یک روش افزایشی[۱] جدید برای کشف مجموعه وابستگی های تابعی تقریبی کمینه با استفاده از عملگرهای بیتی

س. م. فخراحمد* ، م. ذوالقدری جهرمی ، م. ه. صدرالدینی

چکیده

کشف و استخراج رابطه های بین مقادیر خصیصه ها در بانک های اطلاعات رابطه ای که از آنها با عنوان وابستگی های تابعی[۲] یاد می‌کنیم، یک مقوله مهم در داده کاوی[۳] و اخذ دانش[۴] است. تا کنون روش های بسیاری برای کشف وابستگی های تابعی کلاسیک و تقریبی ارائه شده اند. اما حتی بهینه ترین این روش ها هم کارایی قابل قبولی برای مجموعه داده های بسیار حجیم ندارند. به علاوه، اغلب روش های پیشنهاد شده، داده های بانک اطلاعات را ثابت فرض می‌کنند و در نتیجه، با افزوده شدن داده های جدید به بانک اطلاعاتی، پیمایش مجدد کل داده ها جهت کشف وابستگی های جدید لازم می‌شود. در این مقاله، ما یک روش افزایشی جدید به نام AD-Miner برای کشف وابستگی های تابعی تقریبی ارائه می‌دهیم. بخش عمده کارما بر پایه عملگرهای منطقی است. در این روش، به هنگام اضافه شدن تعدادی تاپل به یک رابطه، نیازی به پیمایش کل داده ها نمی‌باشد و کافی است داده های افزوده شده پیمایش شوند. نتایج آزمایش های ما بر روی داده های واقعی و ساختگی نشان می‌دهند که الگوریتم ما از کاراترین الگوریتم های افزایشی و غیرافزایشی موجود در زمینه کشف وابستگی های تابعی کاراتر است. مزیت دیگر روش ما نسبت به سایر روش ها نشان دادن تاپل هایی است که از یک وابستگی تبعیت نمی‌کنند. این ویژگی مهم می‌تواند جهت کشف داده های ناسازگار در یک مجموعه داده مورد استفاده قرار گیرد.

کلمات کلیدی:

وابستگی های تقریبی[۵]، داده کاوی، بانک اطلاعات رابطه ای[۶]، کاوش اطلاعات افزایشی[۷]

A new incremental method for discovery of minimal approximate dependencies using logical operations

S.M. Fakhr Ahmad, M. Zolghadri Jahromi, M.H. Sadreddini
Department of Computer Science & Engineering, School of Engineering,
Shiraz University, Shiraz, Iran

Abstract

Discovery of possible relations between attribute values in a relational database (i.e., functional dependencies), is an important issue in the field of data mining and knowledge discovery. Many search techniques have been proposed to discover classical and extended functional dependencies; but even the most efficient solutions don't have an acceptable performance in the case of large relation instances. In addition, most of the proposed algorithms assume that the database is static and thus database updates require re-scanning of the entire data repeatedly. In this work, we propose a new incremental method, *AD-Miner*, to discover Approximate Dependencies (ADs). The main part of our work is based on logical operations. The method avoids re-scans of database when a set of tuples is added to the relation. Our experimental results indicate that this method is more efficient than the existing incremental and non-incremental algorithms in this field. In addition, our method has the extra advantage of marking the index of the tuples that violate a dependency. This feature can be used to find the exceptional cases that are inconsistent with the rest of the data.

Keywords

Approximate dependencies, Data mining, Relational database, Incremental knowledge discovery

* دانشگاه شیراز، دانشکده مهندسی، بخش مهندسی و علوم کامپیوتر، mfakhrahmad@cse.shirazu.ac.ir

۱- مقدمه

وابستگی های تابعی کلاسیک در فرایند طراحی شمای بانک اطلاعات رابطه ای، جهت نرمال سازی رابطه ها و به منظور جلوگیری از افزونگی داده ها و ناهنجاری های بلیدی[۸] کاربرد دارند. این رده از وابستگی ها، هیچ انعطافی در برابر استثناها و نویز احتمالی داده ها ندارند. اما وابستگی های تقریبی، دسته دیگری از وابستگی ها هستند که لزوماً برای کل مجموعه داده ها برقرار نیستند و در نتیجه در برابر نویز و استثناء انعطاف پذیرند.

کشف وابستگی ها در بانک اطلاعات یک مقوله مهم در زمینه داده کاوی است و توجه بسیاری از محققان را در سال های اخیر به خود جلب کرده است. برقراری وابستگی های تابعی کامل در یک رابطه، غالباً امری بعید است. دلیل عمده این امر استثناهایی است که ممکن است در مجموعه داده ها موجود باشند. این استثناها با رد کردن اکثر وابستگی ها می‌توانند باعث عدم کشف یکسری اطلاعات سودمند شوند. این مشکل منجر به توسعه مفهوم وابستگی های تابعی به نوع جدیدی از وابستگی ها به نام وابستگی های تقریبی شد [1,2,3,4]. یک کاربرد مهم وابستگی های تقریبی در سیستم های پردازش پرس و جو های نادقیق بر روی بانک اطلاعاتی است که نتایج مرتبط را با توجه به مجموعه وابستگی های تقریبی و بدون نیاز به بازخورد کاربر به او بر می گردانند [5].

وابستگی های تابعی، به عنوان ارتباط بین مقادیر خصیصه ها در یک اسکیمای رابطه ای (R) تعریف می‌شوند و قالب کلی آنها بصورت $X \rightarrow A$ می‌باشد که در این رابطه A یک خصیصه از R و X زیر مجموعه ای از خصیصه های موجود در R می‌باشد. X و A به ترتیب LHS[۹] و RHS[۱۰] وابستگی نامیده می‌شود. وابستگی $X \rightarrow A$ برای یک نمونه رابطه r دارای ساختار R ، در صورتی معتبر است که به ازاء هر دو تاپل دلخواه t و u که متعلق به r هستند، داشته باشیم:

$$(t[x_i] = u[x_i] \ , \ \forall x_i \in x) \Rightarrow t[A] = u[A] \qquad (۱)$$

یک وابستگی کمینه[۱۱] نامیده می‌شود اگر و تنها اگر RHS آن به هیچ زیر مجموعه ای از LHS وابسته نباشد. به عنوان مثال اگر وابستگی ای مانند $BC \rightarrow D$ برقرار باشد، دیگر وابستگی $ABC \rightarrow D$ کمینه نیست.

پیدا کردن تمام وابستگی های موجود در یک رابطه نه ضروری و نه مفید است، چرا که بسیاری از وابستگی ها زائد هستند. در نتیجه هدف بسیاری از الگوریتم های ارائه شده یافتن مجموعه وابستگی های کمینه است و خروجی این الگوریتم ها غالباً مجموعه های کوچکی از وابستگی های نسبتاً کوتاه و مفید است. مشکل عمده ی روش های موجود کارایی نسبتاً ضعیف آنها بر روی مجموعه داده های حجیم (شامل تعداد زیادی تاپل) می‌باشد. اما افزایش داده ها تاثیر زیادی بر کند شدن الگوریتم پیشنهادی این مقاله ندارد . ساختار مقاله در ادامه به شرح زیر است: در بخش ۲، روش های موجود برای کشف وابستگی ها به اختصار معرفی می‌شوند. در بخش ۳، به معرفی روش خود و الگوریتم های اصلی آن می‌پردازیم. در بخش ۴، پیچیدگی محاسباتی روش خود را در مقایسه با روش های FastFDs [6]، روش *مقایسه* دو به دو [7] و روش *پارتیشن بندی* [8] مورد تجزیه و تحلیل قرار می-دهیم. نتایج آزمایش های انجام شده بر روی مجموعه داده های واقعی و ساختگی در بخش ۵ قابل ملاحظه است. در پایان نیز نتیجه گیری کلی از مقاله در بخش ۶ ارائه می‌شود.

۲- تحقیقات مرتبط

همان طور که قبلاً ذکر شد، مفهوم وابستگی تابعی مفهوم مهمی است که در طراحی رابطه های نرمال و به دور از آنومالی بکار می‌رود. مساله کشف و استخراج وابستگی ها یک زمینه تحقیقاتی است که جدیداً مورد توجه قرار گرفته است. روش های کشف وابستگی عمدتاً یک روال جستجو را جهت یافتن وابستگی های کاندید بکار می‌گیرند و صحت هر وابستگی توسط روال دیگری کنترل می‌شود تا وابستگی های با درجه اعتبار مورد نظر انتخاب شوند. به طور کلی دو روش برای مساله جستجو وجود دارد که عبارتند از روش بالا به پایین[۱۲] و روش پایین به بالا[۱۳]. روش بالا به پایین کارش را با یک مجموعه تهی آغاز می‌کند و وابستگی هایی را که کشف می‌کند به این مجموعه اضافه می‌کند. اما روش پایین به بالا تمام وابستگی های ممکن را معتبر فرض می‌کند و سپس هر مورد را که توسط تاپل های رابطه نقض شود از مجموعه وابستگی ها حذف می‌کند.

برای چک کردن صحت یک وابستگی، تاکنون روش های مختلفی از جمله روش های مبتنی بر درخت تصمیم گیری[۱۴]، پارتیشن بندی رابطه ها و مقایسه دو به دو پیشنهاد شده اند [7-11]. روش های نامبرده عمدتا برای داده های شامل مقادیر گسسته کاربرد دارند. برای کشف وابستگی ها، یک الگوریتم کارا و غیر افزایشی بر پایه ی جستجوی بالا به پایین در مرجع [9] ارائه شده است. دو الگوریتم افزایشی هم در [7] و [8] پیشنهاد شده اند که یکی بر پایه مقایسه دو به دو بین تاپل ها و دیگری بر اساس پارتیشن بندی رابطه ها طبق مقادیر موجود در دامنه خصیصه ها عمل می‌کند. در نسخه غیر افزایشی الگوریتم مقایسه دو به دو، تاپل ها دو به دو با هم مقایسه می-شوند اما در نسخه افزایشی آن، بین هر تاپلی که به رابطه اضافه می-شود با تمام تاپل های موجود مقایسه انجام می‌شود.

یک روش کارا اما غیر افزایشی به نام Dep-Miner [12] نیز ارائه شده که کاربرد آن در کشف وابستگی تابعی کامل است. این الگوریتم دو بخش اصلی تشکیل شده است. در فاز نخست به ازاء هر دو تاپل موجود در رابطه، مجموعه ای به نام مجموعه *تفاوت ها*[۱۵] بوجود می‌آورد که اعضای این مجموعه خصیصه هایی از رابطه است که دو تاپل در آن خصیصه ها با هم متفاوتند. در ادامه، مجموعه تفاوت

مربوط به کل رابطه از اجتماع تمام مجموعه تفاوت ها بدست می‌آید. این مرحله زمانگیرترین بخش الگوریتم Dep-Miner است که زمان مورد نیاز آن برای رابطه های حجیم بسیار زیاد می‌باشد. در فاز دوم الگوریتم، یک روش جستجوی سطحی[۱۶] جهت کشف کلیه وابستگی های کمینه بکار می‌رود.

یک روش مشابه به Dep-Miner نیز به نام FastFDs [6] وجود دارد که تنها تفاوت آنها در فاز دوم است که FatsFDs از یک جستجوی اول عمق[۱۷] بهره می‌گیرد و به دلیل همین استراتژی متفاوت جستجو کارایی‌اش کمی نسبت به Dep-Miner بهتر است و در حقیقت می‌توان آن را نسخه بهبود یافته Dep-Miner دانست. در مرجع [6]، FastFDs با الگوریتم های دیگری از جمله الگوریتمی به نام Tane [13] مقایسه شده و نتایج بدست آمده نشاندهنده برتری قابل توجه FastFDs بر سایر روش ها می‌باشد. پیچیدگی الگوریتم های بالا در مقایسه با الگوریتم پیشنهادی ما در بخش ۴ مورد تحلیل قرار خواهد گرفت.

۳- الگوریتم پیشنهادی (AD-Miner)

در این بخش به توصیف الگوریتم AD-Miner که یک الگوریتم افزایشی جدید برای کشف وابستگی های تقریبی در بانک اطلاعاتی رابطه ای است، می‌پردازیم. این الگوریتم ویژه ی خصیصه هایی با مقادیر گسسته طراحی شده است و با بکارگیری یکسری عملیات منطقی بر روی رشته های بیتی، مجموعه وابستگی های کمینه که دارای درجه صحت مورد نظر هستند را بدست می‌آورد. در بخش ۳-۱، چند تعریف پایه و در بخش های ۳-۲ و ۳-۳ روش افزایشی خود را بطور کامل شرح می‌دهیم.

۳-۱- تعاریف

تعریف ۱- رشته دودویی عضویت (Membership Binary String) برای خصیصه A (از شمای رابطه ای R) و مقدار گسسته ی λ، در یک رابطه r (از شمای R) (که با MBS(A, λ) نمایش داده می‌شود) ، یک رشته بیتی است که طول آن برابر تعداد تاپل های r است. هر بیت در این رشته متناظر با یک تاپل از رابطه است و مقدار آن یک است در صورتی که خصیصه A از آن تاپل مقدار λ داشته باشد و در غیر اینصورت صفر است.

مثال ۱: شمای رابطه ای R که شامل خصیصه های A، B، C و D است را در نظر بگیرید. رابطه r در شکل (۱) رابطه ای متعلق به شمای R است. با استفاده از تعریف ۱، رشته های بیتی MBC(C, 'L')، MBC(C, 'H') و MBC(C, 'M') بصورت زیر خواهند بود:

MBS(C,'*H*)= 001000001000
MBS(C,'*L*) = 000110110110
MBS(C,'*M*) = 110001000001

A	*B*	*C*	*D*
H	*L*	*M*	*L*
H	*L*	*M*	*L*
H	*L*	*H*	*H*
H	*H*	*L*	*L*
H	*H*	*L*	*L*
H	*M*	*M*	*L*
M	*H*	*L*	*M*
M	*L*	*L*	*M*
M	*L*	*H*	*L*
M	*L*	*L*	*M*
M	*L*	*L*	*M*
H	*H*	*M*	*M*

شکل (۱): r: یک نمونه رابطه شامل مقادیر گسسته

تعریف ۲- مجموعه رشته های دودویی عضویت برای خصیصه A (که با MB-Set(A) نشان داده می‌شود) مجموعه ای است شامل تمام رشته های MBS مربوط به خصیصه A (با توجه به تمام مقادیر گسسته موجود در دامنه A).

مثال ۲: با توجه به تعریف فوق، MB-Set(c) برای رابطه r شکل (۱) به صورت زیر بدست می‌آید:

MB-Set(*C*) = { MBS(*C*,'*H*'), MBS(C,'*M*'), MBS(C,'*L*') }
={001000001000, 110001000001, 000110110110}

۳-۲- محاسبه درجه صحت یک وابستگی تقریبی

برای یک رابطه مانند r، دارای ساختار رابطه ای R، ابتدا MBS هر یک از خصیصه های موجود در R را برای هر یک از مقادیر گسسته ممکن محاسبه کرده و بر اساس آنها مجموعه های MB-Set تمام خصیصه ها را تشکیل می‌دهیم. پیچیدگی زمانی این عمل O(|R|.|r|) است. با این کار تمام اطلاعات مورد نیاز جهت محاسبه میزان اعتبار یک وابستگی در MB-Set ها نهفته می‌شود و به روز شدن رابطه هم نیازی به پیمایش مجدد کل تاپل ها ندارد.

به منظور محاسبه درجه اعتبار یک وابستگی خاص، ابتدا مجموعه ای به نام M را به صورت زیر می‌سازیم. هر یک از رشته های بیتی موجود در مجموعه MB-Set هر کدام از خصیصه های موجود در سمت چپ وابستگی را با اعضای سایر مجموعه های MB-Set سمت چپ وابستگی دو به دو AND منطقی کرده و حاصل را به مجموعه M اضافه می‌کنیم. به عنوان مثال اگر بخش LHS یک وابستگی شامل ۳ خصیصه باشد که هر خصیصه دارای ۴ رشته MBS است، مجموعه M شامل ۴[۳] عضو خواهد بود. در گام بعد مجموعه دیگری به نام F با استفاده از عملیات منطقی خاصی که در الگوریتم شکل (۲) دیده می‌شود، ساخته می‌شود. مجموعه F می‌تواند برای اندازه گیری درجه صحت وابستگی بکار رود. برای این منظور، عمل OR منطقی را روی تمام اعضاء مجموعه F اعمال می‌کنیم تا به یک رشته بیتی واحد برسیم. این رشته را S می‌نامیم. تعداد بیت های ۱ موجود در رشته S نشان دهنده ی تعداد تاپل هایی است که از آن وابستگی تبعیت

نمی‌کنند. با شمارش این بیت ها و استفاده از رابطه ۲، درجه صحت وابستگی محاسبه می‌شود.

الگوریتم MeasureAccuracy

ورودی: مجموعه خصیصه های سمت چپ وابستگی (LHS)، خصیصه سمت راست وابستگی (RHS)

خروجی: درجه صحت وابستگی LHS → RHS

۱- مجموعه جدیدی به نام M بسازید. اعضای مجموعه های MB-Set خصیصه های LHS را دوبدو با هم AND کنید. در صورتیکه حاصل یک رشته تماما صفر نیست، آن را به M اضافه کنید.

۲- یک مجموعه تهی به نام F بسازید.
به ازای هر عضو m از مجموعه M
به ازای هر عضو c_i از مجموعه MB-Set مربوط به RHS
$f_i \leftarrow m \text{ AND } (\text{NOT}(c_i))$
f_i شامل کمترین تعداد بیت ۱ را در tmp ذخیره کنید.
tmp را به مجموعه F اضافه کنید.

۳- تمام اعضای مجموعه F را با هم OR کرده و رشته بیتی حاصل را S بنامید.

۴- تعداد بیت های ۱ موجود در رشته S را شمرده و آن را جهت محاسبه درجه صحت وابستگی در معادله (۲) قرار دهید.

شکل (۲): Measure-Accuracy: محاسبه درجه صحت یک وابستگی

$$\text{درجه صحت} = \frac{\text{تعداد بیت های ۱} - \text{تعداد کل تاپل ها}}{\text{تعداد کل تاپل ها}} \quad (۲)$$

مثال ۳: وابستگی AB → C را در رابطه r که یک رابطه نمونه از شمای رابطه ای R(A, B, C, D) است ، در نظر بگیرید. با استفاده از الگوریتم شکل (۲)، درجه صحت این وابستگی بصورت زیر محاسبه می‌شود:

MB-Set(C) =
= {001000001000, 110001000001, 000110110110}
MB-Set(A) =
= {111111000001, 000000111110, 0000000000000}
MB-Set(B) =
= {000110100001, 000001000000, 111000011110}

M = {000110000001, 000001000000, 111000000000, 000000100000, 00000000000, 0000000011110, 00000000000, 00000000000, 00000000000}

F = {000000000001, 000000000000, 001000000000, 000000000000, 000000001000}

S = 001000001001

در این مثال، مجموعه M با اعمال عملیات AND بر روی مجموعه های MB-Set خصیصه های A و B (که در سمت چپ وابستگی قرار دارند)، بدست آمده است. از آنجا که هر دو مجموعه MB-Set شامل ۳ عضو هستند، مجموعه M دارای ۹=$۳^۲$ عضو می‌باشد. اما از آنجا که ۴ مورد از اعضای M رشته های تمام صفر هستند، وجود این رشته ها تاثیری در نتیجه نهایی ندارد، برای جلوگیری از یکسری عملیات اضافی می‌توان آنها را از مجموعه M حذف کرد. پس در اینجا فقط ۵ عضو باقیمانده مجموعه M جهت ساخت مجموعه F بکار رفته اند و به همین دلیل است که مجموعه F شامل ۵ عضو است.

در این مثال، رشته S شامل ۳ بیت یک است. موقعیت این بیت ها اندیس های ۳، ۹ و ۱۲ است. یک ویژگی کلیدی روش مورد بحث این است که این اندیس ها دقیقاً به محل استثناها یعنی تاپل هایی از رابطه که درجه صحت وابستگی را کم می‌کنند، اشاره می‌کنند. با استفاده از رابطه ۲، درجه صحت وابستگی بصورت زیر محاسبه می‌شود: ٪۷۵ = ۱۲ / (۳ - ۱۲)

۳-۳- الگوریتم جستجوی وابستگی های تقریبی

جهت بدست آوردن مجموعه تمام وابستگی های کمینه از یک الگوریتم جستجو به نام AD-Miner استفاده می‌کنیم. به ازاء هر خصیصه مانند A از شمای رابطه ای R، وابستگی های با فرم A → ? را جستجو می‌کنیم تا تمام ترکیبات ممکن خصیصه ها که کمینه هستند را برای سمت چپ وابستگی بیابیم. در این فرایند جستجو یک ترتیب ثابت برای خصیصه ها (مثلاً بر اساس الفبا) در نظر می‌گیریم. لیست خصیصه هایی که می‌توانند برای سمت چپ وابستگی A → ? به کار روند، تمام خصیصه های R بجز خود A هستند. برای یافتن سمت چپ وابستگی، هر یک از خصیصه ها به نوبت و با شروع از انتهای لیست انتخاب می‌شوند. برای هر خصیصه ای مانند X که انتخاب می‌شود، یک روال جستجوی اول عمق را به کار می‌گیریم تا یک مجموعه از خصیصه ها شامل X را که کمینه بوده و وابستگی حاصل از آن درجه صحت مورد قبولی داشته باشد، بیابیم. برای این منظور ابتدا درجه صحت وابستگی X → A با استفاده از الگوریتم شکل (۲) محاسبه می‌شود. اگر صحت این وابستگی از حد آستانه مورد قبول بیشتر باشد، پذیرفته می‌شود. در غیر اینصورت سمت چپ وابستگی را گسترش می‌دهیم تا زمانی که به یک وابستگی قابل قبول برسیم. گسترش سمت چپ وابستگی بایستی در چارچوب دو قانون زیر انجام شود:

قانون ۱- تنها خصیصه هایی می‌توانند به سمت چپ وابستگی (X) اضافه شوند که در لیست ثابت خصیصه ها بعد از X قرار گرفته باشند.

قانون ۲- مجموعه خصیصه هایی که به طرف چپ وابستگی افزوده می‌شوند، نباید به تنهایی برای طرف چپ وابستگی قابل قبول باشند. به عبارت دیگر وابستگی حاصل از گسترش طرف چپ بایستی کمینه باشد.

جستجو برای یک مجموعه از خصیصه ها شامل خصیصه ی X جهت سمت چپ یک وابستگی زمانی خاتمه می‌یابد که خصیصه دیگری جهت گسترش طرف چپ وابستگی وجود نداشته باشد.

الگوریتم AD-Miner

ورودی: اسکیمای رابطه ای R، مجموعه های MB-Set مربوط به رابطه نمونه r (دارای ساختار R)، حد آستانه درجه صحت

خروجی: مجموعه وابستگی های کمینه با درجه صحت بالاتر از حد آستانه (f_r)

۱- $f_r \leftarrow \{\}$

۲- به ازای هر خصیصه A از R (به عنوان RHS)

به ازای هر خصیصه B ($B \neq A$) از R (به عنوان LHS)

$f \leftarrow$ Generate-ADs(B,A)

$f_r \leftarrow f_r \cup f$

۳- f_r را برگردانید.

تابع Generate-ADs

ورودی: مجموعه خصیصه های سمت چپ (X)، خصیصه سمت راست (Y)

خروجی: مجموعه وابستگی های کمینه (f) که RHS آنها Y بوده و LHS شان شامل X باشد.

درجه صحت $\rightarrow$ MeasureAccuracy(MB-Set(X) , MB-Set(Y))

اگر درجه صحت از حد آستانه کمتر است،

به ازای هر خصیصه Z از R، که ۱) Z در لیست بعد از خصیصه های X واقع باشد و ۲) وابستگی کمینه ای شامل Z در سمت چپ و Y در راست تاکنون تولید نشده باشد.

Z را به مجموعه X اضافه کنید.

$f \leftarrow$ Generate-ADs (X, Y).

در غیر این صورت وابستگی ($X \rightarrow Y$) را به مجموعه f اضافه کنید.

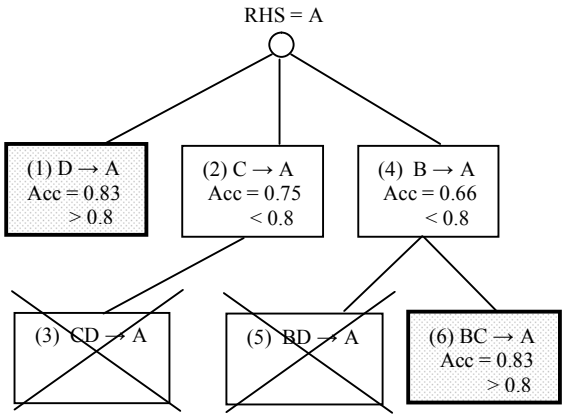
f را برگردانید.

شکل (۳): AD-Miner: الگوریتم کشف وابستگی های تقریبی کمینه

مثال ۴: رابطه r را که در شکل (۱) نشان داده شده، در نظر بگیرید. با فرض حد آستانه ۰/۸ و با استفاده از الگوریتم AD-Miner می‌خواهیم مجموعه وابستگی های کمینه را که سمت راستشان خصیصه A است، بیابیم.

با در نظر گرفتن ترتیب ثابت خصیصه ها (بر اساس حروف الفبا)، خصیصه هایی که می‌توانند بعنوان LHS بکار روند، تمام خصیصه ها بجز A را شامل می‌شوند. الگوریتم کار خود را با انتخاب خصیصه انتهای لیست (خصیصه D) بعنوان LHS آغاز می‌کند. از آنجا که درجه صحت وابستگی $D \rightarrow A$ از حد آستانه بالاتر است، پذیرفته شده و نیازی به گسترش بیشتر LHS نیست. خصیصه C که در لیست بلافاصله قبل از D واقع است، خصیصه بعدی جهت ارزیابی بعنوان LHS است. با اینکه درجه صحت وابستگی $C \rightarrow A$ قابل قبول نیست، اما تنها خصیصه ای که بعد از C در لیست قرار دارد و طبق قانون ۱ می‌تواند برای توسعه LHS بکار رود، خصیصه D است که با توجه به قانون ۲ مجاز به استفاده از آن نیستیم، چرا که با توجه به برقراری وابستگی $D \rightarrow A$ دیگر وابستگی $CD \rightarrow A$ کمینه نیست. فرایند جستجو با انتخاب خصیصه بعدی (B) ادامه می‌یابد. درجه صحت وابستگی $B \rightarrow A$ نیز کافی نیست و LHS نیاز به توسعه دارد. تنها خصیصه های بعد از B در لیست که می‌توانند به LHS اضافه شوند، خصیصه های C و D هستند. وابستگی $BD \rightarrow A$ به همان دلیل فوق رد می‌شود و $BC \rightarrow A$ با توجه به درجه صحت قابل قبول آن پذیرفته می‌شود. بنابراین در این مثال، وابستگی های تقریبی کمینه عبارتند از $D \rightarrow A$ و $BC \rightarrow A$.

در شکل (۴) از یک ساختار درختی بمنظور نمایش شماتیک عملیات فوق استفاده کرده ایم. هر گره در این درخت نشاندهنده یک وابستگی کاندید است. در طی فرایند جستجو، الگوریتم درجه صحت وابستگی هایی را که در شکل با علامت ❋ مشخص شده اند، بدلیل کمینه نبودن محاسبه نمی کند. در هر گره شماره ای دیده می‌شود که این شماره ها ترتیب ملاقات گره ها را در الگوریتم جستجو نشان می‌دهند.



شکل(۴): فرایند جستجوی AD های کمینه بازاء RHS = A

۳-۳- ویژگیهای کلیدی الگوریتم AD-Miner

در بخش قبل، الگوریتم AD-Miner بعنوان یک روش کاوش وابستگی های تقریبی کمینه در بانکهای اطلاعات رابطه ای ارائه شد. این الگوریتم نسبت به استثناها و داده های نویزدار انطاف پذیر است و درجه صحت تمام وابستگی های کشف شده را نیز محاسبه می‌کند. به علاوه، این الگوریتم برای هر وابستگی موقعیت تاپلهایی که در وابستگی صدق نمی کنند (اندیس تاپل ها در رابطه) را نیز نشان می‌دهد. این ویژگی، بخصوص زمانی مفید است که مایل به یافتن استثناها و داده های ناسازگار در یک مجموعه داده باشیم.

ویژگی مهم دیگر AD-Miner، افزایشی بودن آن است. دلیل این امر آن است که تمام اطلاعات مربوط به داده ها در رشته های MBS نگهداری می‌شود. زمانی که چند تاپل جدید به رابطه افزوده می‌شوند، رشته های MBS مربوط به داده های جدید ساخته شده و به انتهای MBS های موجود الحاق می‌شوند. بدنبال آن تغییرات لازم بر روی مجموعه های M و F اعمال شده و رشته S جدید جهت بروز کردن درجه صحت وابستگی ها بکار برده می‌شود که ممکن است منجر به حذف برخی از وابستگی ها شود. بدین ترتیب در هر بار تغییر و بروز شدن یک رابطه نیازی به پیمایش مجدد کل داده ها (که بسیار زمانبر است) جهت هرس کردن و بروز آوری مجموعه وابستگی ها نداریم.

AD-Miner از سایر الگوریتم های کشف وابستگی سریع تر عمل می‌کند. دلیل اصلی این کارایی از آن جا است که AD-Miner بسیاری از عملیات زمانبر را با یکسری عملیات منطقی سریع بر روی رشته های بیتی انجام می‌دهد. تحلیل پیچیدگی زمانی AD-Miner در بخش بعد ارائه می‌شود.

۴- تحلیل پیچیدگی زمانی

فرض کنیم $|R| = n$ تعداد خصیصه ها در اسکیمای رابطه ای R ، $|r| = m$ تعداد خصیصه های رابطه r (که دارای ساختار R است)، $|f_r| = f$ تعداد وابستگی هایی که ارزیابی می‌شوند، $|LHS|$ حداکثر تعداد خصیصه ها در سمت چپ وابستگی‌ها و $|D| = d$ متوسط تعداد مقادیر گسسته موجود در دامنه مقادیر خصیصه ها (به عبارتی تعداد اعضای مجموعه های MB-Set) باشند.

در الگوریتم پیشنهادی عمل محاسبه MB-Set ها از مرتبه $O(n.m)$ و ساختن مجموعه M برای یک وابستگی از مرتبه $O(d^{|LHS|})$ می‌باشد. از آنجا که به ازاء هر عضو مجموعه M، به تعداد d عمل AND منطقی بایستی انجام شود، پیچیدگی محاسبه مجموعه F از مرتبه $O(d^{|LHS|+1})$ و در مجموع پیچیدگی کلی الگوریتم $O(n.m + f.d^{|LHS|+1})$ خواهد بود. بهترین حالت زمانی اتفاق می‌افتد که نیازی به توسعه سمت چپ هیچ یک از وابستگی ها پیدا نشود، به عبارت دیگر درجه صحت تمام وابستگی هایی که شامل ۱ خصیصه در سمت چپ هستند قابل قبول باشد. در این حالت $|LHS| = 1$ است و f کمترین مقدار خود یعنی مقدار n^2 را داراست. بنابراین پیچیدگی کلی الگوریتم در بهترین حالت از مرتبه $O(n.m + n^2.d^2)$ می‌شود. بدترین حالت زمانی اتفاق می‌افتد که هیچ نظم و قاعده ای در بین داده ها وجود نداشته باشد و نتوان هیچ وابستگی جالبی با تعداد کمی خصیصه در سمت چپ پیدا کرد. در این حالت سمت چپ تمام وابستگی ها را باید آنقدر توسعه دهیم تا هیچ خصیصه دیگری برای اضافه شدن به سمت چپ باقی نماند. بنابراین $|LHS|$ برابر با n-1 خواهد بود و تعداد کل وابستگی هایی که تست می‌شوند کل فضای جستجو (یعنی 2^n وابستگی) را شامل می‌شود. اما در عمل، این حالت خصوصا در مورد داده های واقعی بسیار نادر است.

در الگوریتم FastFDs، گام اول محاسبه مجموعه *تفاوت ها*[۱۸] است که پیچیدگی زمانی آن از مرتبه $O(n.m^2.\log n.m^2)$ می‌باشد. بخش اصلی این الگوریتم تابعی به نام FindCovers است که مجموعه وابستگی های کمینه را بدست می‌آورد و پیچیدگی آن $O((1 + w(n)).f)$ است. $w(n)$ تابعی است متناسب با تعداد خصیصه های رابطه که کل زمان تلف شده الگوریتم به دلیل کارهای اضافه ای که در بعضی قسمت ها باید انجام شود را نشان می‌دهد. در مجموع پیچیدگی زمانی الگوریتم FastFDs از مرتبه $O(n.m^2 + n.m^2.\log n.m^2 + n.(1 + w(n)).f)$ می‌باشد. با مقایسه این پیچیدگی و پیچیدگی الگوریتم پیشنهادی می‌توان دریافت که روش ما خصوصا در مواردی که تعداد تاپل ها بسیار زیاد است، بطور قابل توجهی سریع تر است.

علاوه بر حالت کلی، الگوریتم ما بصورت افزایشی نیز عمل می‌کند، یعنی با دادن مجموعه وابستگی های کشف شده به آن، در هنگام اضافه شدن تعدادی تاپل جدید قادر است بدون پیمایش کل داده ها وابستگی ها را بروز رسانی کند. اما از آنجا که الگوریتم FastFDs یک روش افزایشی نیست، برای حالت افزایشی، الگوریتم را با دو الگوریتم افزایشی *مقایسه دو بدو* و روش *پارتیشن بندی تاپل ها* مقایسه می‌کنیم.

زمانی که تعدادی تاپل (مثلا به تعداد k) به رابطه ای مانند r افزوده می‌شود، تنها کاری که بایستی انجام شود این است که با توجه به مقادیر جدید، بیت های مناسبی (k بیت) به انتهای رشته های MBS الحاق شده و بر اساس آنها مجموعه های M و F جدید محاسبه شوند. پیچیدگی این عملیات از مرتبه $O(n.k + f.d^{|LHS|+1})$ می‌باشد.

در الگوریتم *مقایسه دو بدو* هر تاپل جدیدی که به رابطه اضافه می‌شود باید مقادیر خصیصه هایش با مقادیر تک تک تاپل ها مقایسه شود تا بر اساس این مقایسه ها مجموعه وابستگی ها تصحیح شود. بهترین حالت زمانی اتفاق می‌افتد که مجموعه وابستگی ها بدون تغییر بماند. در این حالت پیچیدگی زمانی از مرتبه $O(d^2.n^2.k.f.m)$ می‌شود. در بدترین حالت نیز که تمام وابستگی ها باید اصلاح شوند، مرتبه پیچیدگی الگوریتم $O(d^2.k.n.m.2^n)$ است.

روش پارتیشن بندی تاپل ها بر خلاف روش مقایسه دوبدو با اضافه شدن k تاپل تنها یک مرحله عملیات انجام می‌دهد. پیچیدگی عملیات تصحیح وابستگی ها در بهترین حالت این الگوریتم $O(d^2.n + m.2^n)$ و در بدترین حالت $O(d^2.n + (m + k).n + f)$ می‌باشد.

جدول (۱): پیچیدگی زمانی الگوریتم های افزایشی کاوش وابستگی

پیچیدگی	روش مقایسه دوبدو	روش پارتیشن بندی	روش پیشنهادی
بهترین حالت	$O(d^2.n^2.k.f.m)$	$O(d^2.n+(m+k).n+f)$	$O(k.n + n^2.d^2)$
بدترین حالت	$O(d^2.k.n.m.2^n)$	$O(d^2.n + m.2^n)$	$O(k.n+ 2^n .d^n)$

در عمل، بندرت ما به دنبال وابستگی هایی با تعداد زیاد خصیصه در سمت چپ هستیم و $|LHS|$ معمولا عدد کوچکی است. با در نظر گرفتن این نکته روش ما همواره نسبت به روش FastFDs به زمان بسیار کمتری برای تولید مجموعه وابستگی های کمینه نیاز دارد.

تحلیل پیچیدگی زمانی موارد دیگری از الگوریتم های کاوش وابستگی در مرجع [14] ارائه شده است.

۵- نتایج عملی

۵-۱- داده های تولید شده تصادفی

در این آزمایش، کارایی زمانی الگوریتم AD-Miner را با الگوریتم FastFDs (که از سایر روش های کاوش وابستگی کاراتر است) بر روی داده های تصادفی مقایسه می‌کنیم. هر دو الگوریتم با زبان C++ پیاده سازی شده و بر روی یک سیستم دارای پردازنده ۳ گیگاهرتز و حافظه اصلی ۵۱۲ مگا بایت به اجرا درآمدند. الگوریتم AD-Miner قادر به کاوش وابستگی ها با هر حدی از درجه صحت می‌باشد. اما از آنجا که الگوریتم FastFDs تنها برای کشف وابستگی های دقیق (درجه صحت ۱۰۰٪) طراحی شده است، برای مقایسه دو الگوریتم حد آستانه درجه صحت را برابر با ۱/۰ قرار دادیم.

برای مقایسه زمان های اجرای دو الگوریتم، رابطه هایی با تعداد تاپل ها (m) و تعداد خصیصه های (n) مختلف بصورت تصادفی تولید کردیم. شکل (۵) نشاندهنده زمان اجرای دو الگوریتم بازاء $d = ۳$ و مقادیر مختلف m و n می‌باشد. همانطور که دیده می‌شود زمان اجرای AD-Miner بازاء اکثر مقادیر m و n، خصوصا مقادیر بزرگ m (که گلوگاه عمده ای در مسائل داده کاوی است) کمتر از زمان اجرای FastFDs است. همچنین نرخ رشد این زمان با افزایش تعداد تاپل ها بسیار کمتر از نرخ افزایش زمان اجرا در FastFDs می‌باشد.

۵-۲- داده های واقعی

در این آزمایش، کارایی الگوریتم ها را بر روی تعدادی مجموعه داده واقعی، برگرفته از بانک داده های UCI مورد ارزیابی قرار دادیم. آمار کلی مجموعه داده های بکار رفته در جدول (۲) آمده است. برخی از این مجموعه داده ها دارای خصیصه هایی با مقادیر عددی پیوسته نیز می‌باشند. قبل از شروع عملیات اصلی، مقادیر این گونه خصیصه ها با پارتیشن کردن دامنه آنها به ۵ بخش مساوی به مقادیری گسسته تبدیل شدند. همانند آزمایش قبل، به منظور یکسان کردن شرایط دو الگوریتم، مقدار حد آستانه درجه صحت را برابر با ۱/۰ در نظر گرفتیم. نتایج بدست آمده از اجرای دو الگوریتم در جدول (۳) آمده است.

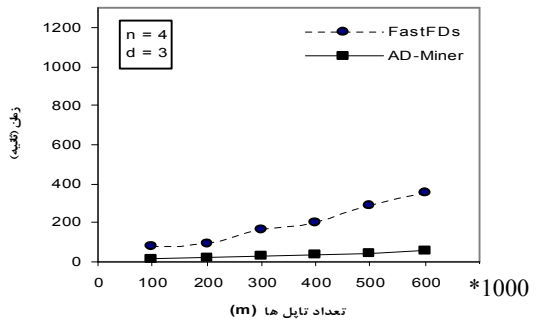
جدول (۲): مشخصات آماری مجموعه داده های آزمایش ۲

مجموعه داده	تعداد خصیصه ها		تعداد تاپل ها
	گسسته	پیوسته	
Flare	-	۱۳	۱۳۸۹
Letter Recognition	۱	۱۶	۲۰,۰۰۰
Adult	۹	۶	۴۸,۸۴۲
Nursery	۹	-	۱۲,۹۶۰
Chess	۷	-	۲۸,۰۵۶

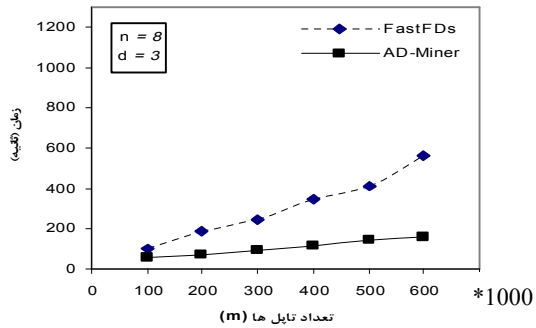
جدول (۳) نتایج آزمایش بر روی داده های واقعی

زمان (ثانیه)

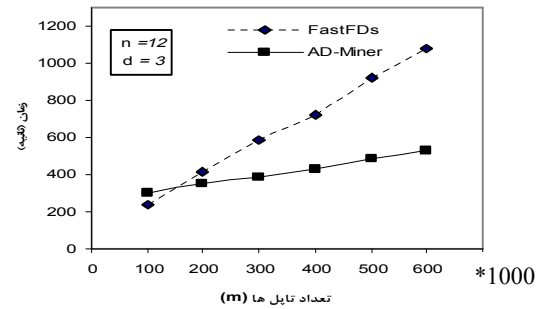
	Nursery	Adult	Letter recognition	Flare
AD-Miner	۸۲	۲,۱۷۵	۹۵۴	۲۹
FastFDs	۲۹۵	۱۰,۱۲۸	۱,۷۶۳	۱۸



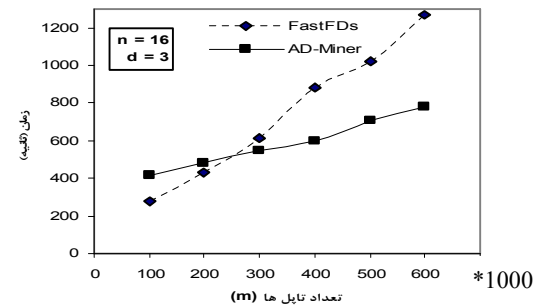
الف- تعداد خصیصه ها: ۴، تعداد تاپل ها: متغیر بین ۱۰۰,۰۰۰ تا ۶۰۰,۰۰۰



ب- تعداد خصیصه ها: ۸، تعداد تاپل ها: متغیر بین ۱۰۰,۰۰۰ تا ۶۰۰,۰۰۰



ج- تعداد خصیصه ها: ۱۲، تعداد تاپل ها: متغیر بین ۱۰۰,۰۰۰ تا ۶۰۰,۰۰۰



د- تعداد خصیصه ها: ۱۶، تعداد تاپل ها: متغیر بین ۱۰۰,۰۰۰ تا ۶۰۰,۰۰۰

شکل (۵): مقایسه زمان اجرای الگوریتم ها بر روی داده های تصادفی، بازاء مقادیر مختلف m (تعداد تاپل ها) و n (تعداد خصیصه ها)

۶- نتیجه

در این مقاله، یک الگوریتم افزایشی جدید برای کاوش وابستگی های تقریبی در بانک های اطلاعات رابطه ای ارائه کردیم. عملیات اصلی در این الگوریتم با استفاده از عملگرهای منطقی بر روی رشته های بیتی صورت می‌گیرد. پیچیدگی زمانی این الگوریتم با برخی از الگوریتم های افزایشی و غیر افزایشی موجود مقایسه و مورد تحلیل قرار گرفت.

برای ارزیابی کارایی الگوریتم از لحاظ زمان اجرا، دو مدل آزمایش ترتیب داده و کارایی الگوریتم را با یکی از الگوریتم های کارآمد به نام FastFDs مقایسه کردیم. نتایج این دو آزمایش نشان می‌دهد که روش ما از کارایی بسیار بالاتری نسبت به روش FastFDs بخصوص در مواردی که تعداد تاپل ها زیاد است، برخوردار می‌باشد. به عبارت دیگر با زیاد شدن تعداد تاپل ها زمان اجرای روش ما بسیار پایدارتر است. دلیل عمده این امر آنست که در روش ما رابطه بین عملیات اصلی با تعداد تاپل ها یک رابطه خطی و در الگوریتم FastFDs مربعی است.

با زیاد شدن تعداد خصیصه ها مشاهده کردیم که کارایی AD-Miner در مقایسه با FastFDs دچار افت می‌شود. علت این مساله این است که ما هیچ محدودیتی بر روی تعداد خصیصه های موجود در سمت چپ وابستگی ها اعمال نمی‌کنیم. اما در عمل و از دیدگاه داده کاوی و اخذ دانش، معمولا وابستگی هایی جالب و ارزشمند هستند که کوچک باشند. پس در صورتی که برای طول وابستگی ها محدودیت قائل شویم، الگوریتم FastFDs حتی برای رابطه های دارای تعداد زیادی خصیصه نیز بسیار سریع تر از FastFDs عمل می‌کند.

مراجع

[1] P. Bosc, L. Lietard, and O. Pivert, "*Functional dependencies revisited under graduality and imprecision*", pp. 57 - 62, NAFIPS, 1997.

[2] F. Berzal, I.Blanco, D. Sanchez, J.M. Serrano and M.A. Vila, "*A definition for fuzzy approximate dependencies*", Fuzzy Sets and Systems, Vol. 149, No. 2, pp. 105 – 129, 2005.

[3] J. Kivinen and H. Mannila, "*Approximate inference of functional dependencies from relations*", Theoretical Computer Science, Vol. 149, No. 1, pp. 129 – 149, 1995.

[4] J.M. Morrissey, "*Imprecise information and uncertainty in information systems*", ACM Transactions on Information Systems, Vol.8, No. 2, pp. 159–180, 1990.

[5] U. Nambiar, and S. Kambhampati, "*Answering Imprecise Queries over Autonomous Web Databases*", In: Proc. ICDE 2006, 22nd International Conference on Data Engineering, 2006.

[6] C. Wyss, C. Giannella, and E. Robertson, "*FastFDs, A Heuristic-Driven Depth-First Algorithm for Mining Functional Dependencies from Relation Instances*", Data Warehousing and Knowledge Discovery, Vol. 2114, No. 1, pp. 101 – 123, 2001.

[7] S.L. Wang, J.W. Shen, and T.P. Hong, "*Discovering functional dependencies Incrementally from Fuzzy Relational Databases*", in: Proc. English National Conference on Fuzzy Theory and Its Applications, pp. 17 – 24, 2000.

[8] S.L. Wang, J.S. Tsai, B.C. Chang, "*Mining Approximate Dependencies using partitions on Similarity-Relation-based Fuzzy databases*", in : Proc. IEEE SMC'99, Vol. 6, pp. 871_875, 1999.

[9] S.L. Wang, J.S. Tsai and T.P. Hong, "*Discovering functional dependencies from Similarity-based Fuzzy Relational Databases*", Intelligent Data Analysis, Vol. 5, No. 2, pp. 131 – 149, 2001.

[10] P. A. Flach and I. Savnik, "*Database dependency discovery: a machine learning approach*", AI communications, Vol. 12, No. 3, pp. 139 – 160, 1999.

[11] H. Mannila and K.J. Räihä. "*Algorithms for inferring functional dependencies from relations*". DKE, Vol. 12, No. 1, pp. 83-99, 1994.

[12] S. Lopes , J.M. Petit , L. Lakhal, "*Efficient Discovery of Functional Dependencies and Armstrong Relations*", in: Proc. ICDT 2000, the 7th International Conference on Extending Database Technology: Advances in Database Technology, Vol. 1777, pp. 350 – 364, 2000.

[13] Y. Huhtala, J. Kärkkäinen, P. Porkka and H. Toivonen, "*TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies*". The Computer Journal. Vol. 42, No. 2, pp. 100 – 111, 1999.

[14] H. Mannila , K.J. Räihä, "*On the complexity of inferring functional dependencies*", Discrete Applied Mathematics, Vol. 40, No. 2, pp. 237 – 243, 1992.

[15] B.P. Buckles and F.E. Petry, "*A fuzzy representation of data for relational databases*", Fuzzy Sets and Systems, Vol. 7, No. 1, pp. 213 – 226, 1982.

زیرنویس‌ها

[1] Incremental method
[2] Functional Dependencies
[3] Data Mining
[4] Knowledge Discovery
[5] Approximate Dependencies
[6] Relational database
[7] Incremental data mining
[8] Belady anomaly
[9] Left-Hand Side
[10] Right-Hand Side
[11] Minimal
[12] Top-down
[13] Bottom-up
[14] Decision tree
[15] Difference Set
[16] Level-wise search
[17] Depth-first search
[18] Difference set