استنتاج درختان تصمیم گیری با یک الگوریتم بهینه سازی کولونی مورچه

چکیده

درختان تصمیم گیری به طور گسترده ای در داده کاوی و یادگیری ماشین به عنوان یک ارائه دانش قابل درک استفاده شده است. در حالی که الگوریتم های بهینه سازی کلونی مورچه (ACO) با موفقیت برای استخراج قوانین طبقه بندی استفاده شده است، استنتاج درخت تصمیم گیری با الگوریتم های ACO هنوز حوزه تحقیقات تقریبا ناشناخته باقی مانده است. در این مقاله ما یک الگوریتم ACO جدید برای استنتاج درختان تصمیم گیری را با ترکیب استفاده شده ترین استراتژی های معمول از هر دو الگوریتم های استنتاج درخت تصمیم گیری سنتی و ACO پیشنهاد می کنیم. الگوریتم پیشنهادی در برابر سه الگوریتم استنتاجی درخت تصمیم گیری مقایسه می شود، یعنی C4.5، CART و cACDT، در ۲۲ مجموعه داده های در دسترس عموم. نتایج نشان داد که دقت پیش بینی کننده الگوریتم پیشنهادی از نظر آماری معنادار, بالاتر از دقت و صحت هر دوی C4.5 و CART است که الگوریتم های معمولی شناخته شده برای استنتاج درخت تصمیم گیری، و دقت الگوریتم درخت تصمیم گیری cACDT ACO است.

کلمات کلیدی: بهینه سازی کولونی مورچه، داده کاوی، طبقه بندی، درخت تصمیم گیری

۱. مقدمه

یکی از مهمترین کارهای داده کاوی مورد مطالعه در ورودی های مختلف, وظیفه طبقه بندی [۱۵،۲۹] است. در اصل، وظیفه طبقه بندی شامل یادگیری یک رابطه پیش بینی کننده بین مقادیر ورودی و یک یک خروجی مورد نظر می شود. هر مثال (نمونه یا ثبت داده ها) توسط یک مجموعه ای از ویژگی های (صفات)–به نام صفات پیش بینی کننده- و یک صفت کلاس توصیف می شود. با توجه به مجموعه ای از نمونه ها، هدف از یک الگوریتم طبقه بندی, ایجاد یک مدل است که نشان دهنده ارتباط بین مقادیر صفات پیش بینی کننده و مقادیر کلاس (برچسب ها)

می باشد و قادر به پیش بینی برچسب کلاس یک مثال جدید (نهان) بر اساس مقادیر صفات پیش بینی کننده آن است. مشکلات طبقه بندی را می توان به عنوان مسائل بهینه سازی در نظر گرفت که در آن هدف, پیدا کردن بهترین تابع (مدل) است که نشان دهنده روابط پیش بینی کننده در داده ها است. مشکل طبقه بندی را می توان به طور رسمی به صورت زیر مشخص نمود:

• با توجه به جفت های $\{(e_1, c_{e_1}), \ldots, (e_n, c_{e_n})\}$ به نمایندگی از داده های آموزشی, D، که در آن هر ei, نشان دهنده مجموعه ای از مقادیر صفات پیش بینی کننده مثال i ام است ($1 \leq i \leq n$, که در آن n, تعداد کل و یا نمونه هاست)، و هر cei نشان دهنده برچسب کلاس مرتبط با مثال iام از m برچسب های کلاس مختلف در دسترس در مجموعه C است.

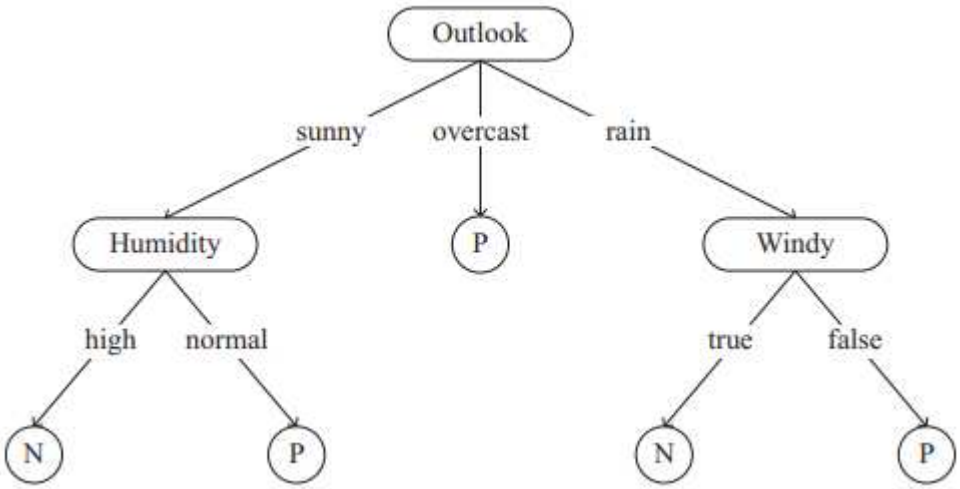
• بیابید: یک تابع $f: D \rightarrow C$ که هر نمونه ei را در D به برچسب کلاس متناظر آن cei در C می نگارد.

هدف اصلی از یک الگوریتم طبقه بندی, ساخت یک مدل است که دقت پیش بینی کننده را حداکثر می کند- تعداد پیش بینی های صحیح در داده های آزمون (نهان در طول آموزش)، اگر چه در بسیاری از حوزه های کاربرد, قابلیت فهم مدل نقش مهمی [۱۵،۸،۱۹] بازی می کند. به عنوان مثال، در تشخیص پزشکی, مدل طبقه بندی باید معتبر باشد و توسط پزشکان تفسیر شود؛ در امتیازدهی اعتبار, مدل طبقه بندی باید توسط یک متخصص تفسیر شود که بهبود اعتماد آنها به مدل می شود؛ در پیش بینی تابع پروتئین, مدل طبقه بندی باید برای ارائه بینش های مفیدی در مورد همبستگی ویژگی های پروتئین و توابع آنها تفسیر شود و در نهایت دانش حال حاضر بیولوژیکی در مورد توابع پروتئین را بهبود بخشد. در این حوزه، تولید مدل های طبقه بندی قابل فهم بسیار حیاتی است.

الگوریتم های بهینه سازی کلونی مورچه (ACO) [11-13] شامل کولونی مورچه ها (عوامل) می شود، که با وجود سادگی نسبی رفتارهای فردی آنها، با یکدیگر برای رسیدن به یک رفتار هوشمند یکپارچه همکاری می کنند. در نتیجه، کولونی, یک سیستم قادر به انجام یک جستجوی قوی برای پیدا کردن با راه حل کیفیت بالا برای مسائل بهینه سازی با یک فضای جستجو بزرگ را تولید می کند. در زمینه وظیفه طبقه بندی در داده کاوی، الگوریتم های

ACO باید دارای مزیت انجام یک جستجوی انعطاف پذیر قوی برای ترکیبی خوب از صفات پیش بینی کننده باشد که به احتمال کمتر از مسئله تعامل صفت [۱۰،۱۷] تحت تاثیر قرار می گیرد.

درختان تصمیم گیری به طور گسترده ای به عنوان یک مدل نمایش قابل فهم استفاده می شوند، با توجه به اینکه می توان آنها را به راحتی در یک فرم گرافیکی و همچنین به عنوان مجموعه ای از قوانین طبقه بندی نشان داد که به طور کلی می توانند در زبان طبیعی در قالب قوانین IFTHEN بیان شوند. بیشتر الگوریتم های ACO برای طبقه بندی در داده کاوی بر استخراج قوانین طبقه بندی [۲۲،۱۸] متمرکز شده اند. در این مقاله، ما یک الگوریتم ACO جدید را برای استنتاج درختان تصمیم گیری پیشنهاد می دهیم. الگوریتم پیشنهادی- به نام Ant-Tree-Miner (استنتاج درخت تصمیم گیری مبتنی بر بهینه سازی کلونی مورچه ها)، در برابر دو الگوریتم استنتاج درخت تصمیم گیری شناخته شده، یعنی C4.5 [۳۱] و CART [۶]، و الگوریتم cACDT بر اساس-ACO 5]] در ۲۲ مجموعه داده عمومی موجود از نظر دقت پیش بینی کننده و اندازه درختان تصمیم گیری استنتاج شده مقایسه می شود.



شکل. ۱. نمونه ای از یک درخت تصمیم گیری، اقتباس شده از [۳۰]. گره های داخلی (از جمله گره ریشه) توسط اسامی و شاخه های صفت نشات گرفته از گره های داخلی متناظر با مقادیر مختلف صفت در یک گره را نشان می دهد. گره های برگ توسط برچسب های کلاس مختلف نشان داده می شوند. در این مثال, سه صفت وجود دارند: چشم انداز {آفتابی، ابری، باران}، رطوبت {بالا، عادی} و بادی {درست غلط}؛ و دو برچسب کلاس {N، P}.

ادامه این مقاله به شرح زیر سازماندهی شده است. بخش ۲ پس زمینه ای از این مقاله را با بحث در مورد استراتژی بالا به پایین معمول استفاده شده برای استنتاج درختان تصمیم گیری, یک مرور کلی از بهینه سازی کلونی مورچه (ACO) و کار مرتبط در الگوریتم ACO برای استنتاج ساختارهای درخت نشان می دهد. الگوریتم پیشنهادی در بخش ۳. توصیف شده است. نتایج محاسباتی در بخش ۴. ارائه شده است. در نهایت، بخش ۵ این مقاله, نتیجه گیری و جهات تحقیقات آینده را ارائه می دهد.

۲. پیش زمینه

۲,۱. استنتاج بالا به پایین درختان تصمیم گیری

درختان تصمیم گیری, یک نمایش گرافیکی قابل فهم از یک مدل طبقه بندی را فراهم می کنند که در آن گره های داخلی مربوط به آزمون های صفت (گره های تصمیم گیری) می شوند و گره های برگ مربوط به برچسب های کلاس پیش بینی شده- نشان داده شده در شکل ۱ می شوند. به منظور طبقه بندی یک مثال، درخت در یک مد از بالا به پایین از گره ریشه به سمت یک گره برگ با توجه به نتیجه آزمایش های صفت ارائه شده توسط گره های داخلی عبور می کند تا زمانی که یک گره برگ به دست آید. در این نقطه، برچسب کلاس مرتبط با گره برگ, برچسب کلاس پیش بینی شده برای مثال است.

یک رویکرد مشترک برای ایجاد درختان تصمیم گیری به صورت خودکار از داده ها, به عنوان یک رویکرد تقسیم و تسخیر شناخته شده است که متشکل از یک رویه بالا به پایین تکراری از انتخاب بهترین صفت برای برچسب گذاری یک گره داخلی درخت است. این رویکرد با انتخاب یک ویژگی به نمایندگی از ریشه درخت شروع می شود. پس از انتخاب اولین صفت، یک شاخه برای هر (مجموعه) امکان پذیر مقدار (های) صفت ایجاد می شود و مجموعه داده ها به زیر مجموعه ها با توجه به مقادیر نمونه "از صفت انتخاب شده تقسیم می شود. سپس رویه انتخاب به صورت بازگشتی برای هر شاخه ای از گره با استفاده از زیر مجموعه مربوطه از نمونه ها به کار گرفته می شود- یعنی، زیر مجموعه با نمونه هایی که دارای مقدار صفت مرتبط با شاخه هستند- و زمانی که همه نمونه ها از زیر مجموعه دارای

همان برچسب کلاس هستند و یا زمانی که معیار توقف دیگر برآورده می شود، یک گره برگ برای نشان دادن یک برچسب کلاس مورد پیش بینی ایجاد می شود. رویکرد تقسیم و تسخیر نشان دهنده یک استراتژی حریص برای ایجاد یک درخت تصمیم گیری است، ازاینرو انتخاب یک صفت در تکرارات اولیه را نمی توان در تکرارهای بعدی دوباره در نظر گرفت- یعنی انتخاب بهترین صفت به صورت موضعی در هر تکرار، بدون در نظر گرفتن تاثیر آن بر تکرارهای بعدی ساخته می شود.

مشکل استنتاج یک درخت تصمیم گیری پس از رویکرد تقسیم-و-تسخیر به مشکلات کوچکتر انتخاب یک صفت مناسب با توجه به مجموعه نمونه ها تقسیم می شود. چند ابتکار برای انتخاب صفات در متون [۲۳،۲۴،۳۳] استفاده شده است. الگوریتم CART [۶] از شاخص Gini به عنوان یک معیار ناخالصی یک صفت استفاده می کند که در آن هرقدر, ناخالصی پایین تر باشد, صفت, بهتر است. Quinlan [۳۰], الگوریتم ID3 را معرفی نمود که صفات را مبتنی بر معیار بهره اطلاعات انتخاب می کند- یک معیار به دست آمده از معیار آنتروپی که معمولاً در نظریه اطلاعات استفاده می شود [۷]. ID3 پیش ساز الگوریتم C4.5 شناخته شده [۳۱] است. اخیراً، یک معیار مبتنی بر فاصله, در الگوریتم CLUS [3،36] استفاده می شود.

الگوریتم C4.5، که احتمالاً بهترین الگوریتم استنتاج درخت تصمیم گیری شناخته شده است، از یک معیار مبتنی بر آنتروپی به منظور انتخاب بهترین صفت برای ایجاد یک گره، به نام نسبت بهره اطلاعات استفاده می کند. در اصل، آنتروپی, ناخالصی مجموعه ای از نمونه ها را نسبت به مقادیر صفت کلاس آنها اندازه گیری می کند که در آن مقادیر آنتروپی بالاتر, با نمونه های یکنواخت تر توزیع شده متناظر است، در حالی که مقادیر آنتروپی پایین متناظر با نمونه های همگن تر (مثالهای بیشتر مرتبط با همان برچسب کلاس). آنتروپی یک مجموعه از نمونه S ها به صورت زیر است

$$\text{Entropy}(S) = \sum_{c=1}^{m} - p_c \cdot \log_2 p_c \,, \qquad (1)$$

که در آن pc, نسبت نمونه ها در S مرتبط با برچسب کلاس cام است و m تعداد کل برچسب های کلاس است. با استفاده از معیار آنتروپی، بهره اطلاعات یک صفت A مربوط به کاهش مورد انتظار در آنتروپی به دست آمده با تقسیم نمونه های آموزشی بر T زیر مجموعه است که در آن T, تعداد مقادیر مختلف در حوزه صفت A است، و به صورت زیر تعریف می شود

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v=1}^{T} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v), \qquad (2)$$

که در آن | SV |, تعداد نمونه ها در زیر مجموعه S است که برای آن, صفت A دارای مقدار vام در دامنه A است و | S | تعداد نمونه ها در S است. محاسبه نسبت بهره اطلاعات شامل جریمه برای صفاتی می شود که از نمونه های آموزشی به زیر مجموعه های بسیار کوچک تقسیم می شوند، به نام Split اطلاعات، که به صورت زیر محاسبه می شود

$$\text{Split Information}(S, A) = \sum_{v=1}^{T} -\frac{|S_v|}{|S|} \cdot \log_2 \frac{|S_v|}{|S|}. \qquad (3)$$

چنین جریمه ای لازم است, زیرا صفاتی که نمونه های آموزش را به زیر مجموعه های بسیار کوچک تقسیم می کنند به احتمال زیاد دارای بهره اطلاعات بالاستت فقط به این خاطر که آنتروپی هر زیر مجموعه به طور مصنوعی کوچک است، با توجه به اینکه تعداد بسیار کمی از نمونه ها در هر زیر مجموعه به راحتی می تواند با یک برچسب کلاس بدون نشان دادن یک توانایی تعمیم خوب مرتبط باشد. یک حالت افراطی, یک صفت خواهد بود که دارای مقداری متفاوت برای هر عنوان مثال آموزش است. این صفت دارای یک بهره اطلاعات بالا خواهد بود، چرا که اگر ما مثال های آموزشی را بر مقادیر تقسیم نماییم، دارای زیر مجموعه هایی با نمونه هایی مرتبط با همان برچسب کلاس خواهیم بود، حتی اگر اندازه هر یک از زیر مجموعه ها, یک باشد. واضح است که این صفت نشان دهنده یک پیش بینی کننده ضعیف (بدون توانایی تعمیم) است و برای طبقه بندی نمونه های نهان مفید نخواهد بود.

در نهایت، نسبت بهره اطلاعات یک صفت A, از بهره و اقدامات اطلاعات Split به دست می آید، و برابر است با

$$\text{Gain Ratio}(S,A) = \frac{\text{Gain}(S,A)}{\text{Split information}(S,A)} \qquad (4)$$

در هر مرحله از رویه بالا به پایین، انتخاب C4.5 به نفع این صفت است که نسبت کسب اطلاعات را حداکثر می کند که مربوط به این صفت است که بهره بزرگتر را از نظر اندازه گیری آنتروپی فراهم می کند. C4.5 با موفقیت در طیف گسترده ای از مشکلات طبقه بندی اعمال شده است و معمولاً در مقایسه های ارزیابی الگوریتم های طبقه بندی جدید [۳۷] استفاده می شود. جزئیات بیشتر از C4.5 را می توان در [۳۱،۳۲] یافت.

۲،۲. بهینه سازی کلونی مورچه

کولونی مورچه ها، با وجود فقدان کنترل متمرکز و سادگی نسبی رفتارهای افراد آنها، سیستم های خود سازمان یافته هستند که می تواند کارهای پیچیده را با داشتن مورچه های فردی خود در تعامل با یکدیگر و با محیط خود انجام دهند. رفتار هوشمند کولونی از ارتباط غیر مستقیم بین مورچه ها به واسطه تغییرات کوچک از محیط پدیدار می شود که به نام نشانه ورزی پدیدار می شود.

بسیاری از گونه های مورچه، حتی با قابلیت های بصری محدود و یا به طور کامل کور، قادر به پیدا کردن کوتاه ترین مسیر بین یک منبع غذایی و لانه با استفاده از فرمون به عنوان یک مکانیسم ارتباطی هستند. مورچه ها, فرمون را بر روی زمین رها می کنند زیرا آنها از یک منبع غذایی به لانه راه می روند و در نتیجه یک دنباله فرمون در مسیر استفاده شده ایجاد می کنند. غلظت فرمون یک مسیر, انتخاب های مورچه ها را را تحت تاثیر قرار می دهند و فرمون بیشتر موجب جذاب تر شدن یک مسیر می شود. با توجه به اینکه مسیرهای کوتاه تر سریع تر از مسیرهای طولانی تر هستند، آنها یک غلظت فرمون قوی تر را پس از یک مدت زمان دارند که به انتخاب شدن و تقویت بیشتر کمک می کند. در نهایت, اکثریت مورچه ها از همان مسیر پیروی خواهند کرد که محتمل ترین و کوتاه ترین مسیر بین منبع غذا و لانه است. با الهام از این رفتار، Dorigo و همکاران [۱۱-۱۳], یک فوق ابتکاری کلونی مورچه مصنوعی را تعریف نموده اند که می توان برای حل مسائل بهینه سازی، به نام بهینه سازی کلونی مورچه (ACO) تعریف کرد.

الگوریتم های ACO از یک کولونی مورچه های مصنوعی استفاده می کنند که در آن مورچه ها, راه حل های کاندید را برای مسائل بهینه سازی شده توسط انتخاب تکراری اجزای راه حل بر اساس فرمون مربوط به آنها و اطلاعات اکتشافی می سازند- که در آن اطلاعات اکتشافی مربوط به اندازه گیری میزان خوب بودن یک جزء راه حل برای این مشکل دردست است. کولونی با استفاده از فرمون برای شناسایی اجزاء برجسته یک راه حل همکاری می کنند و اجزا با غلظت بالاتر از فرومون شانس بیشتری از انتخاب شدن توسط یک مورچه دارد. اجزای مورد استفاده برای ایجاد راه حل های خوب دارای فرمون افزایش یافته است، در حالی که اجزای استفاده نشده, دارای فرمون تدریج کاهش یافته خود خواهد بود. در پایان فرآیند تکرار شونده ایجاد راه حل های نماینده هدایت شده توسط فرمون، کولونی به سمت راه حل های بهینه یا نزدیک به بهینه همگرا می شود. در الگوریتم های ACO، مورچه های مصنوعی, راه حل های نماینده را با عبور از یک نمودار، به نام گراف ساخت می سازند. این نموداری است که در آن هر راس نشان دهنده یک جزء بالقوه از یک راه حل است، و عمل عبور از یک یال بدان معنی است که یک مورچه در حال اضافه کردن راس جزء در پایان یال به راه حل کاندید فعلی است. از این رو، مشکل پیدا کردن بهترین راه حل مربوط به مشکل پیدا کردن بهترین مسیر در گراف ساخت مسئله مورد نظر است. شکل. ۲ شبه کد سطح بالا از یک الگوریتم عمومی ACO را نشان می دهد. چهار روش اصلی درگیر در این قضیه وجود دارد:

```
Input: problem's construction graph
Output: best solution
1. Initialise();
2. while termination condition not met do
3.       ConstructAntSolutions();
4.       ApplyLocalSearch();
5.       UpdatePheromones();
6. end while
7. return best solution;
```

شکل. ۲. شبه کد سطح-بالای یک الگوریتم پایه ACO.

• مقداردهی اولیه: این رویه, پارامترهای الگوریتم را تنظیم می کند و مقدار فرومون (معمولاً به عنوان یک ماتریس فرمون نشان داده می شود) و اطلاعات اکتشافی در ارتباط با راس ها و یا یال های گراف ساخت را مقدار دهی اولیه می نماید.

• ساخت راه حل های مورچه ها: این روش به طور تدریجی راه حل های نماینده را با ایجاد مسیرها بر روی گراف ساخت مسئله با شبیه سازی حرکت یک مورچه مصنوعی ایجاد می کند. با استفاده از یک سیاست تصمیم تصادفی بر اساس استفاده از (وابسته به مسئله) اطلاعات اکتشافی و فرمون, مورچه ها از گراف ساخت مسئله عبور می کنند.

• درخواست جستجوی محلی: این رویه (اختیاری) برای اصلاح بیشتر یک راه حل ایجاد شده توسط یک مورچه استفاده می شود. به طور کلی، اپراتورها / الگوریتم های جستجوی محلی, تغییرات کوچک را برای یک راه حل به منظور کشف راه حل های همسایه معرفی می کنند. Dorigo و Stützle [13] نشان داده اند که برای طیف گسترده ای از مسائل بهینه سازی، استفاده از اپراتورهای / الگوریتم های جستجوی محلی می توانند عملکرد الگوریتم ACO را تقویت نمایند. یک رویه جستجوی محلی, یک نمونه از اقدامات شبح خاص و یا متمرکز بر مسئله (اختیاری) است [۱۳] -یعنی، اقداماتی که می توانند توسط مورچه های فردی انجام شوند.

• به روز رسانی فرومون ها: این رویه, فرمون مرتبط با اجزا- راس و یا یال های- گراف ساخت مسئله را توسط افزایش مقدار فرمون به روز رسانی می نماید، زیرا مورچه ها, فرمون را در مسیر مورد استفاده برای ایجاد راه حل های نماینده خود، و یا کاهش مقدار فرمون، با توجه به شبیه سازی تبخیر فرمون می گذارند. کیفیت یک راه حل نماینده, بر میزان گذاشتن فرومون بر مسیر که نشان دهنده راه حل نماینده است تاثیر می گذارد.

۲,۳. کار مرتبط روی الگوریتم های ACO برای استنتاج ساختارهای درخت

بسیاری از پژوهش های استفاده کننده از الگوریتم های ACO برای کار طبقه بندی در زمینه داده کاوی بر کشف قوانین طبقه بندی متمرکز شده است، همانند الگوریتم Ant-Miner [۲۸] و تغییرات بسیار آن، که به تازگی در [۲۲ ، ۱۸] بررسی شده است. تاکید بر این مورد مهم است که الگوریتم ACO پیشنهادی در این مقاله، به جای مجموعه ای از قوانین، یک درخت تصمیم گیری را می سازد و در نتیجه بسیار متفاوت از Ant-Miner و تغییرات

آن است. بنابراین، در این بخش, ما تنها کار مرتبط در ACO را برای استنتاج ساختارهای درخت-مانند بررسی می کنیم.

Izrailev و 21] Agrafiotis] یک روش مبتنی بر کلونی مورچه را برای ساخت درختان رگرسیون پیشنهاد نمودند. رگرسیون شامل پیدا کردن یک مدل می شود که یک ورودی معین برای یک پیش بینی عددی را می نگارد (به عنوان مثال، صفت هدف, مقادیر پیوسته می گیرد)، در حالی که طبقه بندی شامل پیدا کردن یک مدل می شود که یک مثال معین را به یکی از مجموعه های از پیش تعریف شده از برچسب های گسسته یا اسمی می نگارد (به عنوان مثال، صفت هدف, مقادیر گسسته یا اسمی می گیرد). یک درخت رگرسیون می تواند به عنوان یک مورد خاص از درخت تصمیم گیری در نظر گرفته شود که در آن مقدار پیش بینی شده برای صفت هدف در هر گره برگ درخت, به جای مقدار گسسته یا اسمی, یک مقدار پیوسته است. در روش آنها، یک مورچه, یک درخت رگرسیون را نشان می دهد و ماتریس فرمون توسط یک درخت دودویی مرجع متناظر با اتحاد توپولوژیکی همه درختان ایجادشده نشان داده می شود؛ هر گره تصمیم یک درخت, توسط یک معیار باینری $x_i < v_{ij}$ نشان داده می شود (که در آن vij, مقدار j ام از صفت پیوسته i ام است)؛ فرمون برای انتخاب هر دو صفت و مقدار برای ایجاد یک گره تصمیم گیری استفاده می شود. در نتیجه، روش آنها دارای دو محدودیت مهم است. اول، فقط از صفات پیوسته بر روی گره های تصمیم گیری استفاده می کند (یعنی، نمی تواند با صفات پیش بینی کننده گسسته یا اسمی مقابله نماید). دوم، از اطلاعات اکتشافی استفاده می کند که معمولاً در الگوریتم های ACO استفاده می شود.

به تازگی، [4] Boryczka and Kozak یک الگوریتم کلونی مورچه ها برای ساخت درختان تصمیم دودویی، به نام ACDT را پیشنهاد نمود. یک درخت تصمیم گیری نماینده با انتخاب گره های تصمیم گیری ایجاد می شود- که توسط شرایط دودویی $x_i = v_{ij}$ ایجاد می شود (که در آن Vij, مقدار j ام صفت اسمی iام است) و به تبع آن دقیقا دارای دو یال خروجی (یعنی یکی به نمایندگی از نتیجه «درست» در زمانی که شرایط برآورده می شود و یک به نمایندگی از نتیجه 'غلط' زمانی که شرایط برآورده نمی شود) –مطابق با مقادیر اطلاعات و فرمون اکتشافی می باشد.

```
Input: training examples, list of predictor attributes
Output: best discovered tree
1. InitialisePheromones();
2. ComputeHeuristicInformation();
3. tree_gb ← ∅;
4. m ← 0;
5. while m < maximum iterations and not CheckConvergence() do
6.     tree_ib ← ∅;
7.     for n ← 1 to colony size do
8.         tree_n ← CreateTree(Examples, Attributes, −);
9.         Prune(tree_n);
10.        if Q(tree_n) > Q(tree_ib) then
11.            tree_ib ← tree_n;
12.        end if
13.    end for
14.    UpdatePheromones(tree_ib);
15.    if Q(tree_ib) > Q(tree_gb) then
16.        tree_gb ← tree_ib;
17.    end if
18.    m ← m + 1;
19. end while
20. return tree_gb;
```

شکل ۳. شبه کد سطح-بالا از الگوریتم Ant-Tree-Miner

اطلاعات اکتشافی بر معیار Twoing است که قبلاً در الگوریتم استنتاجی درخت تصمیم گیری CART [۶] استفاده شده است، و مقادیر فرمون نشان دهنده کیفیت ارتباط بین گره های والد و تصمیم گیری کودک می شود. یک فرمت ACDT، به نام cACDT، که می تواند با صفات پیوسته مقابله کند در [۵] ارائه شده است. الگوریتم cACDT در آزمایش های ما برای ارزیابی عملکرد الگوریتم Ant-Tree-Miner گنجانده شده است.

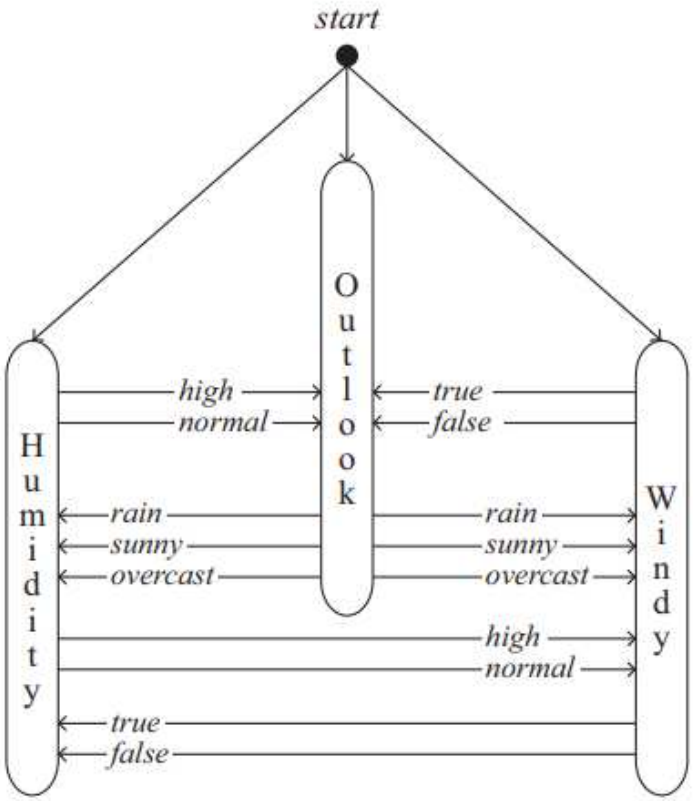
در حالی که تحقیقات در استنتاج درخت تصمیم گیری با الگوریتم های ACO تقریباً ناشناخته باقی مانده است، الگوریتم های تکاملی (غیر ACO) برای استنتاج درخت تصمیم گیری استفاده شده اند [۲]. بر خلاف ACO، الگوریتم های تکاملی از یک اکتشاف موضعی استفاده نمی کنند و جستجوی آنها تنها توسط تابع تناسب اندام (به عنوان مثال، کیفیت کلی درخت تصمیم گیری) هدایت می شود. در الگوریتم های ACO، جستجو توسط هر دو

کیفیت کلی راه حل و اطلاعات اکتشافی محلی هدایت می شود. علاوه بر این، سطوح فرمون, یک بازخورد از کیفیت اجزای راه حل را فراهم می کند و جستجو برای راه حل های برجسته تر را راهنمایی می کند. جزئیات بیشتر الگوریتم های تکاملی برای استنتاج درخت تصمیم گیری را می توان در [۲] یافت.

۳. رویکرد کلونی پیشنهادی مورچه ها برای استنتاج درخت تصمیم گیری

الگوریتم Ant-Tree-Miner پیشنهادی, از ساختار سنتی الگوریتم های ACO پیروی می کند، همانطور که در شکل ۳ ارائه شده است. این کار با مقداردهی مقادیر فرومون و محاسبه اطلاعات اکتشافی برای هر صفت از مجموعه آموزش شروع می شود. سپس، وارد یک حلقه تکرار شونده می شود (حلقه while) که در آن هر مورچه در کولونی, یک درخت تصمیم گیری جدید را ایجاد می کند تا زمانی که حداکثر تعداد تکرارها به دست آید و یا الگوریتم همگرا شود. یک مورچه, یک درخت تصمیم گیری (حلقه for) در مد بالا به پایین را توسط انتخاب احتمالاتی صفت افزوده به عنوان گره های تصمیم گیری بر اساس میزان فرومون (t) و اطلاعات اکتشافی (η) ایجاد می کند. رویه ایجاد درخت تصمیم گیری (روش CreateTree) سه پارامتر، مجموعه ای از نمونه های آموزشی، مجموعه ای از صفات پیش بینی کننده و یال دنبال شده توسط مورچه را در نظر می گیرد که در آغاز این رویه, مربوط به یال پیش فرض است (که با نماد '-' مشخص می شود).

هنگامی که رویه ساخت درخت به پایان می رسد، درخت ایجاد شده به منظور تسهیل درخت هرس می شود و در نتیجه به طور بالقوه از برازش بیش از حد مدل برای داده های آموزشی جلوگیری می کند. برازش بیش از حد, پدیده ای است که در آن مدل برای داده های آموزشی تنظیم می شود و در نتیجه دارای دقت پیش بینی خوب در مجموعه آزمون نیست، و در طول آموزش نهان است. از آنجا که گره های تصمیم گیری, در حالی به درخت اضافه می شوند که صفات دردسترس هستند و مجموعه نمونه ها در گره شامل نمونه هایی از بیش از روی برچسب کلاس می شوند، درخت معمولا بسیار پیچیده است- متشکل از تعداد زیادی از گره ها-که می تواند توان تعمیم آن بر روی داده های نهان را را تحت تاثیر قرار دهد. پس از هرس، درخت ارزیابی می شود و درخت (treeib) بهترین- تکرار به روز می شود، در صورتی که کیفیت درخت به تازگی ایجاد شده بیشتر از کیفیت بهترین-تکرار درخت جاری است.



شکل. ۴. نمونه ای از یک گراف ساخت که می تواند برای تولید درخت تصمیم گیری ارائه شده در شکل ۱ مورد استفاده قرار گیرد. در این مثال، گراف ساخت توسط سه راس به نمایندگی از صفات اسمی تشکیل می شد: چشم انداز {آفتابی، ابری، باران}، رطوبت {بالا، عادی} و {بادخیز درست، غلط}؛ و یک "شروع" راس مجازی.

در نهایت، درخت بهترین-تکرار ساخته شده توسط مورچه ها برای به روز رسانی مقدار فرومون استفاده می شود، درخت بهترین-کلی (treegb) ذخیره سازی / به روز می شود و تکرار جدیدی از الگوریتم شروع می شود. زمانی که حداکثر تعداد تکرار به دست آید یا الگوریتم همگرا شود (روش CheckConvergence)، درخت بهترین-کلی به عنوان درخت تصمیم گیری کشف شده بازگردانده می شود.

۳,۱. گراف ساخت

گراف ساخت شامل n راس می شود که نشان دهنده صفات پیش بینی کننده، یک راس در هر صفت است. این راس ها توسط یال های مربوط به شرایط مختلف شامل مقادیر حوزه صفت متصل می شوند که یال از آن سرچشمه می

گیرد. علاوه بر این، گراف ساخت دارای یک راس 'شروع' مجازی می شود. یک مورچه همیشه, ایجاد یک درخت تصمیم گیری نماینده را از گره 'شروع' شروع می کند و N یال متصل کننده گره 'شروع' به هر راس صفت xi گراف ساخت وجود دارد. شکل. ۴ یک نمونه از گراف ساخت را نشان می دهد.

برای صفات اسمی، هر یال نشان دهنده وضعیتی است که در آن صفت xi دارای مقدار vij است-یعنی،. اصطلاح xi=vij. توجه داشته باشید که یک مورچه نمی تواند همان راس صفت اسمی xi را چند بار در همان مسیر از درخت انتخاب کند- یک مسیر در این زمینه به عنوان لیستی از رئوس صفت بین یک گره و گره ریشه از درخت-برای جلوگیری از تناقضاتی، مانند انتخاب از شرایط، چشم انداز = آفتابی »و« نگاه = باران تعریف می شود. از این رو، N-1 یال برای هر جفت $x_i = v_{ij}$ متصل کننده راس صفت x] به N-1 رئوس صفت باقیمانده وجود دارد, یعنی، برای هر مقدار vij در حوزه صفت xi، یک یال $x_i = v_{ij}$ متصل کننده راس صفت xi به راس صفت xk وجود دارد که در آن $i \neq k$.

صفات پیوسته نشان دهنده یک مورد خاص از رئوس صفت در گراف ساخت است. با توجه به اینکه یک صفت پیوسته دارای یک مجموعه (پیش تعریف شده) از فواصل زمانی ثابت برای تعریف شرایط صفت نیست و در نتیجه نشان دهنده یال های نشات گرفته از راس آن است، یک روش گسسته سازی پویا برای ایجاد فواصل گسسته به عنوان شرایط صفت استفاده می شود. رویه گسسته سازی از یک رویکرد مشابه روش به کار گرفته شده در الگوریتم استنتاج قاعده cAnt-Miner بر اساس ACO-[25، 26] شرح زیر پیروی می کند: ما از دو روش متفاوت برای گسسته سازی مقادیر صفات پیوسته استفاده کرده ایم:

۱. رویکرد اول, همان رویکرد به کار گرفته شده در C4.5 است، که در آن مقدار آستانه انتخاب شده متناظر با مقداری است که بالاترین بهره اطلاعات را تولید می کند. هر صفت اول پیوسته دارای مقادیر طبقه بندی شده آن در مرتبه افزایشی است. مقادیر آستانه نماینده, مقادیر در حوزه صفات پیوسته هستند که در نمونه های مجاور رخ می دهند-که مطابق با مقادیر دسته بندی شده- مرتبط با برچسب های مختلف کلاس است. به عنوان مثال، یک

مجموعه داده فرضی با ۳ نمونه با مقادیر صفت سن {۲۵، ۲۶، ۲۷} همراه با برچسب کلاس {بله، بله، هیچ} را در نظر بگیرید. یک مقدار آستانه نماینده برای آن نمونه ها ۲۶ است، اما ۲۵ نیست، زیرا نمونه ها با مقادیر سن مقدار ۲۵ و ۲۶ دارای برچسب کلاس یکسان هستند. مقدار آستانه t, نمونه های آموزشی را به دو زیر مجموعه در رابطه با مقادیر صفت پیوسته xi تقسیم می کند, که دارای مقدار xi کمتر از حد آستانه (xi<t) و مقادیر دارای یک مقدار بزرگتر از یا برابر با آستانه ($x_i \geq t$) می باشد. این رویکرد همچنین در الگوریتم استنتاج قاعده cAnt-Miner بر اساس ACO-[25] استفاده می شود.

۲. رویکرد دوم بر اساس اصل حداقل طول توصیف (MDL) است و توسط Fayyad و Irani [۱۴] پیشنهاد شد و در زمینه استنتاج درختان تصمیم گیری مورد استفاده قرار گرفت. در گسسته سازی مبتنی بر MDL، فواصل متعدد گسسته را می توان با استفاده از یک رویکرد گسسته سازی باینری استخراج نمود به عنوان مثال، رویکرد گسسته سازی مبتنی بر بهره اطلاعات مورد استفاده در C4.5-به صورت بازگشتی، انتخاب بهترین مقدار آستانه در هر تکرار، و استفاده از اصل استخراج MDL به عنوان یک معیار توقف برای تعیین اینکه آیا مقادیر آستانه باید معرفی شوند یا خیر. این رویکرد نیز در یک فرمت از الگوریتم استنتاج قاعده cAnt-Miner مبتنی بر ACO-[26] استفاده شده است.

توجه داشته باشید که این گراف ساخت پویا را می سازد، زیرا گسسته سازی صفات پیوسته برای مسیر فعلی تنظیم می شود که با دنبال نمودن یک مورچه و زیر مجموعه حال حاضر از مثال های آموزشی در روش گسسته مورد استفاده قرار می گیرد. علاوه بر این، یک صفت پیوسته را می توان چندین بار در همان مسیر درخت تصمیم گیری انتخاب نمود که به الگوریتم, فرصت اصلاح انتخاب فاصله گسسته را می دهد. برای هر بازه گسسته از یک صفت xi پیوسته تولید شده توسط رویه گسسته سازی پویا، N یال متصل کننده راس صفت xi به هر صفت راس xk دیگر (از جمله صفت راس xi) به گراف ساخت اضافه می شوند. از آنجا که روش گسسته سازی, تعیین کننده است، هنگامی که یک مورچه از همان مسیر به یک راس صفت به نمایندگی از یک صفت پیوسته پیروی می کند، همان یال ها استفاده خواهند شد.

۳,۲. اطلاعات اکتشافی

اطلاعات اکتشافی در ارتباط با هر راس صفت xi گراف ساخت متناظر با کیفیت برآوردشده آن با توجه به توانایی آن برای بهبود بخشیدن به دقت پیش بینی کننده درخت تصمیم گیری است. Ant-Tree-Miner از همان اطلاعات اکتشافی الگوریتم استنتاجی درخت تصمیم گیری C4.5 شناخته شده، یعنی نسبت بهره اطلاعات صفات [۳۱] استفاده می کند که با عبارت زیر داده می شود

$$\eta_{x_i} = \text{Gain Ratio}(S, x_i), \quad (5)$$

که در آن xi مربوط به راس صفت i ام و S مربوط به مجموعه نمونه های آموزشی است. تابع نسبت بهره در معادله (۴) توصیف می شود.

به منظور محاسبه نسبت بهره اطلاعات صفات پیوسته، انتخاب مقادیر آستانه پویا برای تعریف فواصل گسسته و پس از آن تقسیم نمونه آموزش به زیر مجموعه ها لازم است. هنگامی که مقادیر آستانه یک صفت پیوسته تولید می شوند و فواصل گسسته آن تعریف می شوند، نسبت بهره اطلاعات داده شده توسط معادله (۴) با این فرض قابل محاسبه است که هر فاصله گسسته نشان دهنده یک مقدار متفاوت و در نتیجه مجموعه دیگری از مثال های آموزشی باشد. گسسته سازی یک صفت پیوسته, یک تبدیل موقت با هدف محاسبه نسبت بهره اطلاعات خود است و مقادیر صفت پیوسته توسط فواصل گسسته جایگزین نمی شوند.

اگر چه هر دو الگوریتم های Ant-Tree-Miner و C4.5 از معیار نسبت بهره اطلاعات به عنوان یک روش اکتشافی برای انتخاب صفت استفاده می کنند، Ant-Tree-Miner با استفاده از نسبت بهره اطلاعات در کنار فرمون نسبت می دهد، در حالی که C4.5 تنها از نسبت بهره اطلاعات استفاده می کند. استفاده از فرمون, یک بازخورد دقیق از کیفیت یک صفت با توجه به اثرات آن در درخت تصمیم گیری کل (نسبت ارزیابی «کلی») را فراهم می کند و بنابراین می تواند بی دقتی های معیار نسبت وزن اطلاعات حریص (نسبت ارزیابی "محلی") را جبران نماید.

Input: training examples (*Examples*), list of predictor attributes (*Attributes*), current edge (*Edge*)
Output: root node of the decision tree

1. $A \leftarrow$ probabilistically selects an attribute from *Attributes* to visit given the current *Edge*;
2. *root* $\leftarrow$ creates a new decision node representing attribute A;
3. *conditions* $\leftarrow \emptyset$;
4. **if** A is a nominal attribute **then**
5. $\quad$ *Attributes* $\leftarrow$ *Attributes* $- \{A\}$;
6. $\quad$ **for all** value v_i in domain of A **do**
7. $\quad\quad$ *conditions* $\leftarrow$ *conditions* $+ \{A = v_i\}$;
8. $\quad$ **end for**
9. **else**
10. $\quad$ *conditions* $\leftarrow$ *Discretise(A, Examples)*;
11. **end if**
12. **for all** attribute condition T in conditions **do**
13. $\quad$ $branch_i \leftarrow$ new branch representing T of *root*;
14. $\quad$ $subset_i \leftarrow$ subset of *Examples* that satisfies T;
15. $\quad$ **if** $subset_i$ is empty **then**
16. $\quad\quad$ Add a leaf node with the majority class label of *Examples* below $branch_i$;
17. $\quad$ **else if** all examples in $subset_i$ have the same class label **then**
18. $\quad\quad$ Add a leaf node with the class label of $subset_i$ below $branch_i$;
19. $\quad$ **else if** number of examples in $subset_i$ is below a threshold **then**
20. $\quad\quad$ Add a leaf node with the majority class label of $subset_i$ below $branch_i$;
21. $\quad$ **else if** Attributes is empty **then**
22. $\quad\quad$ Add a leaf node with the majority class label of $subset_i$ below $branch_i$;
23. $\quad$ **else**
24. $\quad\quad$ Add the subtree returned by *CreateTree*($subset_i$, *Attributes*, $branch_i$) below $branch_i$;
25. $\quad$ **end if**
26. **end for**
27. **return** *root*;

شکل ۵. شبه کد سطح-بالای رویه ساخت درخت تصمیم گیری CreateTree (نمونه ها, صفت, یال) استفاده شده در Ant-Tree-Miner

۳,۳. ساخت راه حل

ساخت یک درخت تصمیم گیری نماینده از یک رویکرد تقسیم و تسخیر پیروی می کند، با این تفاوت که، به جای شباعت قطعی در یک الگوریتم درخت تصمیم گیری استنتاج معمولی, صفات به طور تصادفی مبتنی بر مقادیر اطلاعات و فرمون اکتشافی انتخاب می شوند. در هر تکرار فرآیند ساخت، یک مورچه, یک قاعده احتمالاتی را به کار می برد تا تصمیم بگیرد که که کدام راس صفت مبتنی بر میزان فرومون و اطلاعات اکتشافی بازدید شود. احتمال pi یک مورچه برای بازدید از صفت راس xi برابر است با

$$p_i = \frac{\tau_{(E,L,x_i)} \cdot \eta_i}{\sum_{i \in \mathcal{F}} \tau_{(E,L,x_i)} \cdot \eta_i}, \quad \forall i \in \mathcal{F}, \qquad (6)$$

که در آن:

- $\tau_{(E,L,x_i)}$, میزان فرمون مرتبط با ورودی است (E,L,xi)-E, وضعیت صفت نشان داده شده توسط یال پیروی شده یا – در آغاز رویه ساخت است, L سطح کنونی مورچه در درخت تصمیم گیری یا ۰ در آغاز رویه ساخت است, xi, راس صفت گراف ساخت- در ماتریس فرمون است.
- η_i اطلاعات ابتکاری صفت iام است؛
- F مجموعه صفات دردسترس (عملی) برای انتخاب است.

α و β معمولاً استفاده شده برای کنترل تاثیر فرمون و اطلاعات اکتشافی، به ترتیب، در انتخاب رئوس در ۱ تنظیم می شوند و در نتیجه از معادله (۶) حذف می شوند.

شکل. ۵ یک شبه کد سطح بالا از رویه ساخت درخت تصمیم گیری مورد استفاده در Ant-Tree-Miner را نشان می دهد. یک مورچه, ساخت یک درخت تصمیم گیری نماینده را پیرو یال '-' نشات گرفته از گره مجازی 'شروع'، با مجموعه ای کامل از نمونه های آموزشی و مجموعه ای کامل از صفات پیش بینی کننده آغاز می کند. صفت انتخاب برای ایجاد یک گره تصمیم گیری استفاده می شود. بسته به نوع صفت انتخاب شده، دو مجموعه متفاوت از دستورالعمل اجرا می شوند. اگر صفت انتخاب شده, یک صفت اسمی باشد، از مجموعه ای از صفات پیش بینی کننده حذف می شود و لیستی از شرایط با شرایط صفت A=vi برای تمام مقادیر vi در حوزه صفت A پر می شود. اگر صفت انتخاب شده, یک صفت پیوسته باشد، لیستی از شرایط توسط فواصل گسسته تولید شده توسط روش گسسته سازی نشان داده می شود. گسسته سازی یک صفت پیوسته در مجموعه کنونی نمونه های آموزشی تنظیم می شود، همانطور که قبلا توضیح داده شد. در مقابل صفات اسمی، صفات پیوسته از مجموعه از صفات پیش بینی کننده حذف نمی شوند, زیرا آنها می توانند چندین بار، در همان مسیر درخت، توسط یک مورچه انتخاب شوند.

هنگامی که لیست شرایط صفت تعریف می شود، هر یک از شرایط برای ایجاد شاخه ای از گره تصمیم گیری مورد استفاده قرار می گیرد. بنابراین، مجموعه ای از نمونه های آموزش به یکی از زیر مجموعه ها برای هر وضعیت صفت (واحد) تقسیم می شود که در آن هر زیرمجموعه شامل مثالهای آموزشی برآورده کننده صفت متناظر تقسیم می

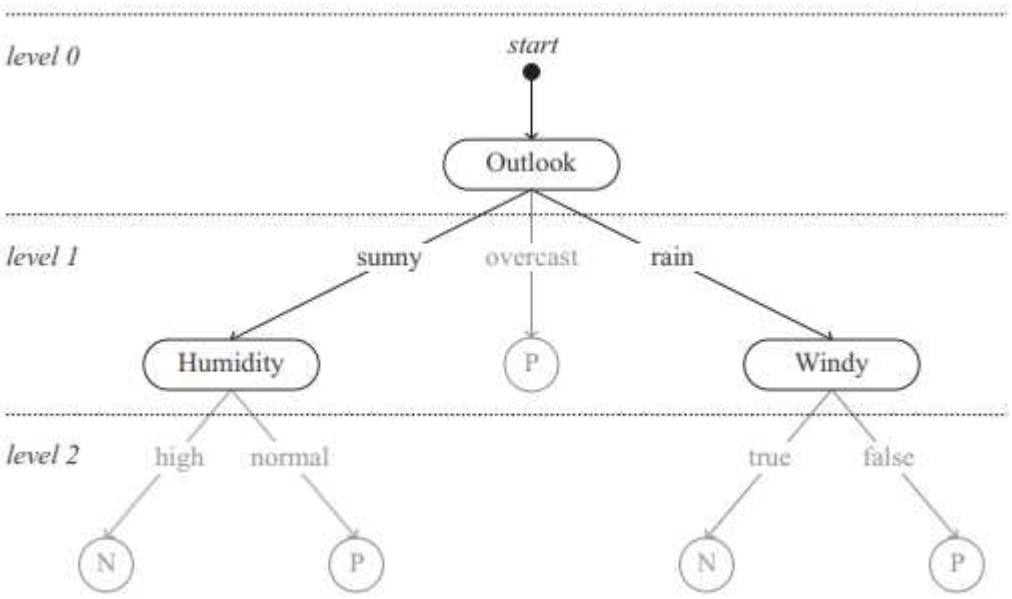
شود. در این نقطه، رویه ساخت چک می کند که آیا یک گره برگ باید زیر شاخه فعلی اضافه شود یا خیر و یا آیا باید به صورت بازگشتی یک زیردرخت را به زیر شاخه جاری اضافه کند یا خیر. تصمیم اضافه کردن یک گره برگ به درخت تصمیم گیری نماینده قطعی است و مبتنی بر شرایط زیر است:

۱. زیر مجموعه حال حاضر از مثالهای آموزشی, خالی است، که مربوط به موردی است که هیچ یک از مثال های آموزشی, شرایط صفت ارائه شده توسط شاخه فعلی را برآورده نمی سازند؛

۲. همه نمونه ها در زیر مجموعه حال حاضر با برچسب همان کلاس در ارتباط است.

۳. تعداد نمونه ها در زیر مجموعه, زیر یک آستانه تعریف شده توسط کاربر است.

۴. مجموعه صفات پیش بینی کننده موجود, خالی است.



شکل. ۶. نمونه ای از یک درخت تصمیم گیری ایجاد شده توسط یک مورچه؛ گره 'شروع' نشان دهنده راس آغازین است. شاخه ها-شاخه های برجسته که منجر به یک گره تصمیم گیری می شوند، از جمله گره ریشه- آنهایی هستند که در نهایت برای به روز رسانی فرومون گره استفاده خواهند شد. شاخه هایی که گره ترمینال آنها, یک گره برگ است، مانند شاخه ها در سطح ۲ در شکل، نتیجه رویه هرس قطعی هستند و یا زمانی که تمام نمونه ها پیروی کننده از شاخه با همان برچسب کلاس در ارتباط هستند، و بنابراین آنها در ماتریس فرومون نشان داده نمی شوند.

اگر هر یک از شرایط فوق مشاهده می شود، یک گره برگ به نمایندگی از یک پیش بینی برچسب کلاس در زیر شاخه فعلی اضافه می شود. در غیر این صورت، رویه ساخت به صورت بازگشتی برای ایجاد یک زیر درخت با توجه به زیر مجموعه ای از مثال های آموزشی، مجموعه حال حاضر از صفات پیش بینی کننده و شاخه جاری اعمال می شود. در نهایت، گره ریشه درخت تصمیم گیری نماینده در پایان روش ساخت بازگردانده می شود.

۳,۴. هرس

پس از اینکه یک درخت تصمیم گیری نماینده ایجاد می شود، تحت یک روش هرس قرار می گیرد. هدف از رویه هرس, بهبود قدرت تعمیم درخت تصمیم گیری و در نتیجه دقت پیش بینی آن با استفاده از حذف گره های تصمیم های غیر ضروری از درخت است. به طور کلی، یک درخت نماینده ایجاد شده توسط رویه ساخت, بیش از حد پیچیده است، زیرا گسترش می یابد, تا زمانی که هیچ صفات موجود و یا مثال های آموزشی وجود نداشته باشد، و یا همه نمونه های آموزش با همان برچسب کلاس در ارتباط باشند. اشکال اصلی داشتن یک درخت پیچیده اینست که درخت به احتمال زیاد با نمونه های آموزش بیش از حد متناسب باشد و دقت پیش بینی آن در نمونه های نهان ضعیف خواهد بود. علاوه بر این، روش ایجاد درخت تصمیم گیری, تصادفی است و درخت تولید شده معمولا بهترین تناسب برای داده های آموزشی را نشان نمی دهد. این تفاوت اصلی بین یک درخت هرس نشده تولید شده توسط C4.5 و Ant-Tree-Miner است. یک درخت تصمیم گیری هرس نشده C4.5 نشان دهنده بهترین تناسب برای داده های آموزشی با استفاده از روش ایجاد قطعی (حریص) مبتنی بر نسبت بهره اطلاعات است. بنابراین، رویه هرس باید به منظور افزایش قدرت تعمیم درخت تصمیم گیری استفاده شود. در Ant-Tree-Miner ، رویه هرس دارای دو وظایف است و به دو مرحله تقسیم می شود: مرحله اول, درخت برای تناسب با داده های آموزشی هرس می شود که به طور بالقوه منجر به برازش بیش از حد داده های آموزشی خواهد شد، و مرحله دوم, درخت را به منظور افزایش قدرت تعمیم آن هرس می کند.

گام اول شامل جایگزین کردن یک گره تصمیم گیری توسط هر رایج ترین شاخه استفاده شده آن و یا توسط یک گره برگ است که منجر به یک دقت طبقه بندی بالاتر در داده های آموزشی می شود. هدف از این مرحله, حذف

گره تصمیم گیری است که توسط رویه ایجاد تصادفی اضافه شده است که دارای یک اثر منفی بر دقت طبقه بندی درخت تصمیم گیری است. این مرحله تا زمانی تکرار می شود که هیچ یک از جایگزین ها منجر به بهبود در صحت و یا درخت شامل فقط یک گره برگ می شود.

مرحله دوم شامل رویه هرس مشابه مورد استفاده در C4.5، به نام هرس مبتنی بر خطا [۳۱] می شود. در اصل، این کار شامل جایگزینی یک گره تصمیم گیری توسط هر رایج ترین شاخه استفاده شده آن و یا یک گره برگ می شود. اگر جایگزینی منجر به نرخ خطای برآوردشده کمتر شود(جزئیات برآورد میزان خطا را می توان در [۳۱، ص ۴۱] یافت)، درخت تصمیم گیری بر این اساس هرس می شود. در مقابل گام قبلی، این تصمیم بر اساس میزان خطای برآوردشده است (معیاری که یک جریمه بیشتر را برای خطاها نشان می دهد و نه اندازه گیری دقت) به جای دقت طبقه بندی در مثال های آموزشی، زیرا استفاده از دقت طبقه بندی می تواند منجر به برازش بیش از حد شود- موردی که در آن درخت تصمیم گیری بیش از حد برای داده های آموزشی تنظیم می شود و به خوبی تعمیم نمی یابد، یعنی، دارای دقت پیش بینی کننده پایین تر در داده های آزمون است. منطق پشت این جایگزینی اینست که اگر میزان خطای هر زیر درخت کاهش یابد, به طور کلی (برآورد) میزان خطای درخت کاهش می یابد. هدف از این مرحله, بهبود قدرت تعمیم درخت با کاهش نرخ خطای برآوردشده آن در داده های آموزشی است. توجه داشته باشید که میزان خطای کمتر تخمین زده شده به یک دقت طبقه بندی بالاتر در داده های آموزشی مرتبط نمی شود و یک گره تصمیم گیری که منجر به یک میزان خطای کمتر برآوردشده نمی شود توسط این مرحله جایگزین نمی شود حتی اگر دقت در داده های آموزشی کاهش یابد-دقت در داده های آموزشی توسط این مرحله محاسبه نمی شود.

۳,۵. نمایندگی ماتریس فرمون

به منظور اینکه مورچه, یک درخت تصمیم گیری را خلق نماید، باید چند مسیر را از گره ریشه به سمت گره های برگ دنبال نماید. گره های تصمیم گیری یک درخت توسط رئوس صفت گراف ساخت نشان داده می شوند و هر شاخه نشات گرفته از یک گره تصمیم گیری توسط یک یال گراف ساخت ارائه می شود. به یاد داشته باشید که هر

یال گراف ساخت توسط یک شرط صفت نشان داده می شود و دارای یک راس مقصد است که راسی است که یال منجر به آن می شود. از آنجا که انتخاب های انجام شده توسط یک مورچه را می توان با یال هایی بیان نمود که در طول ایجاد یک درخت تصمیم گیری دنبال می شوند، هر ورودی در ماتریس فرمون توسط یک سه گانه [edgeij، سطح، xk] نمایش داده می شود که در آن edgeij یال نشاندهنده وضعیت صفت jام از صفت xi است، سطح, سطح درخت تصمیم گیری است که در آن edgeij به نظر می رسد و xk راس صفت مقصد آن است. اطلاعات سطح یال با یک ورودی برای تبعیض بین وقوع های متعدد از همان نوع یال (یعنی شرایط صفت همان) در سطوح مختلف از درخت، یا برای ظهور در همان درخت مسیر ممکن در مورد یال های صفت راس-یا در مسیرهای متفاوت درخت مرتبط می شود. با توجه به اینکه یال ها برای رئوس صفات پیوسته "به صورت پویا ایجاد می شوند، ماتریس فرومون یک ساختار افزایشی است که نشان دهنده نگاشت یک سه گانه [edgeij، سطح، xk] (کلید نگاشت) و یک مقدار فرومون (مقدار نقشه برداری) است.

بازنگری نمونه ای از یک درخت تصمیم گیری ارائه شده در شکل ۱، شکل. ۶ نشان می دهد که درخت تصمیم گیری نماینده متناظر توسط یک مورچه ایجاد می شود. در این مثال، شاخه ای شدن برجسته شده-شاخه هایی که منجر به یک گره تصمیم گیری می شوند، از جمله گره های ریشه – و آنهایی که در نهایت برای به روز رسانی فرومون ها استفاده خواهند شد, و آنها توسط همه ورودی ها نشان داده شده اند:

• [-، ۰، چشم انداز]: این ورود در ماتریس فرومون برای یال اتصال دهنده گره 'شروع' مجازی به گره 'چشم انداز'، گره ریشه درخت است. گره 'شروع' به عنوان مجازی اشاره می شود, زیرا در درخت تصمیم گیری نهایی گنجانده نمی شود. برای مرتبط نمودن فرمون در یال استفاده می شود که منجر به گره ریشه یک درخت تصمیم گیری نماینده می شود. یال های نشات گرفته در گره 'شروع', یک شرط صفت را نشان نمی دهد. آنها به عنوان edgestart نامیده می شوند و توسط نماد '-' نشان داده می شوند؛

• [چشم انداز = آفتابی، ۱، رطوبت]: این ورود در ماتریس فرمون برای یال قرار داده شده در سطح ۱ به نمایندگی از وضعیتی است که صفت "چشم انداز" دارای مقدار برابر با «آفتابی» است که راس صفت «چشم انداز» به راس صفت "رطوبت" متصل می کند.

• [چشم انداز = باران، ۱، بادی]: این ورود در ماتریس فرمون برای یال قرار گرفته در سطح ۱ به نمایندگی از وضعیتی است که در آن صفت "چشم انداز" دارای مقدار برابر با «باران» است، که صفت راس "چشم انداز" را با صفت راس "بادی"متصل می کند.

یال های باقی مانده در مثال در شکل ۶ در طول به روز رسانی فرومون استفاده نمی شوند و در ماتریس فرمون نشان داده نمی شوند, زیرا آنها به طور مستقیم به یک گره برگ به نمایندگی از یک پیش بینی برچسب کلاس منجر می شوند و به طور قطعی در درخت تصمیم گیری درطی فرآیند ایجاد درخت یا با رویه هرس کردن معرفی می شوند.

جدول ۱ خلاصه مجموعه داده های استفاده شده در آزمایشات

auto	automobile	10	15	7	205
balance	balance scale	4	0	3	625
blood-t	blood transfusion	0	4	2	748
breast-l	breast cancer ljubljana	9	0	2	286
breast-t	breast tissue	0	9	6	106
breast-w	breast cancer wisconsin	0	30	2	569
credit-a	credit approval	8	6	2	690
derm	dermatology	33	1	6	366
ecoli	ecoli	0	7	8	336
glass	glass	0	9	7	214
heart-c	heart cleveland	6	7	5	303
heart-h	heart hungarian	6	7	5	294
horse	horse colic	15	7	2	368
hep	hepatitis	13	6	2	155
ionos	ionosphere	0	34	2	351
iris	iris	0	4	3	150
park	parkinsons	0	22	2	195
s-heart	statlog heart	7	6	2	270
soybean	soybean	35	0	19	307
voting	voting records	16	0	2	435
wine	wine	0	13	3	178
zoo	zoo	16	0	7	101

۳,۶. به روز رسانی فرمون

به روز رسانی مقادیر فرمون در Ant-Tree-Miner تحت حاکمیت همان رویکرد سیستم MAX-MIN ANT (MMAS) است [۳۴،۳۵]. در MMAS، مقادیر دنباله فرمون به بازه $[\tau_{\min}, \tau_{\max}]$ محدود می شوند. این محدودیت ها به صورت پویا در هر زمان که بهترین راه حل جدید کلی پیدا می شود به روز می شوند، همانطور که در [۳۴] به تفصیل بیان شده است. همچنین آنها برای تعیین رکود جستجو استفاده می شوند. هنگامی که تمام ورودی ها در ماتریس فرمون استفاده شده توسط یک مورچه برای ایجاد درخت بهترین تکرار $\tau_{\max}$ مرتبط می شوند و ورودی های باقی مانده با $\tau_{\min}$، مرتبط می شوند, جستجو, راکد در نظر گرفته می شود و الگوریتم متوقف می شود.

با توجه به بهترین درخت تصمیم گیری نماینده یک تکرار (راه حل نماینده بهترین-تکرار)، به روز رسانی فرمون در دو مرحله انجام می شود. در مرحله اول، تبخیر فرمون با کاهش میزان فرومون هر ورودی در ماتریس فرومون توسط یک ضریب p انجام می شود (یک پارامتر تعریف شده توسط کاربر). در مرحله دوم، میزان فرومون ورودی های متناظر با شاخه های مورد استفاده در راه حل کاندید بهترین-تکرار مبتنی بر کیفیت راه حل نماینده افزایش می یابند. کیفیت یک درخت تصمیم گیری بر اساس همان معیار مورد استفاده در روش هرس است و به صورت زیر داده می شود

$$Q = \frac{N - \text{Error}}{N}, \qquad (7)$$

که در آن N, تعداد نمونه های تعلیم است و خطا, تعداد تخمینی از خطاهای طبقه بندی درخت است, هرقدر تعداد خطاها کمتر باشد, کیفیت درخت بهتر است. کیفیت یک درخت با توجه به معادله ۷, به بازه [0,1] محدود می شود. در نهایت, قاعده به روزسازی فرمون به صورت زیر می شود

$$\tau_{(E,L,x_i)} = \begin{cases} \rho \cdot \tau_{(E,L,x_i)}, & \text{if } (E, L, x_i) \notin \text{tree}_{ib}; \\ \rho \cdot \tau_{(E,L,x_i)} + Q(\text{tree}_{ib}), & \text{if } (E, L, x_i) \in \text{tree}_{ib}; \end{cases} \qquad (8)$$

که ρ , عامل تبخیر MAX-MIN است, $\tau_{(E,L,x_i)}$, میزان فرمون مرتبط با ورودی (E, L, x_i) است- E, شرط صفت لبه است که این ورودی متناظر با آن است, L سطحی است که در آن لبه رخ می دهد و xi, راس صفت مقصد لبه است- و tree ib, درخت تصمیم گیری بهترین-تکرار است.

۴. نتایج محاسباتی

الگوریتم پیشنهادی Ant-Tree-Miner برابر دو الگوریتم استنتاجی درخت تصمیم گیری شناخته شده اجرا در میز کار WEKA [37]، یعنی C4.5 [31] (الگوریتم J48 WEKA) و CART [۶] (الگوریتم SimpleCART WEKA است) مقایسه شد ، و در برابر الگوریتم درخت تصمیم گیری cACDT مبتنی بر ACO-[5]. ما چهار تغییرات الگوریتم Ant-Tree-Miner را داشته باشد:

۱. استفاده از یک روش گسسته سازی باینری مبتنی بر آنتروپی و هرس درخت دو مرحله ای، که توسط Ant-Tree-Miner مشخص می شود.

۲. استفاده از یک رویه گسسته سازی باینری مبتنی بر آنتروپی و فقط رویه هرس مبتنی بر خطای C4.5، که توسط P-Ant-Tree-Miner مشخص می شود (که در آن "P- 'نشان دهنده الگوریتم بدون مرحله هرس مبتنی بر دقت است).

۳. با استفاده از یک روش گسسته سازی مبتنی بر MDL و هرس درختان دو مرحله است مشخص شده توسط مورچه درخت MinerMDL؛

۴. با استفاده از یک روش گسسته سازی مبتنی بر MDL و فقط رویه هرس مبتنی بر خطا C4.5 که توسط Ant-P MDL-Tree-Miner مشخص می شود.

جدول ۲

دقت پیش بینی کننده متوسط (خطای میانگین _+ استاندارد), برحسب ٪, اندازه گیری شده توسط اعتبار-مقطعی ده تایی. ستون های نشان داده شده توسط ATM با تغییرات ANT-Tree-Miner متناظر هستند. مقدار دقیق ترین الگوریتم برای هر مجموعه داده به صورت توپر نشان داده شده است.

Data set	C4.5	CART	cACDT	ATM	ATM^{-P}	ATM_{MDL}	ATM^{-P}_{MDL}
auto	**81.4 ± 2.5**	76.6 ± 2.6	66.0 ± 0.6	77.2 ± 0.4	77.4 ± 0.6	74.8 ± 0.5	76.2 ± 0.5
balance	63.7 ± 2.2	79.3 ± 1.0	**79.4 ± 0.2**	62.9 ± 0.1	62.9 ± 0.1	62.9 ± 0.1	62.9 ± 0.1
blood-t	73.8 ± 1.3	**74.2 ± 1.2**	71.7 ± 0.0	72.4 ± 0.1	72.3 ± 0.1	72.1 ± 0.1	72.4 ± 0.1
breast-l	72.9 ± 2.3	71.7 ± 1.5	71.5 ± 0.4	73.5 ± 0.1	73.5 ± 0.1	**73.6 ± 0.1**	73.2 ± 0.1
breast-t	59.6 ± 5.7	64.7 ± 5.7	47.8 ± 0.8	**65.2 ± 0.6**	64.5 ± 0.6	62.9 ± 0.5	62.1 ± 0.4
breast-w	**94.9 ± 0.7**	93.8 ± 0.8	93.2 ± 0.2	94.0 ± 0.2	94.1 ± 0.2	93.5 ± 0.1	93.8 ± 0.1
credit-a	85.8 ± 1.0	85.2 ± 1.4	85.2 ± 0.2	85.9 ± 0.1	85.8 ± 0.1	85.9 ± 0.1	**86.0 ± 0.1**
derma	93.5 ± 1.1	94.0 ± 1.4	**95.7 ± 0.2**	94.4 ± 0.1	94.5 ± 0.1	94.4 ± 0.1	94.6 ± 0.1
ecoli	83.7 ± 1.7	81.3 ± 1.9	81.2 ± 0.3	**83.9 ± 0.1**	83.1 ± 0.2	83.2 ± 0.2	83.4 ± 0.2
glass	68.5 ± 1.8	68.4 ± 2.5	65.1 ± 0.5	71.0 ± 0.4	71.2 ± 0.5	77.1 ± 0.4	**77.2 ± 0.4**
heart-c	51.2 ± 1.6	**55.1 ± 1.4**	54.9 ± 0.5	51.9 ± 0.4	52.3 ± 0.4	51.5 ± 0.4	51.8 ± 0.3
heart-h	**66.7 ± 3.0**	62.7 ± 2.1	62.3 ± 0.4	65.9 ± 0.2	66.2 ± 0.2	65.4 ± 0.3	65.6 ± 0.2
hep	80.7 ± 2.5	77.5 ± 2.1	80.4 ± 0.7	**82.5 ± 0.3**	82.2 ± 0.3	80.6 ± 0.2	81.1 ± 0.2
horse	**84.7 ± 1.5**	83.5 ± 1.6	83.1 ± 0.3	83.8 ± 0.1	83.6 ± 0.2	83.0 ± 0.2	83.1 ± 0.2
ionos	90.2 ± 1.2	89.4 ± 1.8	88.0 ± 0.4	**90.8 ± 0.2**	90.7 ± 0.1	89.0 ± 0.4	88.9 ± 0.2
iris	93.9 ± 1.6	93.8 ± 1.3	94.9 ± 0.4	**96.2 ± 0.1**	95.9 ± 0.1	95.1 ± 0.1	94.9 ± 0.1
park	88.2 ± 1.5	86.7 ± 2.2	86.2 ± 0.5	**92.4 ± 0.1**	92.2 ± 0.1	89.2 ± 0.3	89.4 ± 0.2
s-heart	75.6 ± 3.1	78.1 ± 2.3	**79.5 ± 0.3**	77.3 ± 0.3	76.9 ± 0.3	77.3 ± 0.2	77.1 ± 0.2
soybean	85.6 ± 1.7	87.6 ± 1.9	86.4 ± 0.3	**87.4 ± 0.2**	87.3 ± 0.2	87.4 ± 0.3	87.2 ± 0.2
voting	94.5 ± 1.2	93.3 ± 1.2	93.1 ± 0.2	**94.9 ± 0.1**	94.7 ± 0.1	94.8 ± 0.1	94.8 ± 0.1
wine	93.3 ± 2.4	91.0 ± 1.5	89.2 ± 0.5	96.2 ± 0.1	**96.4 ± 0.1**	94.3 ± 0.3	94.9 ± 0.2
zoo	**89.6 ± 5.2**	87.6 ± 5.9	82.4 ± 0.7	88.2 ± 0.2	88.6 ± 0.3	88.8 ± 0.4	88.8 ± 0.4

انگیزه برای تست این تغییرات مختلف, ارزیابی هر دو استراتژی های گسسته سازی و هرس است. از نظر گسسته سازی، ما در حال ارزیابی این مورد هستیم که آیا استفاده از یک گسسته سازی باینری بهتر است یا خیر (به عنوان مثال، یک گسسته سازی که یک تقسیم دوتایی تک ایجاد می کند) و یا چند گسسته سازی فاصله مبتنی بر MDL. از نظر هرس کردن، ما در حال ارزیابی این مورد هستیم که آیا ابتدائاً, تنظیم یک درخت نماینده برای بهترین تناسب داده های آموزش (هرس دو مرحله) و یا استفاده از روشه هرس ساده شده (یک مرحله)- فقط رویه هرس مبتنی بر- خطای C4.5 بهترین کار است یا خیر.

یک مقایسه بین Ant-Tree-Miner و cACDT جالب است، زیرا Ant-Tree-Miner از راهبردی متفاوت برای ایجاد یک درخت تصمیم گیری پیروی می کند. اول، cACDT تنها درختان دودویی را ایجاد می کند (یعنی، درختان با گره های تصمیم گیری با دو یال خروجی)، در حالی که Ant-Tree-Miner به ایجاد درختان دودویی محدود نمی شود (یعنی برای گره های تصمیم گیری به نمایندگی از صفات اسمی، یک شاخه برای هر مقدار در آن دامنه از صفت وجود دارد). دوم، Ant-Tree-Miner, یک ماتریس فرمون ۳ بعدی برای متمایز نمودن چند وقوع یال ها (ارتباط بین صفات) در یک درخت تصمیم گیری را استفاده می کند. سوم، cACDT از معیار Twoing (بر پایه الگوریتم CART) به عنوان اطلاعات اکتشافی و گسسته سازی پویای صفات پیوسته استفاده می کند، در حالی که

Ant-Tree-Miner از معیار مبتنی بر آنتروپی (مبتنی بر الگوریتم C4.5) به عنوان اطلاعات اکتشافی و گسسته سازی پویای صفات پیوسته استفاده می کند.

۴,۱. راه اندازی آزمایشی

آزمایشات با استفاده از ۲۲ مجموعه اطلاعات در دسترس عموم از مخزن UCI یادگیری ماشین [۱۶] انجام شد. خلاصه ای از مجموعه های داده های مورد استفاده در آزمایش در جدول ۱ آمده است. ما اعتبار سنجی متقاطع ده برابر برای هر مجموعه داده را انجام دادیم. یک روش اعتبارسنجی متقاطع ده برابر متشکل از تقسیم مجموعه داده به ده پارتیشن و متناسب با حجم نمونه را انجام دادیم که در آن که هر پارتیشن دارای یک تعداد مشابهی از مثال ها و کلاس توزیع است. برای هر پارتیشن، الگوریتم طبقه بندی با استفاده از نه پارتیشن باقی مانده به عنوان مجموعه آموزش اجرا می شود و عملکرد آن با استفاده از پارتیشن نهان (HOLD کردن) ارزیابی می شود. برای الگوریتم تصادفی Ant-Tree-Miner و تغییرات آن، الگوریتم پانزده بار با استفاده از یک دانه مختلف به صورت تصادفی برای مقداردهی اولیه جستجو برای هر یک از پارتیشن های اعتبار متقاطع اجرا می شود. در مورد الگوریتم های قطعی C4.5 و CART، الگوریتم فقط یک بار برای هر پارتیشن از اعتبارسنجی متقاطع اجرا می شود.

پارامترهای تعریف شده-کاربر از تغییرات Ant-Tree-Miner از مقادیر معمولاً استفاده شده در نوشته ها انتخاب شدند. اگر چه ما هیچ تلاشی برای تنظیم مقادیر پارامتر برای مجموعه داده های فردی انجام نداده ایم، یک تنظیم پارامتر سیستماتیک را به منظور تعیین یک ترکیب مناسب از مقادیری که به خوبی در سراسر مجموعه ای از تنظیم مجموعه داده ها کار می کنند انجام داده ایم. ما سه مقدار مختلف را برای پارامترهای اندازه کولونی = ۵۰، ۱۰۰، ۲۰۰، هر یک با مقادیر مختلف عامل تبخیر MAX-MIN = ۰,۸۵، ۰,۹۰، ۰,۹۵-یعنی، نه ترکیب در مجموع, آزمایش نموده ایم. ما حداکثر تکرارات = ۵۰۰ را ثابت گذاشته ایم، با توجه به اینکه مشاهده کردیم که الگوریتم در کمتر از ۲۰۰ بار تکرار به طور متوسط، و حداقل تعداد نمونه در هر آستانه شاخه = ۳ همگرا می شود. تنظیم پارامتر, تفاوت های قابل توجهی بین نه ترکیب ممکن مقادیر را نشان نداد- یعنی، یافته ها در مورد تفاوت آماری معنی داری بین الگوریتم ها تقریبا در تمام نه ترکیبی از اندازه کولونی و عامل تبخیر مشابه بود. ترکیبی از اندازه کولونی = ۵۰ و

عامل تبخیر = ۰,۹۰ در آزمایشات گزارش شده در این بخش استفاده شد، زیرا به طور مختصر کمی بهتر از ترکیبات, به طور کلی, در تنظیم مجموعه داده ها عمل نمود. الگوریتم های C4.5، CART و cACDT با پارامترهای پیش فرض خود استفاده شدند.

نتایج مربوط به دقت پیش بینی کننده در جدول ۲ خلاصه شده است و نتایج مربوط به اندازه مدل طبقه بندی، که به عنوان میانگین تعداد گره های برگ در درخت تصمیم گیری کشف شده اندازه گیری شد، در جدول ۳. خلاصه شده اند. هر مقدار در آن جداول نشان دهنده ی متوسط مقدار به دست آمده با روش اعتبارسنجی متقاطع پس از خطای استاندارد (خطای متوسط ± استاندارد) برای الگوریتم مربوطه و جفت داده تنظیم شده است. جدول ۴, نتایج آزمون های آماری با توجه به آزمون غیر پارامتری فریدمن با آزمون Hommel موقت [۹،۲۰]، برای دقت پیش بینی و اندازه مدل کشف شده را نشان می دهد. برای هر الگوریتم، جدول, رتبه متوسط آن را نشان می دهد- هرقدر رتبه متوسط پایین تر باشد, عملکرد الگوریتم بهتر است- در ستون اول و مقدار-p تنظیم شده آزمون آماری, زمانی که رتبه متوسط الگوریتم با رتبه متوسط الگوریتم با بهترین رتبه (الگوریتم کنترل) با توجه به آزمون تعقیبی Hommel در مقایسه قرار می گیرد. وقتی که اختلافات معنی دار آماری بین رتبه های متوسط یک الگوریتم و الگوریتم کنترل در سطح ۵٪ ($P \leq 0.05$) مشاهده می شوند، خط در وجه پررنگ جدولبندی می شود.

۴,۲. بحث

با توجه به دقت پیش بینی (جدول ۲)، Ant-Tree-Miner به بالاترین کارایی با رتبه متوسط ۲,۵۲ در تمام مجموعه داده ها دستیابی پیدا می کند که از نظر آماری به طور قابل توجهی بهتر از رتبه های متوسط به دست آمده توسط الگوریتم های Ant-Tree-MinerMDL, CART و cACDT با توجه به آزمون فریدمن غیر پارامتری با آزمون تعقیبی Hommel می باشد. تغییرات Tree-Miner–P

MDL, یک رتبه متوسط ۳,۲۰ و ۳,۹۳، را به ترتیب به دست آورد و هیچ تفاوت آماری معنی دار بین رتبه های متوسط و رتبه متوسط Ant-Tree-Miner وجود ندارد. Ant-Tree-MinerMDL ,C4.5 وCART دارای

اجراهای مشابه، با رتبه متوسط ۴,۰۴، ۴,۱۱ و ۴,۴۹ بودند. cACDT دارای بدترین عملکرد، با رتبه متوسط ۵,۶۸. Ant-Tree-Miner نیز دقیق ترین الگوریتم در ۸ تا از ۲۲ مجموعه داده هاست و پس از آن C4.5 در ۵ مجموعه داده ، cACDT در ۳ مجموعه داده، P MDL-Ant-Tree-Miner و CART هر کدام در ۲ داده مجموعه، و هر دوی P-Ant-Tree-Miner و مورچه درخت MinerMDL در ۱ مجموعه داده وجود دارد.

جدول ۳

تعداد متوسط گره ها (خطای استاندارد +-) متوسط درختان تصمیم گیری که توسط اعتبارسنجی متقاطع ۱۰ تایی اندازه گیری می شود. ستون های نشان داده شده توسط ATM متناظر با تغییرات Ant-Tree-Miner هستند. مقدار این الگوریتم با کمترین تعداد گره های برگ برای هر مجموعه داده ها به صورت توپر نشان داده شده است

Data set	C4.5	CART	cACDT	ATM	ATM^{-P}	ATM$_{MDL}$	ATM$^{-P}_{MDL}$
auto	46.6 ± 2.3	23.3 ± 0.9	**12.4 ± 0.5**	31.1 ± 0.6	29.5 ± 0.6	27.3 ± 0.3	27.7 ± 0.3
balance	34.6 ± 1.2	**25.4 ± 5.2**	47.4 ± 0.4	34.5 ± 0.1	34.6 ± 0.0	34.5 ± 0.1	34.5 ± 0.1
blood-t	6.6 ± 0.7	**6.4 ± 0.7**	8.0 ± 0.3	14.8 ± 0.1	14.8 ± 0.1	14.9 ± 0.1	14.6 ± 0.1
breast-l	8.2 ± 2.6	**2.9 ± 0.4**	10.7 ± 0.2	10.0 ± 0.2	10.3 ± 0.2	10.5 ± 0.3	10.5 ± 0.2
breast-t	12.9 ± 0.5	**7.8 ± 0.8**	15.3 ± 0.4	12.0 ± 0.1	11.9 ± 0.1	13.6 ± 0.1	13.7 ± 0.1
breast-w	11.5 ± 0.5	**7.1 ± 0.5**	10.9 ± 0.2	9.0 ± 0.1	9.0 ± 0.1	11.1 ± 0.1	11.1 ± 0.1
credit-a	19.8 ± 2.2	**2.8 ± 0.4**	14.7 ± 0.3	29.6 ± 0.2	29.8 ± 0.2	29.8 ± 0.3	29.6 ± 0.3
derma	28.0 ± 1.3	**7.5 ± 0.2**	14.6 ± 0.2	20.7 ± 0.1	21.1 ± 0.1	20.9 ± 0.1	20.9 ± 0.1
ecoli	17.9 ± 0.9	**10.1 ± 1.2**	13.9 ± 0.3	15.9 ± 0.1	16.0 ± 0.1	16.8 ± 0.1	16.5 ± 0.1
glass	23.8 ± 0.5	14.7 ± 2.1	**5.6 ± 0.2**	20.2 ± 0.1	20.2 ± 0.1	19.5 ± 0.1	19.4 ± 0.1
heart-c	44.7 ± 1.6	8.8 ± 2.3	**8.4 ± 0.3**	35.8 ± 0.2	36.4 ± 0.3	35.8 ± 0.2	36.4 ± 0.3
heart-h	27.6 ± 0.9	**3.1 ± 0.7**	4.8 ± 0.2	21.8 ± 0.1	21.9 ± 0.1	22.1 ± 0.2	22.0 ± 0.1
hep	9.7 ± 0.6	**2.9 ± 0.8**	10.2 ± 0.3	7.6 ± 0.1	7.5 ± 0.1	8.1 ± 0.1	8.1 ± 0.1
horse	5.2 ± 0.5	**4.0 ± 0.4**	19.7 ± 0.3	10.2 ± 0.1	10.0 ± 0.1	12.7 ± 0.2	13.0 ± 0.2
ionos	13.4 ± 0.7	**8.6 ± 1.6**	8.9 ± 0.2	12.4 ± 0.1	12.2 ± 0.1	14.3 ± 0.1	14.4 ± 0.1
iris	4.5 ± 0.2	4.4 ± 0.3	4.8 ± 0.1	**4.2 ± 0.0**	4.4 ± 0.0	4.4 ± 0.0	4.3 ± 0.0
park	10.7 ± 0.4	**6.2 ± 0.7**	7.9 ± 0.2	8.0 ± 0.0	8.1 ± 0.0	8.3 ± 0.1	8.4 ± 0.1
s-heart	20.8 ± 1.6	**10.1 ± 1.5**	27.4 ± 0.3	18.1 ± 0.2	18.0 ± 0.2	18.4 ± 0.2	18.0 ± 0.2
soybean	55.9 ± 1.6	35.4 ± 2.0	**2.4 ± 0.2**	50.0 ± 0.3	50.3 ± 0.3	50.0 ± 0.3	50.3 ± 0.2
voting	5.7 ± 0.1	**4.8 ± 0.7**	10.8 ± 0.2	6.1 ± 0.0	6.1 ± 0.0	6.1 ± 0.0	6.0 ± 0.0
wine	**5.2 ± 0.1**	5.5 ± 0.3	7.5 ± 0.1	5.6 ± 0.0	5.6 ± 0.0	7.1 ± 0.0	7.1 ± 0.0
zoo	11.2 ± 0.8	**7.3 ± 0.1**	7.4 ± 0.0	10.7 ± 0.0	10.7 ± 0.1	10.5 ± 0.0	10.8 ± 0.1

از نظر اندازه مدل طبقه بندی (جدول ۳)، که به عنوان تعداد گره های برگ درخت تصمیم گیری کشف شده اندازه گیری می شود، CART, الگوریتمی است که درخت تصمیم گیری را با کمترین تعداد برگ کشف می کند و یک رتبه متوسط ۱,۳۹ در تمام مجموعه داده ها به دست می آورد. رتبه متوسط CART از نظر آماری معنی دار بهتر از رتبه های متوسط به دست آمده توسط هر شش الگوریتم دیگر با توجه به آزمون فریدمن غیر پارامتری با آزمون تعقیبی Hommel است. Ant-Tree-Miner در رتبه دوم با رتبه متوسط ۳,۶۸ قرار دارد. سوم cACDT با رتبه متوسط ۴,۰۰ قرار دارد. P-Ant-Tree-Miner دارای رتبه چهارم با متوسط ۴,۲۷ است. P MDL-Ant-Tree-Miner و مورچه درخت MinerMDL نتایج مشابه، با رتبه های متوسط ۴,۷۰ و ۴,۷۵ داشتند. C4.5, الگوریتمی است که به

طور مداوم درختان با تعداد بیشتری از برگ ها (در ۱۴ از ۲۲ مجموعه داده) را کشف می کند و با رتبه متوسط ۵,۲۰ عمل می کند.

نتایج ما نشان می دهند که CART به طور سازگار, الگوریتمی است که درختان تصمیم گیری با کمترین تعداد از گره های برگ را کشف می کند. این به احتمال زیاد با توجه به روشی است که CART, صفات اسمی را در نظر می گیرد. در حالی که C4.5 و Ant-Tree-Miner, یک شاخه را برای هر مقدار از یک صفت اسمی ایجاد می کنند (به عنوان مثال، هر شاخه به نمایندگی از یک وضعیت $x_i = v_{ij}$)، CART می تواند مقادیر مختلف را به یک شاخه گروه بندی نماید (یعنی هر شاخه می تواند یک وضعیت xi در $\{v_{i1}, v_{i2}, \ldots\}$ را نشان دهد. این به طور موثر, ابعاد صفات اسمی را با تعداد زیادی از مقادیر کاهش می دهد و گروه بندی الگوریتم به تعداد کمی از زیر مجموعه ها را میسر می سازد و در نتیجه، باعث کاهش پیچیدگی درخت و تعداد گره های برگ می شود. از سوی دیگر، دارای تاثیر منفی بسیار قابل توجه در دقت پیش بینی درختان تصمیم گیری کشف شده است، زیرا CART, بدترین رتبه دوم را از نظر دقت پیش بینی کسب می کند.

Algorithm	Average rank	Adjusted p_{Homm}
(i) Predictive Accuracy		
Ant-Tree-Miner (control)	2.52	–
Ant-Tree-Miner^{-P}	3.20	0.2952
Ant-Tree-Miner$^{-P}_{MDL}$	3.93	0.0610
C4.5	**4.04**	**0.0458**
Ant-Tree-Miner$_{MDL}$	**4.11**	**0.0437**
CART	**4.49**	**0.0120**
*c*ACDT	**5.68**	**7.4E-6**
(ii) Model Size		
CART (control)	1.39	–
Ant-Tree-Miner	**3.68**	**4.2E-4**
*c*ACDT	**4.00**	**1.2E-4**
Ant-Tree-Miner^{-P}	**4.27**	**2.8E-5**
Ant-Tree-Miner$^{-P}_{MDL}$	**4.70**	**1.4E-6**
Ant-Tree-Miner$_{MDL}$	**4.75**	**9.7E-7**
C4.5	**5.20**	**2.7E-8**

جدول ۴ -نتایج آزمون آماری با توجه به آزمون فریدمن غیر پارامتری با آزمون تعقیبی Holm برای $\alpha = 0.05$.

تفاوت آماری معنی دار بین رتبه های متوسط یک الگوریتم و الگوریتم کنترل به صورت ضخیم نشان داده می شوند

به طور کلی، نتایج به دست آمده توسط الگوریتم Ant-Tree-Miner به وضوح مثبت هستند. این به بهترین رتبه از نظر دقت پیش بینی کننده دست می یابد و با تفاوت های آماری معنی دار، از الگوریتم های معروف C4.5 و CART و همچنین از الگوریتم cACDT ACO بهتر عمل می کند. در حالی که Ant-Tree-Miner, الگوریتمی نیست که درختان تصمیم گیری با کمترین تعداد گره برگ را کشف می کند و به خاطر CART با یک تفاوت آماری معنی دار, عملکرد بهتری دارد، Ant-Tree-Miner به بهترین رتبه از نظر اندازه مدل طبقه بندی دستیابی پیدا می کند؛ هر چند CART به بهترین رتبه بندی در اندازه مدل طبقه بندی دستیابی پیدا می کند، به بدترین دوم رتبه در دقت پیش بینی کننده دستیابی پیدا می کند. بنابراین، Ant-Tree-Miner یک موازنه بهتر بین هر دو دقت و اندازه مدل را به طور کلی، نسبت به C4.5، CART - که دو الگوریتم استنتاجی درخت تصمیم گیری معمولی بسیار محبوب هستند و نسبت به الگوریتم cACDT مبتنی بر ACO- ارائه می دهد.

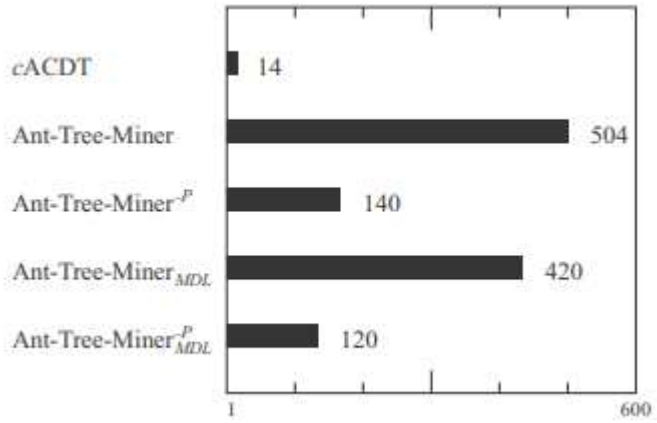
با مقایسه عملکرد تغییرات Ant-Tree-Miner, می توانیم نتیجه گیری های زیر را در مورد استراتژی های گسسته سازی و هرس بگیریم. گسسته سازی باینری (تک فاصله) بهترین عملکرد را از نظر دقت و اندازه مدل طبقه بندی پیش بینی شده دارد، با توجه به اینکه Ant-Tree-Miner و P-Ant-Tree-Miner, رتبه های متوسط بهتر را نسبت به تغییرات متناظر Ant-Tree-MinerMDL وP–Ant-Tree-Miner MDL با استفاده از گسسته سازی (چند فاصله) ارائه می دهد. تفاوت بین مبتنی بر MDL-. تفاوت بین استراتژی های هرس به اندازه این تفاوت در راهبردهای گسسته روشن نیست. در حالی که استفاده از هرس کننده دو مرحله ای به Ant-Tree-Miner با گسسته سازی باینری برای رسیدن به یک دقت پیش بینی بالاتر و تعداد کمتری از برگ در درختان تصمیم گیری کشف شده نسبت به تغییر P-Ant-Tree-Miner کمک می کند، همان اثر در تغییرات گسسته مبتنی بر MDL-مشاهده نشد. P MDL-Ant-Tree-Miner-که از هرس کننده تک مرحله ای استفاده می کند (روش هرس مبتنی بر خطای C4.5)- یک رتبه متوسط بهتر را از نظر دقت پیش بینی و اندازه مدل طبقه بندی به دست آورد، اگر چه تفاوت ها نسبتا کوچک هستند. این به احتمال زیاد با توجه به تفاوت در ساختار درختان در هنگام استفاده از گسسته سازی بازه متعدد است که در آن در این مورد هر دو استراتژی هرس دارای یک اثر مشابه هستند.

با توجه به متوسط زمان محاسبه برای استنتاج یک درخت تصمیم گیری، که در شکل ۷ نشان داده شده است، هیچ جای تعجب نیست که تغییرات Ant-Tree-Miner با استفاده از رویه هرس دو مرحله ای نسبت به الگوریتم های استفاده کننده از هرس مبتنی بر خطای C4.5, نیاز به زمان بیشتری دارند. در حالی که الگوریتم Ant-Tree-Miner بالاترین دقت پیش بینی کلی را به دست آورد، این یک ضریب ۴,۲ کندتر از P -Ant-Tree-Miner MDL است (سریع ترین تنوع P-Ant-Tree-Miner (Ant-Tree-Miner یک موازنه بین دقت، اندازه مدل و زمان محاسباتی را ارائه می دهد: در حالی که دقت پیش بینی آن کمی پایین تر از تغییر Ant-Tree-Miner است، درختان تصمیم گیری ساده تر نیازمند به زمان محاسباتی کمتر را کشف می کند. گسسته سازی مبتنی بر MDL- به کاهش زمان محاسباتی به ضرر دقت پیش بینی پایین تر و درختان تصمیم گیری با برگ های بیشتر کمک می کند.

جدول ۵

دقت پیش بینی کننده متوسط (خطای میانگین _+ استاندارد), برحسب ٪, اندازه گیری شده توسط اعتبار-مقطعی ده تایی. ستون های نشان داده شده توسط ATM با تغییرات ANT-Tree-Miner متناظر هستند. مقدار دقیق ترین الگوریتم برای هر مجموعه داده به صورت توپر نشان داده شده است.

	Predictive accuracy		Model size	
Data set	Ant-Tree-Miner	*c*Ant-Miner$_{PB}$	Ant-Tree-Miner	*c*Ant-Miner$_1$
auto	**77.2 ± 0.4**	65.5 ± 0.9	31.1 ± 0.6	**14.3 ± 0.1**
balance	62.9 ± 0.1	**76.8 ± 0.2**	34.5 ± 0.0	**13.6 ± 0.0**
blood-t	**72.4 ± 0.1**	72.3 ± 0.1	14.8 ± 0.1	**9.8 ± 0.0**
breast-l	**73.5 ± 0.1**	72.3 ± 0.3	**10.0 ± 0.2**	10.4 ± 0.1
breast-t	65.2 ± 0.6	**67.1 ± 0.5**	12.0 ± 0.1	**6.7 ± 0.0**
breast-w	94.0 ± 0.2	**94.3 ± 0.2**	9.0 ± 0.1	**8.3 ± 0.1**
credit-a	**85.9 ± 0.1**	85.7 ± 0.1	29.6 ± 0.2	**12.3 ± 0.1**
derma	**94.4 ± 0.1**	92.5 ± 0.3	20.7 ± 0.1	**19.2 ± 0.1**
ecoli	**83.9 ± 0.1**	79.9 ± 0.2	15.9 ± 0.1	**9.1 ± 0.0**
glass	71.0 ± 0.4	**73.9 ± 0.2**	20.2 ± 0.1	**9.4 ± 0.1**
heart-c	51.9 ± 0.4	**55.5 ± 0.4**	35.8 ± 0.2	**12.8 ± 0.1**
heart-h	**65.9 ± 0.2**	64.7 ± 0.3	21.8 ± 0.1	**11.0 ± 0.1**
hep	**82.5 ± 0.3**	80.1 ± 0.5	**7.5 ± 0.1**	**7.6 ± 0.1**
horse	**83.8 ± 0.1**	83.5 ± 0.3	**10.2 ± 0.1**	10.8 ± 0.2
ionos	**90.8 ± 0.2**	89.6 ± 0.3	12.4 ± 0.1	**9.2 ± 0.1**
iris	**96.2 ± 0.1**	93.2 ± 0.2	**4.2 ± 0.0**	4.8 ± 0.0
park	**92.4 ± 0.1**	87.0 ± 0.5	8.0 ± 0.0	**7.1 ± 0.1**
s-heart	**77.3 ± 0.3**	77.0 ± 0.4	18.1 ± 0.2	**8.6 ± 0.1**
soybean	**87.4 ± 0.2**	80.0 ± 0.3	50.0 ± 0.3	**22.3 ± 0.1**
voting	**94.9 ± 0.1**	93.3 ± 0.2	**6.1 ± 0.0**	**6.4 ± 0.1**
wine	**96.2 ± 0.1**	93.6 ± 0.3	5.6 ± 0.0	**5.4 ± 0.1**
zoo	**88.2 ± 0.2**	78.6 ± 0.9	10.7 ± 0.0	**6.0 ± 0.0**



شکل. ۷. میانگین زمان محاسبه (روی همه مجموعه داده ها) برحسب ثانیه برای تکمیل اعتبارسنجی متقاطع برای هر یک از الگوریتم های مبتنی بر ACO-بر روی یک کامپیوتر با پردازنده ۳,۳۳ گیگاهرتز Xeon اینتل. C4.5 و الگوریتم قطعی CART با متوسط ۱ ثانیه برای تکمیل اعتبارسنجی متقاطع.

در مقایسه با زمان محاسباتی برای ایجاد یک درخت تصمیم گیری از C4.5 و CART، تغییرات Ant-Tree-Miner, یک ضریب ۱۲۰-۵۰۴ کندتر هستند. این انتظار می رود، با توجه به اینکه هر دو C4.5 و CART درخت تصمیم گیری کاندیدای واحد را با استفاده از یک رویه قطعی ایجاد و هرس می نمایند، در حالی که در تغییرات Ant-Tree-Miner, چند درخت تصمیم گیری نماینده ایجاد می شوند و قبل از پیدا کردن بهترین درخت تصمیم گیری هرس می شوند. با مقایسه زمان گرفته شده توسط cACDT، تغییرات Ant-Tree-Miner یک ضریب از ۸,۵-۳۶ آهسته تر است، در حالی که دقت پیش بینی cACDT از نظر معنی دار آماری کمتر از دقت Ant-Tree-Miner است. لازم به ذکر است که در بسیاری از کاربردهای داده کاوی (به عنوان مثال، تشخیص پزشکی، بیوانفورماتیک و امتیازدهی اعتباری) زمان محاسبات گرفته شده توسط الگوریتم برای استنتاج یک مدل طبقه بندی دارای اهمیت نسبتا جزئیست، چرا که آنها کاربردهای آفلاین را نشان می دهد و زمان صرف شده برای جمع آوری و آماده سازی داده ها معمولاً بسیار بیشتر از زمان مورد نیاز برای اجرای طبقه بندی الگوریتم است, اگر چه در کاربردهایی که نیاز به تنظیم پارامتر دارند، تعداد تنظیمات پارامتر ارزیابی شده می تواند با زمان محاسباتی الگوریتم محدود شود. علاوه بر این، الگوریتم های ACO به راحتی می توانند مواسزی سازی شوند, زیرا هر مورچه, یک درخت

تصمیم گیری نماینده مستقل از تمام مورچه های دیگر را می سازد و ارزیابی می کند. بنابراین، یک سرعت بالای بزرگ را می توان با اجرای یک نسخه موازی Ant-Tree-Miner بر روی یک خوشه کامپیوتر و یا یک سیستم محاسبات موازی دیگر در کاربردهایی به دست آورد که در آن زمان پردازش Ant-Tree-Miner یک موضوع مهم است.

۴,۳. مقایسه الگوریتم Ant-Tree-Miner و نمی تونم-MinerPB

در این بخش ما, عملکرد Ant-Tree-Miner در برابر الگوریتم استنتاج قاعده MinerPB-cANT مبتنی بر-ACO را مقایسه نمودیم [27]. چنین مقایسه ای به طور ویژه جالب است چرا که هر دو الگوریتم از یک رویه مبتنی بر ACO برای ایجاد یک مدل طبقه بندی قابل درک استفاده می کنند، در حالی که از نمایش های راه حل ها و استراتژی های جستجوی متفاوت استفاده می کنند. علاوه بر این، درختان تصمیم گیری را می توان به مجموعه ای از قوانین طبقه بندی تبدیل نمود (به عنوان مثال، همانطور که در [۳۱] مورد بحث قرار گرفته است) و بالعکس (به عنوان مثال، همانطور که در [۱] مورد بحث قرار گرفته است). جدول ۵, نتایج Ant-Tree-Miner و cANT-MinerPB ارائه شده را از نظر دقت پیش بینی و اندازه مدل طبقه بندی، با استفاده از همان روش اعتبارسنجی متقاطع ده برابر در بند ۴,۱ نشان می دهد. در مورد MinerPB-cANT، اندازه, متناظر با تعداد کل قوانین در لیست های کشف شده است، در حالی که اندازه در مورد Ant-Tree-Miner متناظر با تعداد گره های برگ در درختان کشف شده است- با توجه به اینکه مسیر از یک گره برگ به گره ریشه درخت را می توان به عنوان یک قاعده در نظر گرفت.

با توجه به آزمون آزمون رتبه- نشان داده شده ویلکاکسون [۹] در سطح $\alpha = 0.05$, عملکرد Ant-Tree-Miner از نظر دقت پیش بینی آماری به طور قابل توجهی بهتر از عملکرد MinerPB-cANT بود (مقدار p = 0.0461). از سوی دیگر، با توجه به آزمون آزمون رتبه- نشان داده شده ویلکاکسون [۹] در سطح $\alpha = 0.05$, cANT-MinerPB از نظر آماری معنادار بهتر از Ant-Tree-Miner, از نظر اندازه مدل طبقه بندی است (مقدار p = 0.0001). این نتایج نشان می دهند که درختان تصمیم گیری ایجاد شده توسط Ant-Tree-Miner دقیق تر از

لیست های قاعده ایجاد شده توسط MinerPB-cANT هستند که به اندازه بزرگتر منجر می شود. همانطور که در [۳۱] مورد بحث قرار گرفته است، کاهش پیچیدگی نمایش درخت تصمیم گیری با تبدیل یک درخت به مجموعه ای از قوانین طبقه بندی، که در آن هر قانون را می توان به صورت جداگانه ساده سازی نمود و در عین حال, همان سطح از دقت پیش بینی را حفظ کرد, ممکن است . استفاده از چنین روشی, یک جهت گیری جالب برای تحقیقات آینده است.

۵. نتیجه گیری ها

در این مقاله, یک الگوریتم جدید کلونی مورچه، به نام Ant-Tree-Miner برای استنتاج درختان تصمیم گیری در زمینه وظیفه طبقه بندی در داده کاوی پیشنهاد شده است. هنگامی که این الگوریتم به عنوان یک مدل طبقه بندی استفاده می شود، درختان تصمیم گیری دارای مزیت نمایش یک مدل قابل فهم هستند و به راحتی در یک فرم گرافیکی و یا به صورت مجموعه ای از قوانین طبقه بندی بیان می شوند. الگوریتم Ant-Tree-Miner از یک رویکرد بالا به پایین توسط انتخاب احتمالاتی صفات به عنوان گره های تصمیم گیری مبتنی بر میزان فرومون و اطلاعات اکتشافی پیروی می کند. نمایش ماتریس فرمون, سطح در درخت تصمیم گیری را در نظر می گیرد که یک شاخه خاص (صفت، مقدار)- جفت- برای تمایز بین چندین رخداد از یک شاخه به نظر می رسد. چهار تغییرات الگوریتم Ant-Tree-Miner در برابر دو الگوریتم استنتاجی درخت تصمیم گیری شناخته شده، یعنی C4.5 (الگوریتم J48 WEKA) و CART (الگوریتم SimpleCART WEKA)، و الگوریتم درخت تصمیم گیری cACDT مبتنی بر ACO- در ۲۲ مجموعه داده در دسترس عموم از نظر دقت پیش بینی و تعداد گره های برگ درختان تصمیم گیری کشف شده مقایسه شدند.

نتایج نشان داده اند که الگوریتم Ant-Tree-Miner بهتر از C4.5، CART و cACDT از نظر دقت پیش بینی, عمل می کند و به یک رتبه متوسط آماری معناداری بهتر در تمام مجموعه داده ها دستیابی پیدا می کند. همچنین یک موازنه خوب بین صحت و اندازه پیش بینی کننده مدل طبقه بندی (تعداد گره های برگ در درختان کشف شده) نیز ارائه می دهد: Ant-Tree-Miner, بهترین رتبه را از نظر دقت پیش بینی و بهترین رتبه دوم را در تعداد

گره های برگ به دست آورد؛ C4.5 بهترین رتبه چهارم در دقت پیش بینی و رتبه ششم در تعداد گره های برگ را به دست آورد. CART رتبه ششم در دقت پیش بینی کننده و رتبه اول در تعداد گره های برگ را به دست آورد. cACDT رتبه هفتم را در دقت پیش بینی و رتبه سوم را در تعداد گره های برگ به دست آورد. ارزیابی تغییرات مختلف الگوریتم Ant-Tree-Miner نشان داده اند که گسسته سازی باینری (تک فاصله) در ترکیب با هرس کننده دو مرحله ای (دقت طبقه بندی و مبتنی بر خطای C4.5), بهترین عملکرد را از نظر دقت پیش بینی کننده و اندازه مدل طبقه بندی فراهم می کند.

چندین مسیر برای تحقیقات آینده وجود دارد. بررسی معیارهای اطلاعات مختلف اکتشافی، و همچنین رویه های گسسته مختلف که بتوانند انتخاب صفت در گره های تصمیم گیری و در نتیجه ساختار درختان تصمیم گیری را بهبود بخشند, جالب خواهد بود. استفاده از رئوس به نمایندگی از گره های برگ در گراف ساخت که انتخاب مورچه ها را در هنگام اضافه کردن یک گره برگ میسر می سازد, می تواند به طور بالقوه از نیاز به یک روش هرس اجتناب نماید. این نه تنها موجب ساده شدن الگوریتم می شود، بلکه یک راه حل ظریف برای مشکل برازش بیش از حد داده های آموزشی را در طول ساخت یک درخت تصمیم گیری نماینده فراهم می کند.

References

[1] A. Abdelhalim, I. Traore, B. Sayed, Rbdt-1: a new rule-based decision tree generation technique, in: Proceedings of the 2009 International Symposium on Rule Interchange and Applications, 2009, pp. 108–121.

[2] R.C. Barros, M.P. Basgalupp, A.C.P.L.F. de Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Transactions Systems, Man, and Cybernetics, Part C: Applications and Reviews (2011).

[3] H. Blockeel, L. Schietgat, J. Struyf, S. Džeroski, A. Clare, Decision trees for hierarchical multilabel classification: a case study in functional genomics, in: Proceedings of the ECML PKDD, 2006, pp. 18–29.

[4] U. Boryczka, J. Kozak, Ant colony decision trees – a new method for constructing decision trees based on ant colony optimization, in: Proceedings of the ICCCI, 2010, pp. 373–382.

[5] U. Boryczka, J. Kozak, An adaptive discretization in the ACDT algorithm for continuous attributes, in: Proceedings of the ICCCI, 2011, pp. 475–484.

[6] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, Chapman and Hall, 1984.

[7] T.M. Cover, J.A. Thomas, Elements of Information Theory, 2nd edition, Wiley-Blackwell, 2006.

[8] M. Daud, D. Corne, Human readable rule induction in medical data mining: a survey of existing algorithms, in: Proceedings of the WSEAS European Computing Conference, 2007.

[9] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Machine Learning Research 7 (2006) 1–30.
[10] V. Dhar, D. Chou, F. Provost, Discovering interesting patterns for investment decision making with GLOWER - a genetic learner overlaid with entropy reduction, Data Mining and Knowledge Discovery 4 (4) (2000) 251–280.
[11] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, 1999, pp. 11–32.
[12] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artificial Life 5 (2) (1999) 137–172.
[13] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, 2004.
[14] U.M. Fayyad, K.B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Thirteenth International Joint Conference on Artifical Inteligence, Morgan Kaufmann, 1993, pp. 1022–1027.
[15] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, From data mining to knowledge discovery: an overview, in: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy (Eds.), Advances in Knowledge Discovery & Data Mining, MIT Press, 1996, pp. 1–34.
[16] A. Frank, A. Asuncion, UCI machine learning repository, 2010.
[17] A.A. Freitas, Understanding the crucial role of attribute interaction in data mining, Artificial Intelligence Review 16 (3) (2001) 177–199.
[18] A.A. Freitas, R.S. Parpinelli, H.S. Lopes, Ant colony algorithms for data classification, in: Encyclopedia of Information Science and Technology, volume 1, 2 edition, 2008, pp. 154–159.
[19] A.A. Freitas, D.C. Wieser, R. Apweiler, On the importance of comprehensible classification models for protein function prediction, IEEE/ACM Transactions on Computational Biology and Bioinformatics 7 (1) (2010) 172–182.
[20] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Machine Learning Research 9 (2008) 2677–2694.
[21] S. Izrailev, D. Agrafiotis, A novel method for building regression tree models for QSAR based on artificial ant colony systems, Journal of Chemical Information and Computer Sciences 41 (2001) 176–180.
[22] D. Martens, B. Baesens, T. Fawcett, Editorial survey: swarm intelligence for data mining, Machine Learning 82 (1) (2011) 1–42.
[23] J. Mingers, An empirical comparison of selection measures for decision-tree induction, Machine Learning 3 (1989) 319–342.
[24] T. Mitchell, Machine Learning, 1st edition, McGraw Hill, 1997.
[25] F.E.B. Otero, A.A. Freitas, C.G. Johnson, cAnt-Miner: an ant colony classification algorithm to cope with continuous attributes, in: Proceedings of the 6th International Conference on Swarm Intelligence (ANTS 2008), LNCS 5217, Springer-Verlag, 2008, pp. 48–59.
[26] F.E.B. Otero, A.A. Freitas, C.G. Johnson, Handling continuous attributes in ant colony classification algorithms, in: Proceedings of the 2009 IEEE Symposium on Computational Intelligence in Data Mining (CIDM 2009), IEEE, 2009, pp. 225–231.
[27] F.E.B. Otero, A.A. Freitas, C.G. Johnson, A new sequential covering strategy for inducing classification rules with ant colony algorithms, IEEE Transactions on Evolutionary Computation (2012).
[28] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with an ant colony optimization algorithm, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 321–332.
[29] G. Piatetsky-Shapiro, W. Frawley, Knowledge Discovery in Databases, AAAI Press, 1991.
[30] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[31] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[32] J.R. Quinlan, Improved use of continuous attributes in C4.5, Artificial Intelligence Research 7 (1996) 77–90.
[33] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers - a survey, IEEE Transactions on Systems, Man and Cybernetics 35 (4) (2005) 476–487.
[34] T. Stützle, H.H. Hoos, Improvements on the ant system: introducing $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, in: Proceedings of the International Conference Artificial Neural Networks and Genetic Algorithms, 1997.
[35] T. Stützle, H.H. Hoos, $\mathcal{MAX}$-$\mathcal{MIN}$ ant system, Future Generation Computer Systems 16 (8) (2000) 889–914.
[36] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, H. Blockeel, Decision trees for hierarchical multi-label classification, Machine Learning 73 (2) (2008) 185–214.
[37] H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition, Morgan Kaufmann, 2005.