljanja pri številskih atributih), $w = 0,5$ (razmerje med pokritostjo in natančnostjo pravil pri ocenjevanju le-teh) ter 5 elitnih mravelj. Tabele od 3.1 do 3.5 prikazujejo povprečne rezultate, dobljene na vseh štirih podatkovnih množicah.

Tabela 3.1 prikazuje rezultate za različno število uporabljenih mravelj pri vsaki iteraciji algoritma. Metoda se upočasnjuje linearno z večanjem števila mravelj, kar je pričakovano, saj je potrebno simulirati vsak sprehod mravlje. Klasifikacijska točnost doseže optimum pri 1000 mravljah. Velikost seznama pravil se z večanjem števila mravelj linearno zmanjšuje. Število pravil se z večanjem števila mravelj manjša, ker je večja verjetnost, da bomo pri več poskusih že na začetku dobili pravilo z visoko pokrivnostjo. Rezultati kažejo, da z večanjem števila mravelj dobimo bolj razumljive (krajše) sezname pravil, vendar se s prevelikim številom mravelj manjša klasifikacijska točnost.

Tabela 3.2 prikazuje rezultate za različne vrednosti w iz enačbe (3.2), ki vpliva na to, ali želimo dobiti bolj pokrivna ali bolj natančna pravila. Nizke vrednosti pomenijo, da metoda daje večji poudarek pravilom z veliko pokritostjo; kot rezultat dobimo krajša pravila. Vrednosti blizu 1 pomenijo poudarek natančnosti pravila, kar pomeni, da dobimo večje število daljših pravil. Ko povečujemo vpliv natančnosti, se metoda začne pretirano prilagajati učnim podatkom. Poskusi kažejo, da je privzeta vrednost $w = 0,5$ smiselna. Klasifikacijska točnost očitno pade le, če se približamo ekstremnim vrednostim blizu 0 ali 1. Za preproste nešumne množice lahko uporabimo večji w, za ostale pa manjši, da se metoda ne prilagodi šumu.

Tabela 3.3 prikazuje rezultate za različne vrednosti parametra ω, ki prilagaja hitrost krčenja širine normalne porazdelitve, s katero posodabljamo poti pri številskih atributih. Rezultati kažejo, da metoda v splošnem na ta parameter ni občutljiva. Parameter ω smo vseeno ohranili v algoritmu, saj dopušča uporabniku iskanje bolj splošnih (ω blizu 1) ali bolj specifičnih pravil (ω blizu 0). Parameter tudi ne vpliva

Tabela 3.1
Povprečni rezultati metode nAnt-Miner za različno število mravelj na štirih domenah.

# mravelj	točnost	# pravil	členov/pravilo	čas (s)
25	88,92 ± 1,59	6,17	1,58	2
50	89,61 ± 1,30	6,24	1,57	3
100	89,26 ± 1,74	5,97	1,56	3
200	89,55 ± 1,08	5,72	1,55	4
500	89,74 ± 1,55	5,26	1,51	9
1000	90,40 ± 1,00	5,30	1,49	15
2000	89,64 ± 1,44	5,23	1,45	29
5000	89,89 ± 1,54	4,93	1,45	63
10000	89,31 ± 1,75	4,90	1,48	120

Tabela 3.2
Povprečni rezultati metode nAnt-Miner za različne vrednosti w na štirih domenah.

w	točnost	# pravil	členov/pravilo	čas (s)
0,00	75,24 ± 4,00	2,80	1,09	16
0,25	91,00 ± 1,62	5,15	1,57	33
0,50	92,08 ± 0,85	5,91	1,84	34
0,75	91,92 ± 0,89	7,02	2,07	39
1,00	83,45 ± 2,61	11,54	1,74	35

na časovno zahtevnost algoritma.

Tabela 3.4 prikazuje rezultate za različno število uporabljenih elitnih mravelj, ki jih uporabljamo za posodabljanje feromonov v vsaki iteraciji algoritma. Vrednost o pomeni, da lahko le globalno najboljša mravlja posodobi feromone. Rezultati so pokazali, da večje število elitnih mravelj poveča klasifikacijsko točnost na račun časa izvajanja. Za privzeto vrednost in nadaljnje teste smo izbrali 5 elitnih mravelj. Večanje števila elitnih mravelj nekoliko upočasnjuje delovanje algoritma. Upočasnjevanje algoritma z večanjem števila elitnih mravelj je pričakovano, saj mora metoda posodabljati več poti vsako iteracijo izvajanja. Hkrati se število pravil pri uporabi elitnih mravelj zmanjša, saj metoda več časa preiskuje prostor pravil, ker ohranja

Tabela 3.3
Povprečni rezultati metode nAnt-Miner za različne vrednosti ω na štirih domenah.

ω	točnost	# pravil	# členov/pravilo	čas (s)
0,05	90,03 ± 1,66	5,51	1,50	9
0,10	90,27 ± 1,35	5,47	1,52	9
0,15	90,28 ± 1,21	5,46	1,52	9
0,20	90,04 ± 1,27	5,43	1,51	9
0,25	89,75 ± 1,62	5,56	1,53	9
0,30	89,57 ± 0,95	5,38	1,51	9
0,35	89,77 ± 1,60	5,37	1,51	9
0,40	89,95 ± 1,43	3,98	1,50	9
0,45	90,33 ± 1,34	5,46	1,52	9
0,50	90,37 ± 1,17	5,44	1,49	9
0,55	90,24 ± 1,28	5,41	1,52	9
0,60	89,99 ± 1,37	5,34	1,51	9
0,65	90,35 ± 1,54	5,29	1,52	9
0,70	89,99 ± 1,30	5,38	1,51	9
0,75	89,65 ± 1,11	5,40	1,49	9
0,80	89,74 ± 1,55	5,48	1,51	9
0,85	90,05 ± 1,78	5,43	1,54	9
0,90	90,13 ± 1,41	5,24	1,52	9
0,95	90,16 ± 1,15	5,49	1,52	9
0,99	89,74 ± 1,79	5,41	1,53	9

feromon na več poteh v grafu.

Tabela 3.5 prikazuje različne vrednosti za mejo konvergence. Opazimo, da se dolžina seznama pravil krajša z večjo vrednostjo tega parametra in da se klasifikacijska točnost veča. Z večanjem tega parametra se linearno veča tudi izvajalni čas algoritma. Izbrana privzeta vrednost parametra je 10, kot kompromis med želeno točnostjo in časom izvajanja. Iz tega sledi, da je priporočljivo izbrati čim večjo mejo za konvergenco, če imamo na voljo dovolj časa, saj s tem dobimo točnejša in razumljivejša pravila. V tem primeru je to pričakovan rezultat, saj se metoda z večanjem vrednosti za mejo konvergence lažje izogne lokalnim optimumom, ker ima več poskusov, da se temu izogne.

Ugotovitve tega razdelka smo uporabili za nastavitev privzetih vrednosti naše

Tabela 3.4
Povprečni rezultati metode nAnt-Miner za različno število elitnih mravelj na štirih domenah.

elite	točnost	# pravil	# členov/pravilo	čas (s)
0	89,07 ± 1,59	6,24	1,53	8
1	88,84 ± 1,65	5,84	1,53	8
5	89,74 ± 1,55	5,48	1,51	9
10	89,81 ± 1,09	5,44	1,52	10
20	89,96 ± 1,09	5,53	1,52	10
30	89,80 ± 1,63	5,60	1,53	10
40	90,20 ± 1,10	5,56	1,53	11
50	89,39 ± 1,25	5,61	1,58	12
100	90,31 ± 1,22	5,62	1,57	17

Tabela 3.5
Povprečni rezultati metode nAnt-Miner za različne vrednosti meje konvergence na štirih domenah.

kon.	točnost	# pravil	# členov/pravilo	čas (s)
3	90,84 ± 1,07	6,65	1,78	11
5	91,85 ± 0,78	6,43	1,84	21
10	92,08 ± 0,85	5,92	1,84	40
50	92,34 ± 0,79	5,80	1,88	103
100	92,27 ± 0,84	5,82	1,87	177
200	92,37 ± 0,74	5,71	1,84	320

implementacije. Uporabljene vrednosti so: število mravelj 500, faktor izhlapevanja $\rho = 0,85$, $\omega = 0,8$, meja konvergence je nastavljena na 10 iteracij, število elitnih mravelj je 5. Parameter ρ smo izbrali med $\rho \in [0,85,\ 0,99]$. Prenizko izbran ρ, nižji od 0,85, bi pomenil prehitro izhlapevanje feromonov na že obiskanih poteh. Višja vrednost tega parametra vodi k večji prostorski kompleksnosti našega algoritma, kot je pojasnjeno v razdelku 3.1.3. Za število mravelj smo privzeli vrednost 500, kljub temu da daje vrednost 1000 nekoliko boljše rezultate, ker časovna zahtevnost algoritma linearno narašča s tem parametrom. Enak razlog smo uporabili pri nastavitvi privzete vrednosti parametra za konvergenco, saj želimo

obdržati dobro razmerje med časom izvajanja in klasifikacijsko točnostjo.

3.2.2 *Primerjava z ostalimi metodami*

Metodo nAnt-Miner smo primerjali na 11-ih učnih množicah, pridobljenih iz UCI Machine Learning repositorija [42]. Učne množice so bile izbrane glede na število številskih atributov. Izbrali smo nekaj množic brez številskih atributov, nekaj z mešanimi atributi in nekaj s samo številskimi atributi. S tem želimo pokazati, da metoda nAnt-Miner deluje neodvisno od tipa atributov podatkovne množice. Uporabljene učne množice so: Tic-Tac-Toe Endgame (*ttt*), Australian Credit Approval (*aus*), German Credit Data (*ger*), Balance Scale (*bal*), Teaching Assistant Evaluation (*tae*), Breast Cancer Wisconsin (*bcw*), Car Evaluation (*car*), Ripley dataset (*rip*), Contraceptive Method Choice (*cmc*), Glass Identification (*gla*) in Seeds (*seed*).

Tabela 3.6 prikazuje osnovne podatke izbranih množic. Stolpci prikazujejo število atributov, število številskih atributov, število učnih primerov in število razredov. Vsi testi so bili ponovljeni 5-krat z 2-kratnim prečnim preverjanjem. Metodo nAnt-Miner (nAM) smo primerjali z metodami: Ant-Miner+ [2] (AM+), RIPPER [5], FURIA [4] in CN2 [10]. Metoda Ant-Miner+ je izbrana, ker je metoda nAnt-Miner zasnovana na njej in je trenutno vodilna metoda za rudarjenje trdih pravil z uporabo kolonije mravelj. Metodi RIPPER in FURIA sta trenutno med najboljšimi metodami za rudarjenje odločitvenih pravil. RIPPER rudari trda pravila, FURIA pa mehka. CN2 spada med preprostejše preiskovalne algoritme in mnogokrat služi kot osnova za primerjavo klasifikacijskih pravil.

Tabela 3.6
Opis uporabljenih množic pri testiranju izbranih metod.

Množica	# at.	# št. at.	# prim.	# raz.
ttt	9	0	958	2
aus	14	6	690	2
ger	20	7	1000	2
bal	4	0	625	3
tae	5	1	151	3
bcw	10	0	699	2
car	6	0	1728	4
rip	2	2	1250	2
cmc	9	2	1473	3
gla	9	9	214	6
seed	7	7	210	3

Pri metodi Ant-Miner+ smo opravili teste s privzetimi vrednostmi, priporočenimi s strani avtorjev. Uporabili smo 1000 mravelj v vsaki iteraciji in omejili izvajanje na največ 200 iteracij za iskanje posameznega pravila. Hevristika za ocenjevanje pravila je razmerje med pokritostjo in zaupanjem. Faktor izhlapevanja smo pustili na privzeti vrednosti 0,85. Ker metoda Ant-Miner+ ne more implicitno obravnavati številskih atributov, smo atribute za to metodo predhodno diskretizirali z metodo Fayyad-Irani [3]. Za teste smo uporabili implementacijo, dostopno na www.antminerplus.com. Pri ostalih metodah smo uporabili privzete parametre metode.

Metodi nAnt-Miner smo nastavili število mravelj na 500, faktor izhlapevanja $\rho = 0,85$, $\omega = 0,8$, meja konvergence je nastavljena na 10, število elitnih mravelj na 5 in parameter w na 0,5. Parametri so izbrani glede na rezultate testov iz razdelka 3.2.1 in so uporabljeni tudi v nadaljevanju disertacije.

Tabela 3.7

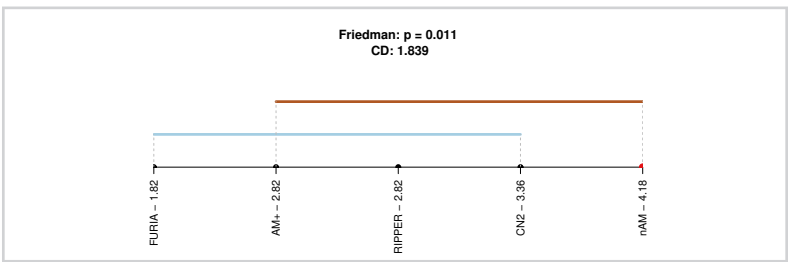
Dobljene klasifikacijske točnosti in njihovi standardni odkloni za izbrane metode na vseh 11-ih množicah. Zadnja vrstica predstavlja povprečni rang vsake metode in njegov standardni odklon.

Metoda	nAM	AM+	RIPPER	FURIA	CN2
ttt	97,68 ± 0,50 (3)	95,64 ± 4,75 (4)	98,12 ± 0,30 (1)	98,04 ± 0,12 (2)	78,65 ± 4,50 (5)
aus	76,72 ± 3,85 (5)	84,41 ± 1,03 (3)	84,96 ± 0,82 (1)	84,93 ± 0,35 (2)	77,41 ± 3,24 (4)
ger	67,62 ± 2,25 (4)	71,12 ± 0,83 (2)	70,10 ± 1,25 (3)	72,42 ± 0,83 (1)	66,97 ± 1,37 (5)
bal	45,57 ± 1,91 (5)	76,70 ± 2,00 (3)	73,18 ± 1,71 (4)	78,11 ± 1,45 (1)	77,25 ± 1,88 (2)
tae	41,72 ± 8,68 (2)	38,94 ± 4,12 (3)	38,28 ± 2,67 (4)	38,01 ± 1,59 (5)	54,29 ± 7,59 (1)
bcw	83,75 ± 4,76 (5)	94,74 ± 0,75 (1)	93,19 ± 0,60 (3)	93,33 ± 0,08 (2)	92,71 ± 1,51 (4)
car	68,92 ± 1,20 (5)	88,65 ± 2,02 (3)	83,19 ± 0,88 (4)	89,11 ± 0,66 (2)	93,36 ± 1,32 (1)
rip	88,24 ± 1,73 (3)	88,55 ± 0,92 (1)	87,98 ± 0,74 (4)	88,38 ± 0,50 (2)	69,27 ± 7,36 (5)
cmc	41,33 ± 1,09 (5)	45,66 ± 0,89 (4)	52,57 ± 1,08 (2)	53,52 ± 1,80 (1)	45,78 ± 1,80 (3)
gla	52,34 ± 10,4 (4)	51,03 ± 5,11 (5)	62,15 ± 1,78 (2)	62,24 ± 3,66 (1)	56,14 ± 5,35 (3)
seed	85,90 ± 3,93 (5)	88,67 ± 1,32 (2)	88,29 ± 1,29 (3)	91,62 ± 0,99 (1)	85,99 ± 4,28 (4)
rang	4,18 ± 1,08	2,82 ± 1,25	2,82 ± 1,17	1,82 ± 1,17	3,36 ± 1,50

Tabela 3.7 prikazuje dobljene klasifikacijske točnosti na 11-ih množicah. Zadnja vrstica v tabeli predstavlja povprečni rang uvrstitve vsake metode in standardni odklon uvrstitve. Rezultate metod smo po Demšar [43] testirali najprej s Friedmanovim testom. P-vrednost statističnega testa je 0,0105, s čimer lahko ovržemo ničelno hipotezo, da so vse metode enakovredne. Nato smo naredili še parni test Nemenyi med vsemi metodami. Dobljene p-vrednosti so prikazane v tabeli 3.8. Iz tabele je razvidno, da sta statistično različni samo metodi nAnt-Miner in FURIA. Metoda FURIA je ena od vodilnih metod za rudarjenje mehkih pravil, ostale štiri metode pa iščejo trda pravila. To daje metodi FURIA prednost. Grafična predstavitev rezultatov je vidna na sliki 3.2, kjer so s črto povezane skupine algoritmov, za katere ne moremo statistično trditi, da se med seboj razlikujejo. Kritična razlika za test Nemenyi je v tem primeru 1,839, ki pove, da se statistično razlikujejo metode, ki imajo razliko povprečnega ranga večjo od 1,839.

Tabela 3.8
P-vrednosti parnega testa Nemenyi.

	nAM	AM+	RIPPER	FURIA
AM+	0,2550	-	-	-
RIPPER	0,2550	1,0000	-	-
FURIA	0,0042	0,5734	0,5734	-
CN2	0,7435	0,9280	0,9280	0,1473



Slika 3.2
Grafični prikaz rezultatov testa Nemenyi.

Tabela 3.9 prikazuje povprečno dobljeno število pravil (R), povprečno število členov na pravilo (T) in povprečno število členov celotnega seznama (S). V osmih

primeri izmed enajstih ima najmanjše povprečno število členov v celotnem seznamu metoda Ant-Miner+. S primerjavo metod Ant-Miner+ in nAnt-Miner vidimo, da ima nAnt-Miner v povprečju krajša pravila (T) kot metoda Ant-Miner+. To nakazuje, da so posamezna pravila metode nAnt-Miner razumljivejša, vendar zato rudari daljše sezname. Metoda RIPPER ravno tako išče krajša pravila, vendar dobi večje sezname kot Ant-Miner+. Metodi FURIA in CN2 imata daljša pravila in več pravil na seznam v primerjavi z ostalimi tremi metodami. Največji skok v številu pravil glede na metodo Ant-Miner+ imata metodi nAnt-Miner in RIPPER pri množici *car*. Ta množica odstopa od ostalih po tem, da ima več učnih primerov in večje število razredov. Ker pri metodah RIPPER, FURIA in CN2 zaradi uporabljenih implementacij v programu Weka [44] ali Orange [45] nismo dobili seznama pravil za vsak pregib prečnega preverjanja, je pri teh metodah povprečje izračunano le na končnih seznamih za vsak test, kar lahko vpliva na kakovost rezultatov.

Tabela 3.9

Število pravil (R), število členov (T), velikost seznama pravil (S) pri primerajnju vseh metod.

Metoda	nAM			AM+			RIPPER			FURIA			CN2		
	R	T	S	R	T	S	R	T	S	R	T	S	R	T	S
ttt	11,8	2,7	31,86 (3)	8,0	3,07	24,56 (1)	9,4	2,91	27,35 (2)	21,6	3,51	75,82 (4)	119,3	2,29	272,71 (5)
aus	9,4	2,9	27,26 (4)	3,0	2,27	6,81 (2)	3,2	1,69	5,41 (1)	8,4	2,83	25,06 (3)	64,1	1,54	98,71 (5)
ger	5,5	3,0	16,50 (3)	2,3	4,14	9,52 (1)	4,2	2,38	10,00 (2)	9,8	2,27	22,25 (4)	78,5	2,84	223,17 (5)
bal	9,4	2,2	20,68 (2)	5,7	3,53	20,12 (1)	15,0	2,05	30,75 (3)	46,8	2,35	109,98 (4)	182,7	1,74	318,54 (5)
tae	20,7	1,0	20,70 (3)	9,4	2,06	19,36 (2)	5,4	7,80	42,12 (4)	9,0	1,31	11,79 (1)	36,3	1,96	71,09 (5)
bcw	8,3	1,6	13,28 (2)	2,6	1,88	4,89 (1)	12,4	1,44	17,86 (3)	22,8	1,99	45,37 (5)	24,5	1,72	42,05 (4)
car	19,3	4,8	92,64 (2)	6,6	4,75	31,35 (1)	40,6	3,89	157,93 (3)	92,6	4,38	405,59 (5)	79,5	2,06	163,39 (4)
rip	11,1	1,7	18,87 (4)	2,4	1,85	4,44 (1)	6,0	2,13	12,78 (2)	7,2	1,97	14,18 (3)	190,9	2,01	384,06 (5)
cmc	4,6	4,2	19,32 (2)	4,4	5,58	24,55 (3)	4,2	2,71	11,38 (1)	10,6	2,36	25,02 (4)	369,3	3,22	1190 (5)
gla	9,0	2,4	21,60 (3)	3,7	3,09	11,43 (1)	8,6	2,28	19,61 (2)	17,0	3,26	55,42 (4)	36,8	1,59	58,53 (5)
seed	6,4	1,7	10,88 (3)	3,4	2,15	7,31 (1)	5,0	1,80	9,00 (2)	8,2	2,41	19,76 (5)	12,8	1,29	16,54 (4)

Tabela 3.10
Izvajalni čas metod Ant-Miner+ in nAnt-Miner

Množica	ttt	aus	ger	bal	tae	bcw
AM+	30	48	52	47	63	91
nAM	15	260	326	14	13	109
Množica	car	rip	cmc	gla	seed	
AM+	154	4	30	77	20	
nAM	197	162	116	141	61	

Tabela 3.10 prikazuje čase izvajanja metod nAnt-Miner in Ant-Miner+. Ostale metode so veliko hitrejše, ker ne uporabljajo kolonije mravelj za preiskovanje. Časi so merjeni v sekundah za eno prečno preverjanje. Opazimo, da nobena metoda ni boljša na vseh množicah, ampak da so časi izvajanja odvisni od posamezne množice. Množica *rip* je posebno zahtevna za metodo nAnt-Miner, saj pri tej množici algoritem potrebuje veliko iteracij, preden se ustavi. Enako velja tudi za množici *aus* in *ger*. Množice *ttt*, *bal* in *tae* se z metodo nAnt-Miner obdelajo hitreje. Na dveh izmed teh množic (*ttt* in *tae*) nAnt-Miner dosega tudi boljšo klasifikacijsko točnost kot Ant-Miner+. To nakazuje, da je za ti dve množici metoda nAnt-Miner primernejša. Pri množici *bal* dobi nAnt-Miner nizko klasifikacijsko točnost. Razlog za to je v tem, da ima ta množica močno odvisnost med atributi v obliki množenja atributov. V članku [7] smo z desetkratnim prečnim preverjanjem na tej množici dobili boljše rezultate, kar nakazuje, da je imela metoda nAnt-Miner zaradi dvakratnega prečnega preverjanja na voljo premalo učnih primerov, da bi zaznala to odvisnost.

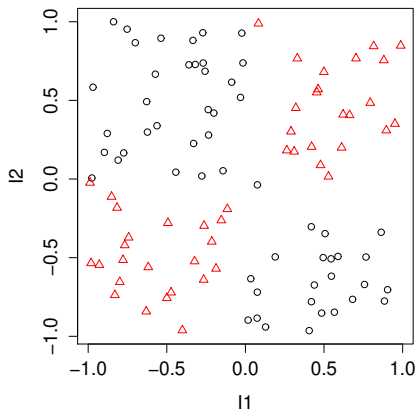
3.2.3 Umetne podatkovne množice

Da smo preverili, če metoda nAnt-Miner zazna pogojne odvisnosti med atributi, smo metodo testirali tudi na štirih umetnih problemih z močnimi pogojnimi odvisnostmi med atributi. Množice so pridobljene iz članka avtorjev Robnik-Šikonja in Kononenko [46]. Osnovni podatki množic so predstavljeni v tabeli 3.11.

Tabela 3.11
Opis umetnih množic

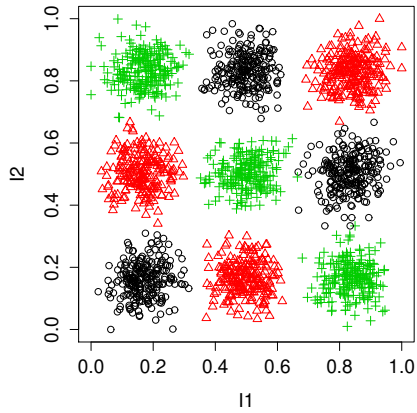
množica	# atributov	# primerov	# razredov
xor	2	100	2
chess	4	100	2
modGroup	4	100	3
xorCross	6	100	2

xor Množica XOR je sestavljena iz dveh številskih atributov z vrednostmi med -1 in 1. Če sta vrednosti atributov nad ali pod 0 hkrati, potem primer pripada pozitivnemu razredu, drugače negativnemu razredu. Množico prikazuje slika 3.3.



Slika 3.3
Prikaz podatkovne množice *xor*. Trikotniki in krogi prikazujejo dva različna razreda.

groups Na sliki 3.4 sta prikazana dva pomembna atributa I_1 in I_2, ter razred. Vrednosti so razpršene okoli centrov skupin, ki predstavljajo razred. Množici sta dodana tudi dva atributa, ki nista v povezavi z razredom. Primeri, ki pripadajo razredu 0, 1 in 2, so prikazani s krogi, trikotniki in znaki plus v tem vrstnem redu.



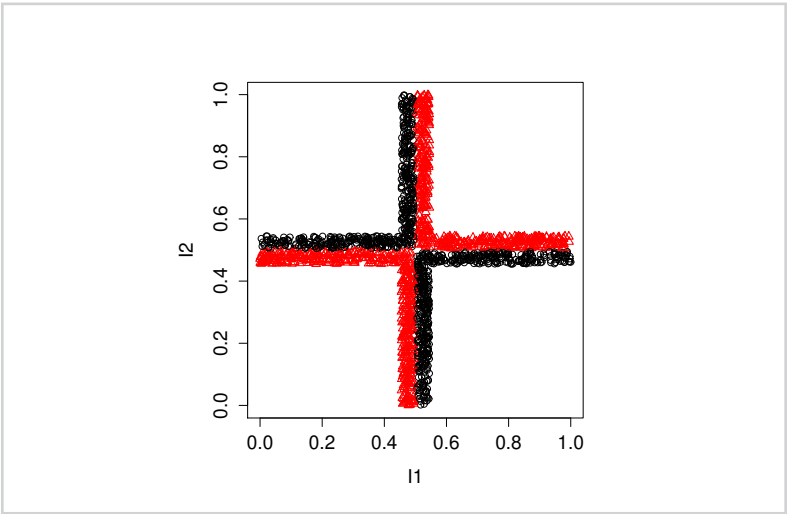
Slika 3.4
Prikaz dveh pomenbnih atributov na množici *groups*.

cross Slika 3.5 prikazuje dva pomembna atributa, I_1 in I_2 množice. Razred 1, označen z rdečimi trikotniki, ustreza pogoju $(I_1 - 0,5)(I_2 - 0,5) > 0$. Množica ima še štiri atribute, ki ne vplivajo na razred.

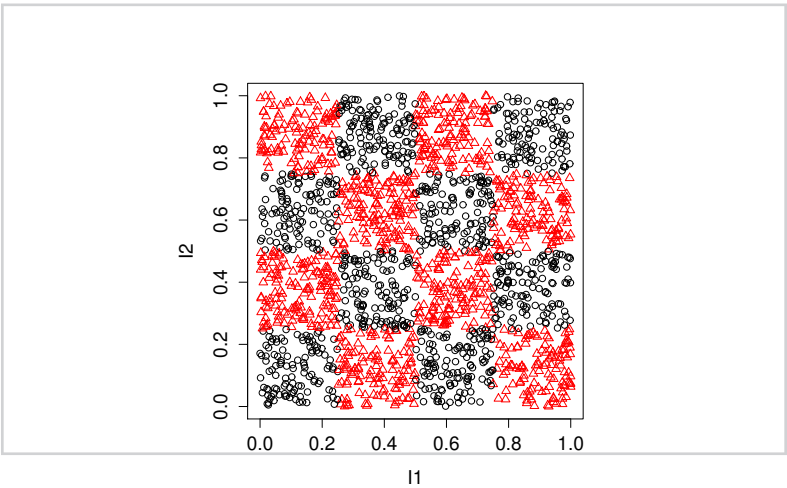
chess Na sliki (3.6) sta prikazana dva pomembna atributa I_1 in I_2, ter razred. Množica je sestavljena iz 4×4 šahovnice, kjer črni krogi predstavljajo razred 0 in rdeči trikotniki razred 1. Množica ima tudi dva atributa, ki ne vplivata na razred.

Tabela 3.12
Klasifikacijske točnosti na umetnih množicah z močno pogojno odvisnostjo med atributi.

Metoda	nAM	AM+	RIPPER	FURIA
xor	96,80 ± 1,93	51,40 ± 6,23	88,90 ± 3,11	98,00 ± 0,71
chess	58,20 ± 7,15	47,70 ± 5,37	60,60 ± 2,07	55,8 ± 3,77
modGroup	65,20 ± 9,85	31,40 ± 1,95	42,40 ± 10,90	37,40 ± 9,84
xorCross	89,00 ± 8,12	47,70 ± 5,37	84,40 ± 10,92	89,80 ± 0,45



Slika 3.5
Prikaz pomembnih atributov na množici *cross*.



Slika 3.6
Prikaz množice *chess*.

Tabela 3.12 prikazuje klasifikacijske točnosti, dobljene na umetnih podatkovnih množicah. Vidimo, da metoda Ant-Miner+ deluje slabše v primerjavi z ostalimi primerjanimi metodami na vseh štirih množicah. Razlog je predvsem v predhodni diskretizaciji z metodo Fayyan-Irani, ki ne zazna odvisnosti med atributi in zato

napačno postavi meje. V dveh primerih ima metoda FURIA najvišjo klasifikacijsko točnost, na teh dveh množicah je tudi metoda nAnt-Miner uspešna. Metodi nAnt-Miner in Ripper imata vsaka v enem primeru najvišjo klasifikacijsko točnost, vendar metoda nAnt-Miner dosega veliko boljše rezultate na množici *modGroup*, kar nakazuje na to, da je edina metoda, ki je v tem primeru zaznala močne odvisnosti med atributi. Množica *modGrop* se od ostalih razlikuje po tem, da ima tri različne razrede in ne samo dva kot ostale. Kadar slutimo, da podatki vsebujejo močne pogojne odvisnosti med atributi, priporočamo uporabo metode nAnt-Miner. S pravilno predhodno diskretizacijo atributov, bi tudi metoda Ant-Miner+ zaznala močne odvisnosti med atributi in dosegla boljše rezultate. Zaznavanje pogojne odvisnosti med atributi je mogoče zaradi oblike preiskovalnega grafa pri obeh metodah nAnt-Miner in Ant-Miner+. Tukaj ima nAnt-Miner prednost pred Ant-Miner+, ker lahko avtomatsko zazna pravilne meje za diskretizacijo in uporabnik ne potrebuje predznanja o dotični učni množici.

3.2.4 Medicinska domena

Metodo smo preizkušali še na realni podatkovni množici *syncope*. Množica vsebuje informacije pri izvajanju testa z nagibno mizo (tilt-table) za 94 pacientov. Za vsakega pacienta so bile meritve krvnega tlaka in EKG-ja preoblikovane v standardno BrS (Brugada EKG vzorec) in frekvenčne EKG indikatorje (nizek, visok in skupen). Za vsakega pacienta imamo 11 meritev za vsak indikator. Meritve so bile opravljene vsakih 5 minut. Razred prikazuje, ali je test povzročil sinkopo, kar pomeni, da je pacient omedlel oziroma izgubil zavest. Test smo opravili na dveh različicah podatkov. Prvega smo poimenovali *syncope123* in vsebuje le meritve prvih 15 minut testa. Drugi test *syncope* vsebuje vse meritve. Prvi test je še posebno zanimiv, ker bi dobri rezultati tega testa zmanjšali čas testiranja pacientov. Vsi testi so bili pognani s privzetimi nastavitvami metod.

Tabela 3.13 prikazuje klasifikacijske točnosti, dobljene z izbranimi metodami. Na tej domeni se najbolje izkaže metoda Ant-Miner+, ki na obeh množicah doseže najboljši rezultat. V iskanju razloga, zakaj metoda nAnt-Miner doseže nizko klasifikacijsko točnost na množici *syncope*, smo jo preizkusili tudi z 10-kratnim prečnim preverjanjem. V tem primeru je klasifikacijska točnost narasla na 64,52. Razlog je v tem, da metoda potrebuje več podatkov, da se nauči pravil. Iz tabele 3.14 vidimo, da ima metoda Ant-Miner+ kratka pravila na množici

syncope123, bolj natančno rezultat 1.0 pomeni, da je vedno uporabljeno le privzeto pravilo. To kaže na to, da je na tej množici večinski klasifikator dober in da se druge metode z dodajanjem pravil preveč prilagajajo šumu v podatkih. Pri metodi FURIA klasifikacijska točnost z dodajanjem pravil ne pada tako izrazito, ker gradi mehka pravila. Iz tabele 3.15 je razvidno, da je najpočasnejša metoda nAnt-Miner. Metoda Ant-Miner+ ima v prvem primeru nizek čas, ker gradi le privzeta pravila.

Tabela 3.13
Klasifikacijske točnosti na medicinski domeni.

Metoda	nAM	AM+	RIPPER	FURIA
syncope123	47,13 ± 6,27	53,78 ± 17,0	40,64 ± 5,81	43,01 ± 5,99
syncope	50,74 ± 7,65	64,25 ± 10,6	62,58 ± 2,33	61,08 ± 2,45

Tabela 3.14
Velikost pravil na medicinski domeni. Podana je vrednost S, ki je zmnožek števila pravil in povprečnega števila členov na pravilo.

Metoda	nAM	AM+	RIPPER	FURIA
syncope123	8,55	1,00	1,6	9,2
syncope	8,46	6,21	3,2	16

Tabela 3.15
Čas izvajanja metod za 5 × 2 prečno preverjanje v sekundah.

Metoda	nAM	AM+	RIPPER	FURIA
syncope123	53	<1	<1	<1
syncope	315	22	<1	<1

3.2.5 Zaključek

Analiza parametrov in empirično vrednotenje metode nakazuje, da je največja prednost metode nAnt-Miner pred Ant-Miner+ v tem, da implicitno zazna močne odvisnosti med atributi, kot kaže razdelek 3.2.3. Ker je metoda nAnt-Miner zasnovana na metodi Ant-Miner+, bi s pravilno podanimi diskretnimi mejami tudi metoda

Ant-Miner+ dosegla višjo klasifikacijsko točnost, vendar se pri metodi nAnt-Miner končnemu uporabniku prihrani korak izbire primerne diskretizacije.Slabost metode sta nizka klasifikacijska točnost - če uporabimo premalo učnih primerov - in njen čas izvajanja. Ker smo želeli za to delo zmanjšati čas izvajanja metode nAnt-Miner na nekaj minut za vsako množico, smo uporabili 5 × 2 prečno preverjanje. Rezultati primerjav v članku avtorjev Pičulin in Robnik-Šikonja [7] so dobljeni z 10 × 10 prečnim preverjanjem in se zato nekoliko razlikujejo od teh v disertaciji. Predvsem se izkaže, da je metoda bolj primerljiva z ostalimi glede na klasifikacijsko točnost. Glede na prednosti in slabosti metode priporočamo uporabo metode nAnt-Miner na množicah, za katere sumimo, da imajo močno odvisnost med atributi, pri tem moramo seveda imeti na razpolago dovolj učnih primerov.

4

Metoda ProAnt-Miner

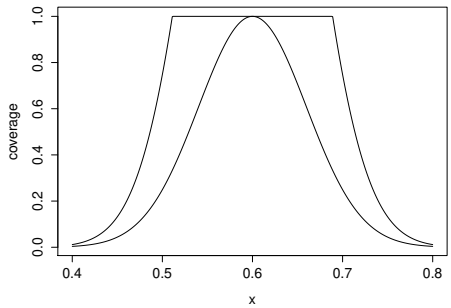
Metoda ProAnt-Miner (PRObalistic ANT-based rule miner) se razlikuje od obstoječih metod za učenje pravil, ker sočasno gradi mehke intervale in gradi verjetnostna pravila. Oblika pravil, ki jih rudari metoda ProAnt-Miner, je podrobno opisana v razdelku 4.1, uporaba pravil s to obliko pri napovedovanju v razdelku 4.2 ter razširitev te oblike pravil v razdelku 4.3. Delovanje metode je opisano v razdelku 4.4, njeno ovrednotenje pa v razdelku 4.5.

4.1 Oblika pravil metode ProAnt-Miner

Odločitvena pravila zgrajena z metodo ProAnt-Miner so v obliki IF-THEN pravil. Členi pravila za nominalne attribute so oblike $X = v_i$, kjer je X ime atributa, v_i pa je i-ta vrednost tega atributa. Členi pravila za številske atribute imajo novo predstavitev pravil, v obliki porezane Gaussove krivulje. Primer pravil je:

$$\begin{array}{llll} \text{IF } X_1{=}N(12,8,\ 1,6) & \text{AND } X_2{=}N(14,5,\ 1,9) & \text{THEN } \Delta \\ \text{IF } X_1{=}N(4,5,\ 0,9) & \text{AND } X_2{=}N(13,5,\ 1,6) & \text{THEN } \circ \\ \text{IF } X_1{=}N(13,2,\ 1,6) & \text{AND } X_2{=}N(1,4,\ 1,6) & \text{THEN } \circ \\ \text{IF } X_1{=}N(2,3,\ 1,6) & \text{AND } X_2{=}N(1,2,\ 1,6) & \text{THEN } \Delta \end{array} \tag{4.1}$$

Pravila so zgrajena za podatke, prikazane na sliki 4.1. Podatki predstavljajo dva številska atributa X_1 in X_2, ki sta med seboj pogojno odvisna glede na razred. Če sta vrednosti obeh atributov pod 7,5, ali če sta oba atributa nad to mejo, potem primer pripada razredu Δ, sicer razredu $\circ$. Slika 4.1 prikazuje pravila iz zbirke (4.1). $N(\mu, \sigma)$ v členih predstavlja Gaussovo porazdelitev s središčem μ in standardnim odklonom σ. Na sliki 4.1 so ta pravila prikazana z elipsami, kjer izohipse predstavljajo normalno porazdelitev pri standardnih deviacijah $\sigma = 1, 2$ in 3. Člen $X = N(\mu, \sigma)$ lahko interpretiramo kot; X je blizu μ. V našem primeru to pomeni, če je X_1 blizu 12,8 in X_2 blizu 14,5 potem primer pripada razredu Δ. Vrednosti σ določajo pomen besede blizu. Manjša, kot je σ, bližje vrednosti μ mora biti primer, da ga pravilo pokrije.



Slika 4.1
Primer Gaussove distribucije in mehkega pravila, ki ga iz nje izpeljemo.

4.2 Napovedovanje z verjetnostnimi pravili

Naj bo pravilo R_i iz seznama pravil dolžine s

$$\text{IF } T_{i1} \text{ AND } T_{i2} \text{ AND } \ldots \text{AND } T_{it} \text{ THEN } l_i, \tag{4.2}$$

kjer je R_i i-to pravilo v seznamu. T_{ij} je j-ti člen pravila R_i, l_i pa razred, ki pripada množici razredov L. Člen T_{ij} pokrije primer k s pokritjem $c_{i,j,k}$. Če člen T_{ij} trenutno predstavlja nominalni atribut oblike $A_f = v$, potem je $c_{i,j,k} = 1$, če ima primer k vrednost atributa $A_f = v$, sicer pa o. Če člen T_{ij} predstavlja številski atribut oblike $A_f = N(\mu_{ij}, \sigma_{ij})$, potem člen T_{ij} pokrije primer k, ki ima vrednost v_k z vrednostjo:

$$c_{i,j,k} = \begin{cases} 2\,\mathrm{cdf}(v_k, \mu_{ij}, \sigma_{ij}) & \text{če } v_k < \mu_{ij}, \\ 2\,\mathrm{cdf}(2\mu_{ij} - v_k, \mu_{ij}, \sigma_{ij}) & \text{če } v_k \geq \mu_{ij}, \end{cases} \tag{4.3}$$

kjer je $\mathrm{cdf}(x, \mu, \sigma)$ kumulativna distribucijska funkcija definirana kot:

$$\mathrm{cdf}(x, \mu, \sigma) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - \mu}{\sigma\sqrt{2}}\right)\right), \tag{4.4}$$

pri čemer je $\mathrm{erf}(x)$ funkcija napake definirana kot:

$$\mathrm{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2}\,\mathrm{d}t. \tag{4.5}$$

Ta postopek priredi večjo verjetnost primerom blizu centrov Gaussovih krivulj in manjšo verjetnost bolj oddaljenim primerom. Vsako pravilo pokrije primer k z verjetnostjo:

$$c_{i,k} = \prod_{j=1}^{t} c_{i,j,k} \tag{4.6}$$

Vsak seznam pravil pokrije primer k z razredom l_c z vrednostjo pokritosti:

$$c_k^{l_c} = \sum_{i \in \{x | l_x = l_c\}} c_{i,k} \tag{4.7}$$

Za vsak razred imamo svojo pokritost. Verjetnost, da primer k pripada razredu l_c, je torej:

$$p_k^{l_c} = \frac{c_k^{l_c}}{\sum_{m=1}^{|L|} c_k^{l_m}} \tag{4.8}$$

Napovedan je razred z največjo verjetnostjo.

4.3 Mehka pravila z večjim pokritjem

Vrednosti, ki jih dobimo z enačbo (4.7) so navadno nizke, saj med seboj množimo majhne vrednosti. Ker hočemo, da pravilo vsaj nekatere primere bolj ali celo popolnoma pokrije, vsako Gaussovo krivuljo množimo s parametrom λ, kjer je produkt navzgor omejen. Kot rezultat dobimo mehka pravila, podobna trapezom z gladkim prehodom med pokritjem in nepokritjem. Slika 4.1 prikazuje normalno distribucijo z $\mu = 0,6$ in $\sigma = 0,06$ in mehko trapezno obliko, pri čemer smo uporabili $\lambda = 3$. Na ta način pravilo popolnoma pokrije primere, ki padejo med 0,512 ter 0,688. Ta oblika se nekoliko približa trapezoidni obliki mehkih pravil, kot jih rudari metoda FURIA [4].

4.4 ProAnt-Miner

ProAnt-Miner je metoda za rudarjenje pravil, kot so opisana v razdelku 4.3. Metoda uporablja princip loči in obvladaj [23]. To pomeni, da algoritem poišče eno

pravilo in nato vse primere, ki so pokriti, odstrani iz učne množice. Na preostanku primerov išče novo pravilo, ki ga bo dodal med dobljena pravila. Ker oblika predlaganih pravil ne pokrije nujno celotnih primerov, so le-ti uteženi z deležem še nepokritega primera. Utež 1 pomeni, da je primer popolnoma nepokrit, utež 0 pa, da je popolnoma pokrit. Ko algoritem pokrije dovolj velik delež primerov, se ustavi in vrne dobljeni seznam pravil.

Metoda uporablja pristop optimizacije s kolonijo mravelj, kjer distribucija feromonov predstavlja obliko mehkih pravil. Mravlje ponavljajo pohod, dokler distribucije feromonov ne konvergirajo. Iz oblike feromonov pri konvergenci sklepamo na obliko pravila.

Okvirna ideja je prikazana v pseudokodi algoritma 4. Vrstici 1-2 inicializirata učno množico (TS) in seznam pravil (RL). Glavna zanka *while* v vrsticah 5-22 zgradi celoten seznam pravil. Glavna zanka inicializira feromone v vrstici 6. Zanka *while* v vrsticah 8-19 za vsak razred zgradi n pravil in nato najboljše pravilo v vrstici 17 shrani v R_{best} ter nato na podlagi kakovosti tega pravila v vrstici 18 posodobi vrednosti feromonov v grafu. Dvojna zanka *for* v vrsticah 9-16 za vsak razred zgradi n pravil. Vrstica 12 simulira sprehod mravlje, ki zgradi pravilo $R_{i,l}$, vrstica 13 to pravilo obreže, vrstica 14 pa izračuna kakovost tega pravila. Ko feromoni zanke *while* v vrsticah 8-19 konvergirajo v R_{best} ostane najboljše pravilo te iteracije, ki se nato v vrstici 20 doda v seznam pravil RL. S tem pravilom v vrstici 21 zmanjšamo uteži pokritosti za vsak učni primer. Vrstica 23 vrne končni seznam pravil.

4.4.1 Inicializacija feromonov

Vsak atribut je predstavljen s svojo distribucijo feromonov. Vrednosti feromonov ležijo med $\tau_{min} = 0$ in $\tau_{max} = 1$. Nominalni atributi imajo število košev pri aproksimaciji distribucije enako številu različnih vrednosti, ki jih lahko zavzemajo. Za številske atribute lahko število košev določimo sami s podanim parametrom b. Empirično priporočena vrednost je $b = 128$, saj želimo dovolj podrobno razdelitev za izračun empirične kumulativne distribucijske funkcije (ecdf).

Pri nominalnih atributih vsak koš predstavlja eno od vrednosti v atributa. Začetna vrednost v tem košu je relativna frekvenca vrednosti v v podatkih. Za številske atribute zgradimo ekvidistančni histogram, vrednost vsakega koša pa je frekvenca podatkov iz intervala, ki ga predstavlja koš.

Algorithm 4 : ProAnt-Miner.
Input: dataset, number of ants, number of bins, minCover, α, λ
Output: set of rules

```
1: TS ← {All training instances}
2: RL ← {}
3: Initialize weights of instances to 1
4: //Find a rule set
5: while not stopping criterion do
6:    Initialize pheromones (τ)
7:    //Find a single rule
8:    while pheromones not converged do
9:       for each class l do
10:         //Find a rule candidate
11:         for i in 1...n ants do
12:            Let an ant construct a rule R_{i,l}
13:            Refine and prune the rule R_{i,l}
14:            Calculate the quality of the rule R_{i,l}
15:         end for
16:      end for
17:      Find best rule R_best based on the rule quality
18:      Use R_best to update pheromones
19:   end while
20:   Add R_best to RL
21:   Update weights of instances based on R_best
22: end while
23: return RL
```

Feromoni na začetku predstavljajo distribucijo vrednosti atributov. Visoke vrednosti feromonov usmerjajo iskanje v smeri večje gostote, kar povzroči, da algoritem išče bolj splošna pravila.

4.4.2 Gradnja pravil

Vsak obhod mravlje zgradi eno pravilo. Za vsak razred posebej se zgradi n pravil, kjer je n vhodni parameter. Vsaka mravlja izbere naključni vrsti red, v katerem bo obiskala atribute in sproti gradi pravilo z dodajanjem členov. Permutacija izbire vrstnega reda je potrebna, ker je vsak dodani člen odvisen od trenutno zgrajenega pravila.

Ko mravlja izbere atribut A_f, uporabi njegovo distribucijo feromonov P_f, da zgradi prvi člen. Ker so vrednosti feromonov normalizirane na $[0, 1]$, je vrednost vsakega koša hkrati tudi verjetnost, da bo ta koš izbran. Če je A_f nominalen atribut in mravlja izbere koš z vrednostjo v, se zgradi člen $A_f = v$.

Če je A_f numeričen atribut, se zgradi pravilo oblike $A_f = N(\mu, \sigma)$. Vrednost μ je srednja vrednost koša, ki ga izbere mravlja. Vrednost σ je na začetku izračunana z enačbo (4.9), kjer sta $\max(A_f)$ in $\min(A_f)$ maksimalna in minimalna vrednost atributa A_f. Konstanta 100 v enačbi je izbrana empirično in je dovolj velika, da zagotavlja, da člen na začetku pokrije le majhen del podatkov. σ se nato povečuje, dokler s tem narašča tudi kakovost pravila. Če se po večkratnem povečanju σ-me (npr. 30-krat) pravilo še vedno izboljšuje, ta člen odstranimo, saj v tem primeru distribucija $N(\mu, \sigma)$ pokrije vse vrednosti atributa, kar pomeni, da je člen neuporaben v tem pravilu.

$$\sigma = \frac{\max(A_f) - \min(A_f)}{100} \tag{4.9}$$

Postopek ponavljamo, dokler ne dodamo vseh členov v pravilo. Pravilo nato izboljšamo tako, da iterativno širimo širino σ oziroma odstranjujemo nominalne člene, če s tem pridobimo boljšo kakovost pravila. S tem posplošujemo dobljena pravila.

S tem postopkom vsaka mravlja zgradi svoje pravilo. Ko so vsa pravila zgrajena, izberemo pravilo R_{best} z najvišjo kakovostjo in z njim posodobimo vrednosti feromonov, kot je opisano v razdelku 4.4.5.

Postopek ponavljamo, dokler se vrednosti feromonov pri vseh atributih ne stabilizirajo. Za detekcijo stabilizacije številskih atributov uporabljamo Hellingerjevo razdaljo med trenutno vrednostjo feromonov in vrednostjo feromonov v prejšnji iteraciji. Hellingerjeva razdalja je izračunana z enačbo

$$H(P_i, P_{i-1}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{k=1}^{b} \left(\sqrt{p_{i,k}} - \sqrt{p_{i-1,k}} \right)^2}, \tag{4.10}$$

kjer sta $P_i = (p_{i,1}, p_{i,2}, ..., p_{i,b})$ in $P_{i-1} = (p_{i-1,1}, p_{i-1,2}, ..., p_{i-1,b})$ diskretni distribuciji feromonov v trenutni in prejšnji iteraciji za izbrani atribut. Metoda se ustavi, ko je $H(P_i, P_{i-1}) < \text{minH}$ za vsak atribut. Vrednost $minH$ je določena kot parameter, s privzeto vrednostjo 0,05. Ko se algoritem zaključi, se zadnje pravilo R_{best} doda v seznam pravil.

4.4.3 *Izboljševanje pravil*

Vsaka mravlja na začetku zgradi pravilo R_i oblike IF T_{i1} AND T_{i2} AND ... AND T_{it} THEN l_i. To pravilo skušamo posplošiti. Ko gradimo pravilo R_i, najprej določimo člen T_{i1} in nato T_{i2}. Pri gradnji člena T_{i2} že upoštevamo, da je člen T_{i1} zmanjšal preiskovalni prostor. Ko dodamo T_{i2}, tudi ta člen zmanjša prostor, a tega zmanjšanja člen T_{i1} pri gradnji ni upošteval. Zato iterativno poskušamo povečati σ vseh členov, dokler se kakovost pravila še izboljšuje. V tem postopku lahko nekatere člene tudi odstranimo. Iterativno izboljševanje pravila smo prikazali v algoritmu 5.

4.4.4 *Kakovost pravil*

Kakovost Q pravila R_i je lahko izračunana na več načinov. Najpogosteje uporabljena mera za oceno kakovosti pravila pri klasifikatorjih, ki uporabljajo kolonijo mravelj, je:

$$Q = Pokritost(R_i) \cdot \alpha + Zaupanje(R_i) \cdot (1 - \alpha), \tag{4.11}$$

kjer je α vhodni parameter z vrednostmi med 0 in 1. Funkcija Pokritost(R_i) vrne pokritost pravila R_i, ki je definirana kot delež števila pokritih primerov s tem pravilom. Funkcija Zaupanje(R_i) je zaupanje v pravilo R_i, definirana kot razmerje med pravilno pokritimi primeri in vsemi pokritimi primeri. Vsak primer je zaradi podane oblike pravil le delno pokrit. Pokritost člena $c_{i,j}$ je izračunana kot vsota delnih pokritosti:

Algorithm 5: Rule refinement.
Input: rule, dataset
Output: improved rule

```
change ← TRUE
while change = TRUE do
    change ← FALSE
    for each term t in the rule do
        increase σ of the term
        if rule improved on dataset then
            change ← TRUE
            if σ increased 20 times then
                remove term t
            end if
        end if
    end for
end while
```

$$c_{i,j} = \sum_{k=0}^{a} c_{i,j,k} \ , \tag{4.12}$$

kjer k teče preko vseh a atributov.

Pokritost(R_i) pravila R_i je zmnožek pokritosti njegovih členov:

$$\text{Pokritost}(\mathrm{R}_i) = \prod_{j=0}^{t} c_{i,j} \tag{4.13}$$

Zaupanje(R_i) pravila R_i izračunamo kot:

$$\text{Zaupanje}(\mathrm{R}_i) = \frac{\prod_{j=0}^{t} \sum_{k=0}^{a} c_{i,j,k}^{\mathrm{l}_c}}{\text{Pokritost}(\mathrm{R}_i)}, \tag{4.14}$$

kjer je $c_{i,j,k}^{\mathrm{l}_c}$ definiran kot:

$$c_{i,j,k}^{\mathrm{l}_c} = \begin{cases} c_{i,j,k} & \text{če } i \in \{x; \mathrm{l}_x = \mathrm{l}_c\}, \\ 0 & \text{sicer}, \end{cases} \tag{4.15}$$

kar pomeni, da so upoštevana samo pokritja primerov pravilnega razreda.

Do sedaj opisan postopek ne upošteva uteži učnih primerov. Vsak učni primer ima za vsak razred svojo utež $w_k^{l_i}$. Z upoštevanjem uteži učnih primerov se enačba (4.12) spremeni v enačbo (4.16) ter enačba (4.14) v enačbo (4.17).

$$c_{i,j} = \sum_{k=0}^{a} c_{i,j,k} \cdot w_k^{l_i} \tag{4.16}$$

$$\text{Zaupanje}(\mathrm{R}_i) = \frac{\prod_{j=0}^{t} \sum_{k=0}^{a} c_{i,j,k}^{l_i} \cdot w_k^{l_c}}{\text{Pokritost}(\mathrm{R}_i)} \tag{4.17}$$

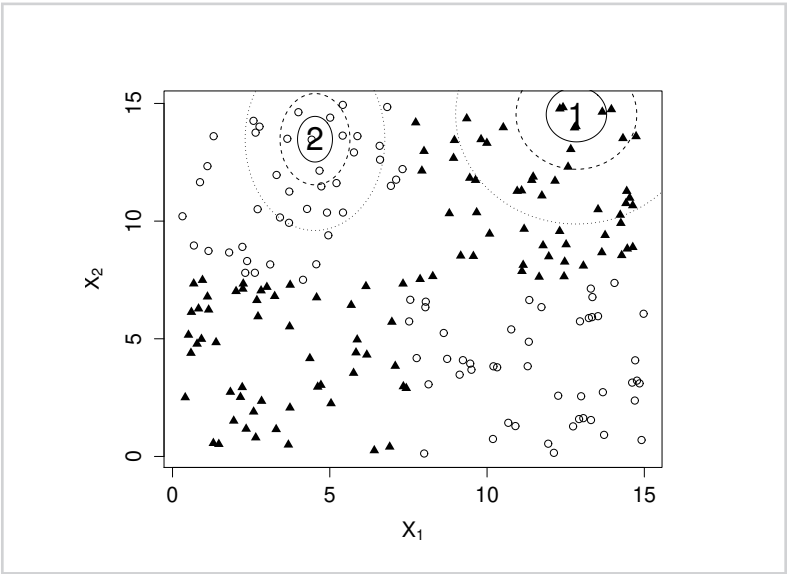
Pri gradnji pravila moramo upoštevati le pokritost iskanega razreda, ostale primere pa obravnavamo kot nepokrite. Sliki 4.2 in 4.3 prikazujeta, zakaj je to potrebno. Na sliki 4.2 sta vidni pravilno zgrajeni prvi pravili, na sliki 4.3 pa sta vidni napačno zgrajeni prvi pravili. V prvem primeru so uporabljene uteži za vsak razred posebej, v drugem pa samo ena utež pokritosti za vsak primer. V drugem primeru se po gradnji prvega pravila najbolj zmanjšajo uteži primerom v zgornjem desnem kotu. Pri gradnji drugega pravila so ti primeri za pravilo 2 že pokriti in zato to pravilo poskušamo posplošiti tudi čez te primere v smeri atributa X_1. To preprečimo tako, da pravilo 2 upošteva le uteži za isti razred. Na ta način so primeri iz ostalih razredov upoštevani kot nepokriti, kar omejuje pretirane posplošitve, izognemo pa se tudi večkratnemu pokrivanju istega prostora.

4.4.5 Posodobitev feromonov

Feromoni se posodabljajo v dveh korakih, in sicer v koraku ojačanja in izhlapevanja. Ojačanje poveča vrednost feromonov na dobrih poteh, izhlapevanje pa zmanjšuje vrednost feromonov skozi iteracije. Ojačanje se izvede za najboljše pravilo R_{best}, izhlapevanje pa se simulira z normalizacijo vsote vrednosti na 1, kar pomeni, da vrednosti feromonov delimo z vsoto vseh feromonov. Tako se vrednost feromona na ojačani poti zviša, na vseh ostalih poteh pa se vrednost feromona zaradi normalizacije zmanjša.

Pri nominalnih atributih je ojačana le vrednost, ki nastopa v pravilu. Dodana vrednost je $\beta \cdot \mathrm{Q}_{best}$. Izhlapevanje simuliramo z normalizacijo vseh intervalov na vsoto 1.

Pri številskih atributih so ojačani vsi intervali. Vzemimo na primer numerični člen T_{ij} oblike $\mathrm{A} = \mathrm{N}(\mu_{ij}, \sigma_{ij})$. Zgradimo Gaussovo krivuljo $\mathrm{N}(\mu_{ij}, \sigma_{ij})$ in jo



Slika 4.2
Prikaz pravilno zgrajenih prvih dveh pravil na množici *xor*.

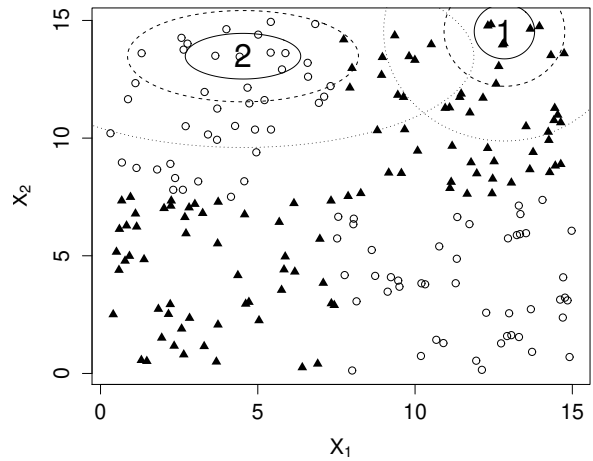
normaliziramo tako, da je skupna vsota 1, kot prikazuje naslednja enačba.

$$P_{i,j}^{\text{update}} = \left\{ \frac{\frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(x_{fm}-\mu_{ij})^2}{2\sigma_{ij}^2}}}{\sum_{o=1}^{b} \left(\frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(x_{fo}-\mu_{ij})^2}{2\sigma_{ij}^2}} \right)} ; m \in \{1, \ldots, b\} \right\}, \tag{4.18}$$

kjer je x_{fm} srednja vrednost intervala m atributa A_f. Dobljena distribucija, pomnožena z β in Q_{best} se prišteje k trenutni distribuciji feromonov.

4.4.6 Uteži primerov

Ko je pravilo zgrajeno, pokritim primerom zmanjšamo uteži glede na kakovost pravila Q_i in vrednostjo, s katero je primer pokrit. Pokritost je izračunana po enačbi (4.6). Vsaka utež w_k je zmanjšanja za vrednost $Q_i \cdot c_{ik}$. Če vrednost uteži pade pod prag (na primer 0,05), jo nastavimo na 0, zato ne vpliva več na nadaljnje preiskovanje. Na ta način hranimo tudi uteži, odvisne od razreda $w_k^{l_c}$. Izračun teh uteži upošteva le primere, kjer je razred l_c enak razredu pravila R_{best}, kar



Slika 4.3
Prikaz nepravilno zgrajenih prvih dveh pravil na množici *xor*.

preprečuje pretirane posplošitve pravil.

4.4.7 Ustavitveni pogoj

Algoritem gradi pravila, dokler ni pokritih dovolj primerov ali doseženo maksimalno število pravil. Vsak primer začne z utežjo $w_k = 1$. Vsota uteži se z novimi pravili manjša. Ko vsota pade pod minimalno (*minCover*), ali ko dosežemo maksimalno število pravil (*maxRules*), se algoritem ustavi.

4.5 Empirično vrednotenje metode

V tem razdelku metodo ProAnt-Miner primerjamo na enajstih realnih množicah, dobljenih iz UCI repozitorija [42] in na štirih umetnih množicah z močnimi pogojnimi odvisnostmi med atributi, opisanih v razdelku 3.2.3. Na koncu smo ponovno naredili analizo na podatkih o sinkopi.

4.5.1 Realne podatkovne množice

V tem razdelku primerjamo metodo ProAnt-Miner z naključnimi gozdovi (RF) [47], metodo RIPPER [5], metodo Ant-Miner+ [2], metodo FURIA [4], metodo CN2 [10] in našo metodo nAnt-Miner [7] na enajstih realnih podatkovnih množicah. Množice, na katerih smo primerjali metode, so Balance Scale (*bal*), Glass Identification (*glass*), Teaching Assistant Evaluation (*tae*), Wine (*wine*), Haberman's Survival (*hab*), Ionosphere (*iono*), Statlog(Heart) (*heart*), Liver Disorder (*bupa*), Pima Indians Diabetes (*pima*), Sonar (*sonar*) in Ecoli (*ecoli*). Množice so bile izbrane glede na število vsebovanih številskih atributov. Tabela 4.1 prikazuje osnovne podatke izbranih množic. Metodo ProAnt-Miner primerjamo tudi z metodo naključnih gozdov, ker robustno dosega visoko klasifikacijsko točnost. Rezultati te metode predstavljajo oceno zgornje meje klasifikacijske točnosti, ki jo lahko dosežemo. Metodo primerjamo tudi z metodami Ant-Miner+, RIPPER, FURIA in CN2, ki so opisane v razdelku 3.2.2.

Teste smo naredili s 5 × 2 prečnim preverjanjem. Uporabljene metode naključnih gozdov, RIPPER in FURIA so dostopne v programu Weka [44]. Implementacija uporabljene metode CN2 pa je dostopna v programu Orange [45]. Metoda Ant-Miner+ je prosto dostopna na spletu[1]. Pri metodi naključnih gozdov smo uporabili 100 dreves, pri vseh ostalih metodah smo uporabili privzete parametre. Pri metodi Ant-Miner+ smo uporabili 1000 mravelj, faktor izhlapevanja 0,85, največje število iteracij pa smo omejili na 200. Za oceno kakovosti pravil smo uporabili razmerje med pokritostjo in zaupanjem pravila, za diskretizacijo zveznih atributov pa metodo Fayyad-Irani. Pri metodi FURIA smo za določanje širine mehkih intervalov uporabili privzeto metodo raztezanja. Privzeti parametri za metodo ProAnt-Miner so 10 mravelj, minCover = $n/2$, kjer je n število primerov v učni množici, $\alpha = 0,05$ (glej enačbo 4.11) in $\lambda = 3$ (faktor, s katerim množimo Gaussove krivulje, zmnožek je omejen navzgor z 1). Privzeti parametri so bili nastavljeni s poskušanjem metode na množici *Iris*.

[1] www.antminerplus.com

Tabela 4.1

Opis množic. Vrednosti stolpcev so: *#at.* - število atributov, *#st.at.* - število številskih atributov, *#prim* - število primerov in *#raz* - število razredov

Množica	# at.	# št. at.	# prim.	# raz
bal	4	4	625	3
glass	9	9	214	6
tae	5	3	151	3
wine	13	13	178	3
hab	3	3	306	2
iono	34	34	351	2
heart	13	7	270	2
bupa	6	6	345	2
pima	8	8	768	2
sonar	60	60	208	2
ecoli	7	7	336	8

Tabela 4.2 prikazuje dobljene klasifikacijske točnosti na 11-ih množicah. Rezultate metod smo po [43] testirali najprej s Friedmanovim testom. Ker smo metodo naključnih gozdov uporabili predvsem za določanje zgornje meje klasifikacijske točnosti in ne zato, da bi pokazali, da lahko s to metodo primerjamo razvito metodo ProAnt-Miner, smo to metodo odstranili iz statističnega testa. Dobljena p-vrednost statističnega testa $5,57 \cdot 10^{-5}$ na primerjanih šestih metodah zavrne ničelno hipotezo, da so vse metode enakovredne. Rezultati Nemenyjevega testa so prikazani v tabeli 4.3 in na sliki 4.4. Iz te tabele in slike vidimo, da je metoda FURIA statistično boljša od metod Ant-Miner+, CN2 in nAnt-Miner ($p < 0,05$), ravno tako je metoda RIPPER statistično boljša kot metoda nAnt-Miner. Kritična razdalja je v tem primeru 2,273. Iz slike 4.4 vidimo, da je vodilna skupina metod, za katere ne moremo trditi, da se med seboj razlikujejo, sestavljena iz metod FURIA, RIPPER in ProAnt-Miner.

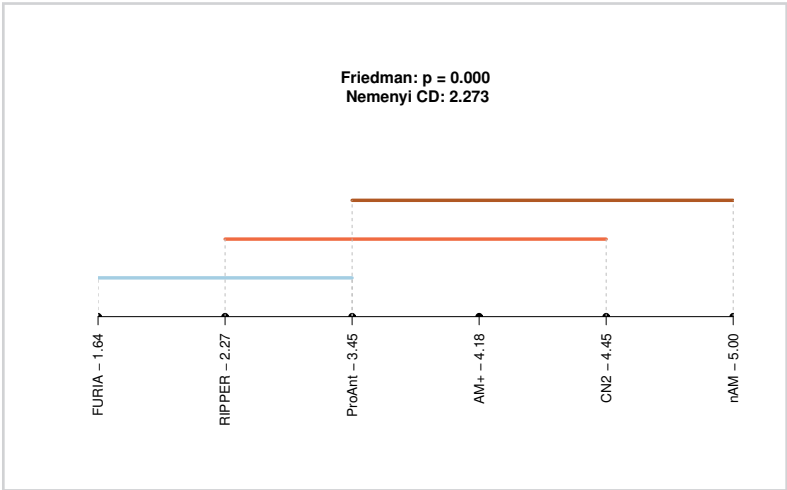
Tabela 4.2

Primerjava in rangiranje algoritmov RF, ProAnt-Miner, Ant-Miner+, RIPPER in FURIA glede na dobljenje klasifikacijske točnosti. Odebeljeni so najboljši rezultati brez upoštevanje metode RF. Rang v zadnji vrstici predstavlja povprečen rang metode in standardni odklon ranga.

Metoda	ProAnt	nAM	AM+	RIPPER	FURIA	CN_2	RF
bal	77,28 ± 3,08 (5)	46,46 ± 2,48 (7)	74,62 ± 3,24 (6)	80,67 ± 0,74 (3)	83,46 ± 0,56 (2)	77,68 ± 2,70 (4)	84,83 ± 0,46 (1)
glass	59,81 ± 4,93 (4)	55,70 ± 9,68 (7)	58,79 ± 5,07 (6)	62,15 ± 1,78 (3)	62,24 ± 3,66 (2)	59,38 ± 5,78 (5)	74,86 ± 0,90 (1)
tae	43,99 ± 8,42 (3)	38,03 ± 7,07 (5)	36,56 ± 5,12 (7)	38,28 ± 2,66 (4)	38,01 ± 1,60 (6)	54,26 ± 7,54 (1)	53,37 ± 1,37 (2)
wine	88,20 ± 9,83 (4)	87,19 ± 2,71 (5)	85,17 ± 2,20 (6)	88,54 ± 1,84 (3)	95,10 ± 0,00 (2)	83,27 ± 4,01 (7)	96,97 ± 0,85 (1)
hab	72,16 ± 2,41 (3)	69,54 ± 7,09 (5)	47,12 ± 20,7 (7)	73,14 ± 1,02 (2)	74,68 ± 0,94 (1)	63,78 ± 1,49 (6)	70,79 ± 0,18 (4)
iono	81,93 ± 7,20 (7)	83,71 ± 4,14 (5)	90,09 ± 1,01 (2)	88,09 ± 1,38 (3)	87,12 ± 0,85 (4)	82,23 ± 3,83 (6)	93,11 ± 0,33 (1)
heart	77,19 ± 2,92 (2)	50,81 ± 5,56 (7)	76,15 ± 3,12 (4)	74,74 ± 4,77 (5)	76,96 ± 1,13 (3)	70,37 ± 4,10 (6)	78,37 ± 0,89 (1)
bupa	60,46 ± 2,32 (4)	55,77 ± 5,57 (6)	43,65 ± 2,06 (7)	60,58 ± 2,19 (3)	62,61 ± 0,66 (2)	57,50 ± 3,81 (5)	65,86 ± 0,76 (1)
pima	69,95 ± 2,39 (5)	67,84 ± 2,03 (6)	71,80 ± 1,68 (4)	72,81 ± 1,08 (3)	73,80 ± 1,25 (2)	66,04 ± 2,39 (7)	75,34 ± 0,62 (1)
sonar	63,85 ± 4,97 (5)	60,96 ± 6,70 (6)	69,23 ± 2,26 (4)	71,25 ± 3,74 (3)	72,12 ± 0,80 (2)	60,53 ± 7,60 (7)	78,46 ± 1,19 (1)
ecoli	64,05 ± 5,19 (5)	60,36 ± 9,96 (7)	77,44 ± 1,90 (4)	79,05 ± 1,60 (3)	82,80 ± 0,79 (2)	72,18 ± 5,35 (5)	83,39 ± 0,39 (1)
rang	4,33 ± 1,37	5,67 ± 1,44	4,83 ± 1,99	3,5 ± 1,31	2,83 ± 1,64	5,17 ± 1,80	1,67 ± 1,37

Tabela 4.3
P-vrednosti parnega testa Nemenyi.

	ProAnt	nAM	AM+	RIPPER	FURIA
nAM	0,37919	-	-	-	-
AM+	0,94368	0,90950	-	-	-
RIPPER	0,67620	0,00826	0,15849	-	-
FURIA	0,20250	0,00036	0,01779	0,96799	-
CN2	0,81015	0,98379	0,99939	0,06851	0,00551



Slika 4.4
Grafični prikaz testa Nemenyi.

Tabela 4.4 prikazuje klasifikacijske točnosti metode ProAnt-Miner pri različnem številu uporabljenih mravelj od 3 do 50. Če se osredotočimo le na povprečen rang pri različnem številu mravelj, bi to lahko kazalo, da je najboljša vrednost 50, vendar vidimo, da so najboljši rezultati dokaj enakomerno razporejeni čez vse vrednosti parametra. To nakazuje, da je metoda občutljiva na število mravelj in je treba ta parameter določiti za vsako množico posebej.

Tabela 4.4
Klasifikacijske točnosti glede na uporabljeno število mravelj pri metodi ProAnt-Miner.

Ants	3	5	10	20	50
bal	77,57 ± 2,63 (5)	78,46 ± 2,86 (4)	79,07 ± 5,42 (3)	79,74 ± 1,93 (1)	79,49 ± 2,32 (2)
glass	51,50 ± 5,51 (5)	54,49 ± 3,66 (4)	55,42 ± 8,81 (3)	57,76 ± 5,84 (2)	58,97 ± 7,34 (1)
tae	42,79 ± 7,88 (2)	42,27 ± 6,21 (5)	45,43 ± 6,80 (1)	42,51 ± 8,25 (4)	42,54 ± 8,26 (3)
wine	92,02 ± 3,53 (3)	92,70 ± 3,97 (1)	88,43 ± 8,86 (5)	90,45 ± 4,53 (4)	92,25 ± 4,12 (2)
hab	72,42 ± 1,87 (4)	73,07 ± 3,23 (1)	72,75 ± 1,63 (3)	71,83 ± 2,47 (5)	72,87 ± 1,31 (2)
iono	82,34 ± 5,59 (4)	81,94 ± 4,33 (5)	85,65 ± 4,17 (1)	83,30 ± 4,31 (3)	83,76 ± 5,59 (2)
heart	76,52 ± 4,94 (1)	72,60 ± 5,55 (4)	74,89 ± 4,17 (2)	74,44 ± 4,24 (3)	71,41 ± 7,26 (5)
bupa	56,41 ± 3,70 (5)	58,61 ± 4,30 (1)	57,04 ± 2,25 (4)	57,28 ± 3,73 (3)	58,32 ± 3,06 (2)
pima	69,66 ± 3,83 (4)	69,79 ± 2,54 (3)	70,55 ± 2,35 (2)	69,45 ± 5,36 (5)	70,99 ± 3,37 (1)
sonar	65,29 ± 7,40 (1)	60,29 ± 6,10 (5)	63,37 ± 4,85 (2)	63,27 ± 6,13 (3)	63,08 ± 6,85 (4)
ecoli	63,63 ± 6,07 (3)	61,31 ± 7,07 (5)	62,86 ± 9,01 (4)	65,48 ± 7,32 (2)	65,95 ± 4,58 (1)
rang	3,36 ± 1,50	3,45 ± 1,69	2,73 ± 1,27	3,18 ± 1,25	2,27 ± 1,27

Tabela 4.5 predstavlja rezultate za različne vrednosti parametra λ. Tudi tukaj rangi glede na vrednosti parametra nakazujejo, da privzetega parametra ne moremo preprosto določiti, ampak ga je treba določiti za vsako podatkovno množico posebej. Tako velikih odstopanj, kot pri parametru Ants tukaj ni, razlog je, da parametra σ in λ podobno razširita mehki interval, kar pomeni, da imajo pravila z nizko vrednostjo λ večkrat višjo vrednost σ. Pri tem metoda ProAnt-Miner implicitno določa vrednost σ.

Tabela 4.5
Klasifikacijske točnosti glede na različne vrednosti parametra λ pri metodi ProAnt-Miner.

λ	1	3	5	10
bal	78,53 ± 3,97 (3)	78,58 ± 4,34 (2)	78,05 ± 4,81 (4)	79,94 ± 3,41 (1)
glass	59,63 ± 7,67 (2)	55,05 ± 6,82 (4)	56,17 ± 7,57 (3)	60,0 9± 4,09 (1)
tae	41,87 ± 5,88 (3)	40,39 ± 8,01 (4)	42,67 ± 9,91 (1)	42,51 ± 6,31 (2)
wine	91,67 ± 4,18 (2)	93,03 ± 3,47 (1)	91,24 ± 2,53 (3)	90,34 ± 3,19 (4)
hab	72,35 ± 1,27 (3)	71,57 ± 3,83 (4)	72,48 ± 2,23 (2)	73,27 ± 1,55 (1)
iono	83,82 ± 5,36 (3)	83,48 ± 5,39 (4)	86,15 ± 2,88 (1)	86,15 ± 3,93 (1)
heart	75,26 ± 3,92 (3)	75,78 ± 1,60 (2)	76,74 ± 2,52 (1)	75,11 ± 4,05 (4)
bupa	58,15 ± 2,93 (2)	58,38 ± 5,01 (1)	57,97 ± 2,95 (3)	57,86 ± 3,30 (4)
pima	71,35 ± 2,65 (1)	71,04 ± 1,94 (2)	69,58 ± 4,04 (4)	69,87 ± 2,41 (3)
sonar	66,15 ± 5,38 (2)	59,71 ± 8,93 (4)	64,04 ± 6,59 (3)	71,06 ± 5,60 (1)
ecoli	63,99 ± 4,87 (1)	59,40 ± 10,6 (4)	62,56 ± 8,71 (2)	61,85 ± 10,4 (3)
mean	2,27 ± 0,79	2,91 ± 1,30	2,45 ± 1,13	2,27 ± 1,35

4.5.2 Umetne množice

Metodo smo preizkusili na umetnih množicah z znanimi močnimi pogojnimi odvisnostmi med atributi. Množice so pridobljene iz članka [46] in so iste kot v razdelku 3.2.3.

Tabela 4.6 prikazuje dobljene klasifikacijske točnosti na umetnih množicah. Metoda Ant-Miner+ je tukaj odpovedala, ker je prišlo do izgube informacije pri uporabi metode Fayyad-Irani za diskretizacijo. Na množicah xor, chess in xorCross metoda ProAnt-Miner dosega nižje klasifikacijske točnosti od metod nAnt-Miner, RIPPER in FURIA. Razlog za to je v tem, da metoda ProAnt-Miner gradi verjetnostna oz. Gaussovska pravila, ki so elipsne oblike, kot je prikazano na sliki

4.3 in zaradi tega zelo težko dobi stroge meje, kot jih imajo vse te tri množice. Na množici modGroups metoda ProAnt-Miner deluje najboljše, ker nima težav s strogimi ravnimi mejami, čeprav ima ta množica več razredov in močno pogojno odvisnost med atributi. Ker ima FURIA trapezoidno obliko mehkih pravil se ta pravila lažje prilegajo umetnim množicam.

Tabela 4.6
Klasifikacijske točnosti na umetnih množicah z močno pogojno odvisnostjo med atributi.

Metoda	ProAnt	nAM	AM+
xor	82,40 ± 8,88	96,80 ± 1,93	51,40 ± 6,23
chess	51,00 ± 5,83	58,20 ± 7,15	47,70 ± 5,37
modGroup	65,40 ± 14,8	65,20 ± 9,85	31,40 ± 1,95
xorCross	58,00 ± 7,42	89,00 ± 8,12	47,70 ± 5,37
Metoda	RIPPER	FURIA	
xor	88,90 ± 3,11	98,00 ± 0,71	
chess	60,60 ± 2,07	55,8 ± 3,77	
modGroup	42,40 ± 10,90	37,40 ± 9,84	
xorCross	84,40 ± 10,92	89,80 ± 0,45	

4.5.3 Medicinska domena

Metodo smo preizkušali še na realni podatkovni množici *syncope*, kot v poglavju 3.2.4. Test smo opravili na dveh različicah podatkov. Prvega smo poimenovali *syncope123* in vsebuje le meritve prvih 15 minut testa. Drugi test *syncope* vsebuje vse meritve. Vsi testi so bili pognani s privzetimi nastavitvami metod.

Tabela 4.7 prikazuje rezultate klasifikacijske točnosti. Vidimo, da metoda Ant-Miner+ dobi dobre rezultate pri obeh množicah. Zanimivo je, da so vse tri metode, zasnovane na osnovi kolonije mravelj, ProAnt-Miner, nAnt-Miner in Ant-Miner+ boljše od preostalih metod na množici *syncope123*, na množici *syncope* z več atributi pa dosegajo slabšo klasifikacijsko točnost. Izjema je Ant-Miner+, ki tudi tukaj dosega visoko klasifikacijsko točnost. Število pravil pri metodi ProAnt-Miner je visoko, vendar kot je opisano v razdelku 4.5.4, ni vedno treba, da uporabimo vsa pravila, saj velikokrat že nekaj prvih pravil zadošča za klasifikacijo in je to število potrebno naknadno določiti za boljše rezultate. Razvidno je tudi,

da z upoštevanjem optimalnega števila pravil metoda ProAnt-Miner dosega boljše rezultate od predstavljenih v tem poglavju. Čas izvajanja metode ProAnt-Miner je tudi manjši v primerjavi z nAnt-Miner, predvsem, ker ima manjši preiskovalni graf, saj so pri tej metodi zvezni atributi predstavljeni z le enim vozliščem.

Tabela 4.7

Klasifikacijske točnosti na medicinski domeni.

Metoda	ProAnt	nAM	AM+
syncope123	47,98 ± 7,17	47,13 ± 6,27	53,78 ± 17,0
syncope	53,93 ± 8,51	50,74 ± 7,65	64,25 ± 10,6
Metoda	RIPPER	FURIA	RF
syncope123	40,64 ± 5,81	43,01 ± 5,99	44,96 ± 2,45
syncope	62,58 ± 2,33	61,08 ± 2,45	65,16 ± 2,10

Tabela 4.8

Velikost pravil na medicinski domeni. Podana je vrednost S, ki je zmnožek števila pravil in povprečnega števila členov na pravilo.

Metoda	ProAnt	nAM	AM+	RIPPER	FURIA
syncope123	16,8	8,55	1,00	1,6	9,2
syncope	12,5	8,46	6,21	3,2	16

Tabela 4.9

Čas izvajanja metod za 5 × 2 prečno preverjanje v sekundah.

Metoda	ProAnt	nAM	AM+
syncope123	31	53	<1
syncope	33	315	22
Metoda	RIPPER	FURIA	RF
syncope123	<1	<1	<1
syncope	<1	<1	<1

4.5.4 Vpliv velikosti seznama pravil

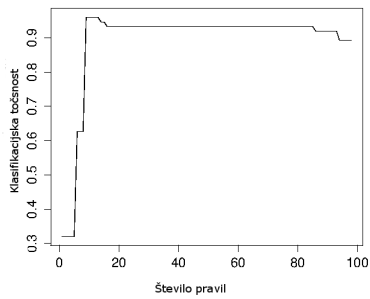
ProAnt-Miner rudari verjetnostna pravila, dokler ne pokrije dovolj učnih primerov. Če algoritem ustavimo prezgodaj, ne dobimo dovolj pravil za visoko klasifikacijsko točnost. Če algoritem pustimo, da zgradi velik seznam pravil, v večini primerov povečamo točnost, vendar je velik seznam pravil nerazumljiv. V tem razdelku smo opazovali klasifikacijsko točnost na množicah *iris*, *balance* in *syncopa* glede na velikost dobljenega seznama pravil. Parametre metode smo nastavili na $\alpha = 0,05$, število mravelj = 5, $\lambda = 1$ in minCoverage = 1. Parameter minCoverage = 1 zagotavlja, da bomo dobili velik seznam pravil. Za to testiranje smo množice razdelili na dve polovici, prvo polovico smo uporabili za učenje, drugo pa za testiranje. Slike 4.5-4.7 prikazujejo rezultate za en zagon algoritma.

Vidimo, da metoda že z nizkim številom pravil doseže visoko klasifikacijsko točnost. Le-ta z dodajanjem pravil ostane več ali manj enaka oziroma v zadnjem primeru celo pade. To pomeni, da lahko naknadno dobljeni seznam odrežemo in s tem povečamo klasifikacijsko točnost. To mejo bi lahko avtomatsko določili z internim prečnim preverjanjem ali z uporabo validacijske množice.

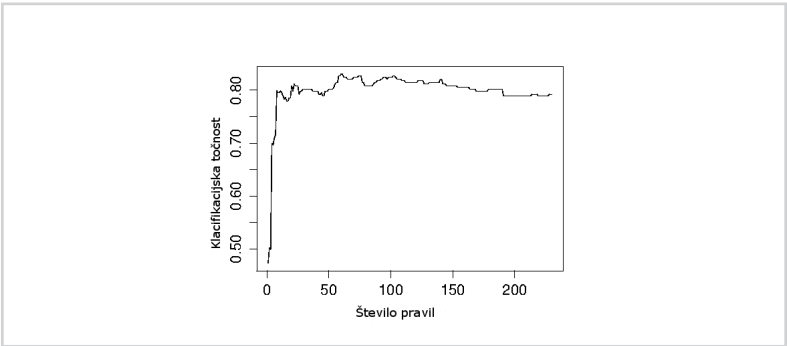
Na naslednji strani je podan primer začetka dobljenega seznama. V tem primeru imajo vsa pravila v seznamu samo en člen. Prvi člen prvega pravila petallength = $N(1,40, 0,51)$, nam pove, da primeri z vrednostjo petallength blizu 1,4 pripadajo razredu *setosa*. Standardni odklon 0,51, nam pove, kako velikim odmikom od 1,4 še lahko zaupamo. Podobno za razred *versicolor* velja, da morajo biti vrednosti petalwidth blizu 1,18, vendar ker je v tem primeru standardni odklon 0,18, morajo biti vrednosti bližje. Vidimo, da so vsa pravila sestavljena tako, da vrednosti petalwidth nad 1,7 napovedujejo razred *virginica*, vrednosti pod 1,7 pa razred *versicolor*. Bolj kot se vrednost atributa petalwidth v členih pravil bliža vrednosti 1,7, manjši je standardni pripadajoči odklon, kar nakazuje, da so napovedi blizu

vrednosti 1,7 manj zaupljive.

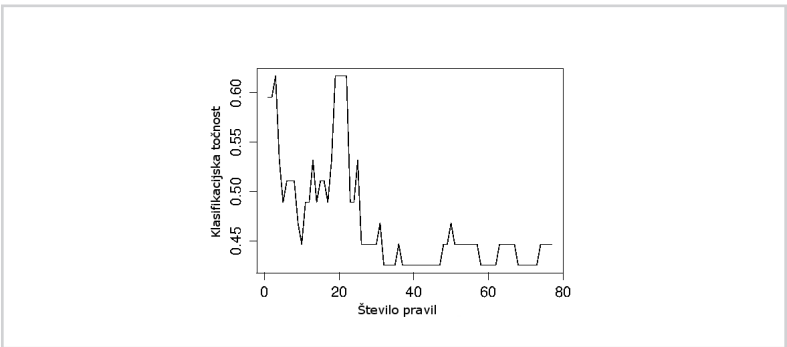
IF petallength=N(1, 40, 0, 51) THEN setosa
IF petalwidth=N(1, 18, 0, 18) THEN versicolor
IF petalwidth=N(2, 10, 0, 15) THEN virginica
IF petalwidth=N(1, 80, 0, 06) THEN virginica
IF petalwidth=N(2, 40, 0, 21) THEN virginica
IF petalwidth=N(1, 50, 0, 05) THEN versicolor
IF petalwidth=N(1, 60, 0, 02) THEN versicolor
...



Slika 4.5
Klasifikacijska točnost glede na velikost seznama pravil na množici *iris*.



Slika 4.6
Klasifikacijska točnost glede na velikost seznama pravil na množici *balance*.



Slika 4.7
Klasifikacijska točnost glede na velikost seznama pravil na množici *syncope*.

4.5.5 Zaključek

Rezultati v razdelku 4.5.1 so pokazali, da je metoda ProAnt-Miner primerljiva z ostalimi metodami za iskanje pravil. Razvidno je, da je parametra število mravelj in λ (faktor za množenje Gaussovskih krivulj) treba prilagajati vsaki množici posebej, če želimo dobiti dobre rezultate. Paziti je treba tudi na velikost seznama pravil in ga prilagajati podatkovni množici. Z uporabo pazljivo izbranih parametrov je rezultate torej mogoče izboljšati. Nadaljnje delo je zato v avtomatskem iskanju najboljših parametrov, na primer z uporabo množice za ovrednotenje (evalvation set). Razdelek 4.5.2 pokaže na slabost metode ProAnt-Miner, da ne išče ostrih oziroma pravokotnih mej v podatkih, kar bi bila možna razširitev metode. Prednost metode je v tem, da za klasifikacijo zadostuje že nekaj prvih dobljenih pravil in

lahko seznam pravil na koncu porežemo, da je bolj razumljiv.

5

Zaključek

V tem poglavju najprej povzamemo narejeno delo in nato podamo ideje za nadaljnje delo.

V disertaciji opišemo razvoj dveh metod za učenje klasifikacijskih pravil na podlagi optimizacije s kolonijo mravelj, in sicer nAnt-Miner in ProAnt. Metoda nAnt-Miner je primerna za učenje v domenah s številskimi atributi, kadar imamo na voljo dovolj učnih primerov in sumimo, da so v podatkih močne pogojne odvisnosti med atributi, kar smo pokazali v razdelku 3.2.3. Za zaznavo nekaterih pogojnih odvisnosti, kot je v množici *bal*, bi metoda potrebovala več učnih primerov, kot je razvidno iz članka [7], kjer je za učenje uporabljenih 90 odstotkov učnih primerov in so zato rezultati na tej množici primerljivi z metodo Ant-Miner+. V vseh testih je metoda nAnt-Miner uporabljala 10 odstotkov učne množice za validacijsko množico. To bi lahko na majhnih množicah za boljše rezultate odstranili. Ker metoda nAnt-Miner uporablja kolonijo mravelj za iskanje pravil, je počasnejša od metod, ki tega pristopa ne uporabljajo. Vrednotenje metod v razdelku 3.2.1 kaže, da lahko predvsem s parametrom za nastavitev meje konvergence dosežemo boljše rezultate na račun daljšega časa izvajanja.

Metoda ProAnt se po klasifikacijski točnosti statistično ne razlikuje od metod FURIA in RIPPER, ki sta vodilni metodi za iskanje pravil. Metoda ProAnt-Miner išče verjetnostna pravila, ki jih, pomnožene s parameterom λ, lahko interpretiramo podobno kot mehka pravila pri metodi FURIA. Vrednotenje metode na umetnih množicah v razdelku 4.5.2 je pokazalo, da ima metoda težave z določanjem ostrih in ravnih mej. Ovrednotenje metode v razdelkih 4.5.1 in 4.5.4 pa kaže, da je metodo treba prilagajati vsaki množici posebej, predvsem število mravelj, parameter λ in število uporabljenih pravil za klasifikacijo. Prednost metode ProAnt-Miner pred metodo nAnt-Miner je tudi v času izvajanja in porabljenem pomnilniku za delovanje, ker si metoda ProAnt-Miner številske atribute predstavi z distribucijo feromonov in ima zato veliko manj povezav v grafu preiskovanja. Ta težava je pri metodi nAnt-Miner privedla do uporabe dinamičnega grafa preiskovanja.

Iz dobljenih rezultatov sledi, da je kot nadaljnje delo potrebno avtomatsko določanje parametrov metode ProAnt-Miner in razširitev metode, da bo iskala tako verjetnostna pravila kot ostra pravila na številskih atributih. Prvi del lahko realiziramo podobno, kot so opisali avtorji Pellegrini in sodelavci [41].

Pri iskanju pravil s kolonijo mravelj ima na dobljena pravila velik vpliv tudi mera za ocenjevanje pravil. Smiselno je preizkusiti uporabo drugih mer za izra-

čun kakovosti pravil oziroma konstruirati primerno oceno kakovosti pravil za to metodo, podobno kot Medland in Otero [29].

Glede na to, da so verjetnostna pravila s porezano normalno obliko podobna trapezoidnim mehkim intervalom, je smiselno pretvoriti verjetnostna pravila v mehka pravila in sistem preizkusiti za rudarjenje mehkih pravil.

Metoda ProAnt-Miner pri klasificiranju upošteva vsa pravila hkrati. Glede na to bi lahko algoritem zagnali večkrat s spremenjenimi parametri in dobili ansambel pravil. Pri nadaljnjem delu bomo skušali metodo verjetnostnih pravil razširiti v ansambel in tako izboljšati klasifikacijsko točnost na račun razumljivosti.

Metodi sta bili razviti z namenom praktične uporabe na farmacevtskih podatkih oziroma tam, kjer med atributi verjetno obstajajo odvisnosti, obenem pa želimo razumljiv opis. Takšen primer so stranski učinki jemanja več vrst zdravil.

LITERATURA

[1] Otero FE, Freitas AA, Johnson CG (2008) cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes in *International Conference on Ant Colony Optimization and Swarm Intelligence*. doi: 10.1007/978-3-540-87527-7_5.

[2] Martens D et al. (2007) Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation* doi: 10.1109/TEVC.2006.890229.

[3] Fayyad U, Irani K (1993) Multi-interval discretization of continuous-valued attributes for classification learning in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. pp. 1022–1029.

[4] Hühn J, Hüllermeier E (2009) FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery* doi: 10.1007/s10618-009-0131-8.

[5] Cohen WW (1995) Fast effective rule induction in *Machine Learning Proceedings*. pp. 115–123.

[6] Dorigo M, Caro GD, Gambardella LM (1999) Ant algorithms for discrete optimization. *Artificial life* doi: 10.1162/106454699568728.

[7] Piculin M, Robnik-Šikonja M (2014) Handling numeric attributes with ant colony based classifier for medical decision making. *Expert systems with applications* doi: 10.1016/j.eswa.2014.06.017.

[8] Kononenko I, Kukar M (2007) *Machine learning and data mining: Introduction to principles and algorithms*. (Horwood Publishing).

[9] Fürnkranz J, Gamberger D, Lavrač N (2012) *Foundations of rule learning*. (Springer).

[10] Clark P, Niblett T (1989) The CN2 induction algorithm. *Machine learning* doi: 10.1023/A:1022641700528.

[11] Hühn JC, Hüllermeier E (2010) An analysis of the FURIA algorithm for fuzzy rule induction in *Proceedings of Advances in machine learning I*. doi: 10.1007/978-3-642-05177-7_16.

[12] Michalski RS (1969) On the quasi-minimal solution of the general covering problem in *Proceedings of the Fifth International Symposium on Information Processing*. pp. 125–128.

[13] Bagallo G, Haussler D (1990) Boolean feature discovery in empirical learning. *Machine learning* doi: 10.1007/bf00115895.

[14] Cendrowska J (1987) PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* 27(4):349–370.

[15] Quinlan JR (1983) Learning efficient classification procedures and their application to chess end games in *Machine learning: An artificial intelligence approach*. pp. 463–482.

[16] Quinlan JR (1990) Learning logical definitions from relations. *Machine learning* doi: 10.1007/BF00117105.

[17] Quinlan JR (1993) *C4.5: programs for machine learning*. (Morgan Kaufmann Publishers Inc.).

[18] Stützle T, Hoos HH (2000) MAX-MIN ant system. *Future generation computer systems* doi: 10.1016/S0167-739X(00)00043-1.

[19] Dorigo M, Di Caro G (1999) Ant colony optimization: A new meta-heuristic in *Proceedings of the 1999 Congress on Evolutionary Computation*. Vol. 2, doi: 10.1109/CEC.1999.782657.

[20] Parpinelli RS, Lopes HS, Freitas AA (2002) Data mining with an ant colony optimization algorithm. *IEEE transactions on evolutionary computation* doi: 10.1109/TEVC.2002.802452.

[21] Wang Z, Feng B (2004) Classification rule mining with an improved ant colony algorithm in *AI 2004: Advances in Artificial Intelligence*. doi: 10.1007/978-3-540-30549-1_32.

[22] Liu B, Abbas HA, McKay B (2003) Classification rule discovery with ant colony optimization in *IEEE/WIC International Conference on Intelligent Agent Technology*. doi: 10.1109/iat.2003.1241052.

[23] Fürnkranz J (1999) Separate-and-conquer rule learning. *Artificial Intelligence Review* doi: 10.1023/A:1006524209794.

[24] Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* doi: 10.1109/3477.484436.

[25] Yildirim UM, Çatay B (2012) A time-based pheromone approach for the ant system. *Optimization Letters* doi: 10.1007/s11590-012-0451-2.

[26] Cordón O, de Viana IF, Herrera F (2002) Analysis of the best-worst ant system and its variants on the QAP in *International Workshop on Ant Algorithms*. doi: 10.1007/3-540-45724-0_20.

[27] Salama KM, Abdelbar AM (2010) Extensions to the Ant-Miner classification rule discovery algorithm in *International Conference on Swarm Intelligence*. doi: 10.1007/978-3-642-15461-4_15.

[28] Chan A, Freitas A (2005) A new classification-rule pruning procedure for an ant colony algorithm in *International Conference on Artificial Evolution (Evolution Artificielle)*. doi: 10.1007/11740698_3.

[29] Medland M, Otero F (2012) A study of different quality evaluation functions in the cAnt-Miner(PB) classification algorithm in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. doi: 10.1145/2330163.2330171.

[30] Liu B, Abbass HA, McKay B (2002) Density-based heuristic for rule discovery with Ant-Miner in *The 6th Australia-Japan joint workshop on intelligent and evolutionary system*. Vol. 184, pp. 180–184.

[31] Baig AR, Shahzad W (2012) A correlation-based Ant Miner for classification rule discovery. *Neural Computing and Applications* doi: 10.1007/s00521-010-0490-5.

[32] Otero FE, Freitas AA, Johnson CG (2009) Handling continuous attributes in ant colony classification algorithms in *IEEE Symposium on Computational Intelligence and Data Mining*. doi: 10.1109/cidm.2009.4938653.

[33] Otero FE, Freitas AA, Johnson CG (2013) A new sequential covering strategy for inducing classification rules with ant colony algorithms. *IEEE Transactions on Evolutionary Computation* doi: 10.1109/tevc.2012.2185846.

[34] López-Ibáñez M, Stützle T (2012) An experimental analysis of design choices of multi-objective ant colony optimization algorithms. *Swarm Intelligence* doi: 10.1007/s11721-012-0070-7.

[35] Said N, Hammami M, Ghedira K (2011) MulO-AntMiner: A new ant colony algorithm for the multi-objective classification problem in *International Conference on Computational Science and Its Applications*. doi: 10.1007/978-3-642-21887-3_45.

[36] Otero FE, Freitas AA, Johnson CG (2009) A hierarchical classification ant colony algorithm for predicting gene ontology terms in *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. doi: 10.1007/978-3-642-01184-9_7.

[37] Aribarg T, Supratid S, Lursinsap C (2012) Optimizing the modified fuzzy Ant-Miner for efficient medical diagnosis. *Applied Intelligence* doi: 10.1007/s10489-011-0332-x.

[38] Bounhas M, Prade H, Serrurier M, Mellouli K (2012) A possibilistic rule-based classifier in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. doi: 10.1007/978-3-642-31709-5_3.

[39] McGarry KJ, Tait J, Wermter S, MacIntyre J (1999) Rule-extraction from radial basis function networks in *Proceedings of the 9th International Conference on Artificial Neural Networks*. Vol. 2, doi: 10.1049/cp:19991178.

[40] Tresp V, Hollatz J, Ahmad S (1997) Representing probabilistic rules with networks of gaussian basis functions. *Machine Learning* doi: 10.1023/A:1007381408604.

[41] Pellegrini P, Stützle T, Birattari M (2012) A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intelligence* doi: 10.1007/s11721-011-0061-0.

[42] Asuncion A, Newman D (2010) UCI machine learning repository (http://archive.ics.uci.edu/ml).

[43] Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7(1):1–30.

[44] Hall M et al. (2009) The WEKA data mining software: An update. *SIGKDD explorations newsletter* doi: 10.1145/1656274.1656278.

[45] Demšar J et al. (2013) Orange: data mining toolbox in Python. *The Journal of Machine Learning Research* 14(1):2349–2353.

[46] Robnik-Šikonja M, Kononenko I (2008) Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering* doi: 10.1109/tkde.2007.190734.

[47] Breiman L (2001) Random forests. *Machine learning* doi: 10.1023/A:1010933404324.