min National Conference
سومین همایش ملی برق و کامپیوتر
ایران
اردیبهشت ماه ۹۶
dataacademy.ir

الگوریتم های به روز رسانی افزایشی الگوهای پرتکرار مبتنی بر FP-growth

نعیمه نجارزاده ورنوسفادرانی*[۱]، ندا نجارزاده ورنوسفادرانی[۲]، امیرحسین امیرخانی[۳]
[۱]دانشجو، دانشکده مهندسی کامپیوتر، واحد نجف آباد، دانشگاه آزاد اسلامی، نجف‌آباد، اصفهان، ایران
naeemehnajarzadeh@yahoo.com
[۲] دانشجو، دانشکده مهندسی کامپیوتر، واحد نجف آباد، دانشگاه آزاد اسلامی، نجف‌آباد، اصفهان، ایران
neda.najarzadeh.90@gmail.com
[۳] دانشیار دانشکده مدیریت دانشگاه پیام نور تهران
amirhosseinamirkhani944@gmail.com

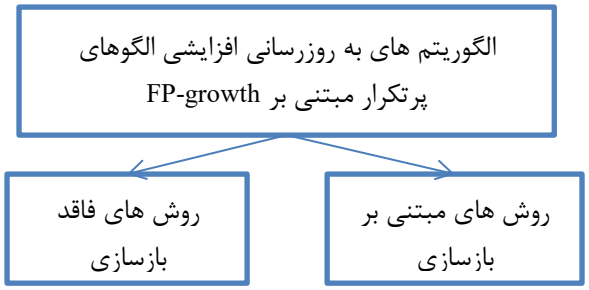
چکیده - کاویدن الگوهای پرتکرار یک موضوع مهم تحقیقاتی در داده کاوی به شمار می آید. در بسیاری از برنامه های کاربردی، پایگاه داده ها نیاز به به روزرسانی اعم از اضافه، حذف یا تغییر تراکنش ها دارند. یک راه حل برای این مسئله کاویدن تمام الگوهای پرتکرار از ابتدا می باشد که در پایگاه داده های بزرگ بسیار پرهزینه می باشد. از اینرو راه حل پیشنهادی بهینه، کاویدن افزایشی برای به روزرسانی الگوهای پرتکرار به جای کاویدن تمام الگوها از ابتدا می باشد. در این مقاله الگوریتم های مبتنی بر *FP-growth* برای به روزرسانی الگوهای پرتکرار مورد نقد و بررسی قرار می گیرند. این روش ها شامل دو دسته الگوریتم های مبتنی بر متد بازسازی و الگوریتم های فاقد متد بازسازی می باشد.

کلید واژه- داده کاوی؛ الگوهای پرتکرار؛ به روزرسانی افزایشی

* naeemehnajarzadeh@yahoo.com

۱- مقدمه

از زمان معرفی داده کاوی، داده کاوی وکشف دانش از پایگاه داده ها توجه زیادی از محققان را به خود جلب کرده است و تحقیقات مهم و متعددی در این زمینه صورت گرفته است [۹-۱]. استخراج الگوهای پر تکرار یک زمینه تحقیقاتی بسیار مهم در زمینه داده کاوی با کاربردهای بسیار متنوع از جمله بازاریابی، تجارت الکترونیک، پزشکی، صنعت، شیمی و بسیاری از علوم دیگر می باشد. الگوریتم Apriori اولین الگوریتم برای استخراج الگوهای پرتکرار بود که معرفی شد [10] و با روش تولید کاندید و تست الگوها را استخراج میکرد که به دلیل دو مشکل تولید کاندیداها و اسکن مکرر پایگاه داده به خوبی عمل نمی کرد. از اینرو در سال ۲۰۰۰ آقای Han و همکارانش الگوریتم FP-growth را ارئه دادند [11]. این الگوریتم مبتنی بر ساختار درختی، الگوها را بدون تولید کاندید و تنها با دو اسکن از پیاگاه داده استخراج می کند. هر چند این الگوریتم مشکل الگوریتم Apriori را به خوبی حل کرد؛ اما خود این الگوریتم نیز از مشکل نداشتن قابلیت افزایشی رنج می برد. در بسیاری از حوزه ها مثل بازاریابی و تجارت الکترونیک، داده ها دائما در حال تغییر و پایگاه داده ها به صورت پویا می باشند که کاربران نیاز دارند با ورود اطلاعات جدید به صورت آنلاین تصمیم گیری و الگوها را استخراج نمایند. پس از آنجایی که در بسیاری از برنامه های کاربردی، پایگاه داده ها اغلب با حذف و اضافه و یا تغییر تراکنش روبرو هستند در نتیجه الگوهای پرتکرار استخراج شده از انها باید بروز رسانی شوند. یک راه حل ساخت و استخراج الگوها از ابتدا می باشد که در پایگاه داده های بزرگ بسیار پر هزینه می باشد. به همین دلیل در این زمینه، محققان به روزرسانی افزایشی را برای بروز رسانی موثر الگوهای پرتکرار به جای کاویدن همه الگوهای پرتکرار از ابتدا پیشنهاد می کنند و تحقیقات در این رابطه با ارائه الگوریتم های جدید و یا بهبود الگوریتم های پیشین برای استخراج الگوهای پر تکرار ادامه یافته است. الگوریتم های مبتنی بر FP-growth برای به روزرسانی افزایشی الگوهای پرتکرار به دو دسته الگویتم های فاقد بازسازی و مبتنی بر بازسازی تقسیم می شوند که در شکل ۱ مشاهده می شود. در این مقاله مهمترین الگوریتم های مبتنی بر FP-growth برای به روز رسانی افزایشی الگوهای پرتکرار از جمله CanTree، FUFP، CP-tree، IP-tree، SPO-tree و DFP-tree مورد بررسی قرار می گیرند.



شکل ۱: روش های ساخت درخت در الگوریتم های به روزرسانی افزایشی مبتنی بر FP-growth

۲- الگوریتم های فاقد متد بازسازی

در این روش های برای ساخت درخت به بازسازی شاخه های درخت نیازی نمی باشد.

۲-۱- الگوریتم CanTree

در سال ۲۰۰۷، leung و همکارانش الگوریتم CanTree را ارائه کردند[۱۲] .در CanTree آیتم ها بر اساس ترتیب متعارفی که می تواند حروف الفبا باشد مرتب می شوند. ساختCanTree تنها نیاز به یک اسکن از پایگاه داده دارد. نودها به راحتی بدون هیچ هزینه جستجویی برای ادغام کردن مسیرها می توانند در درخت CanTreeدرج شوند و ازآنجایی که ترتیب متعارف ثابت است هیچ تغییری در تعداد تکرار آیتم ها ناشی از به روزرسانی افزایشی در ترتیب آیتم ها در CanTree تاثیری ندارد.

محدودیت ها:

- درخت غیرفشرده تولید می کند و بنابراین سرعت استخراج پایین است.
- به دلیل غیرفشردگی درخت حافظه زیادی مصرف می کند.

۲-۲- الگوریتم FUFP[۲]

در سال ۲۰۰۸، Tzung-Pei الگوریتم FUFP را برای به روزرسانی افزایشی الگوهای پرتکرار معرفی کرد.[13] این الگوریتم بعد از ساخت درخت اگر پایگاه داده جدیدی اضافه شود، به منظور به روزرسانی الگوهای پرتکرار، زمانی که آیتم جدیدی اضافه می شود بسته به شرایط آیتم ورودی، برای درج آیتم جدید هر تراکنش در درخت به یکی از چهار روش زیر عمل میکند:

(i) اگر قبل و بعد از اضافه شدن پایگاه داده جدید، آیتمی که پرتکرار بوده همچنان پرتکرار بماند، نیاز به تغییر نداریم.

(ii) اگر غیر پرتکرار باشد و پرتکرار بشود به عنوان گره برگ اضافه می شود.

(iii) اگر پرتکرار باشد و غیر پرتکرار بشود از پایگاه داده حذف میشود.

(iv) اگر در هر دو غیر پرتکرار باشد، نیاز به تغییر ندارد.

محدودیت ها:

-درخت ساخته شده دارای فشردگی لازم نیست، زیرا گره جدید که به عنوان برگ به شاخه ی درخت اضافه میشود باعث میشود سایز درخت بزرگ بشود.

-وقتی یک ایتم پرتکرار می شود نیاز به اسکن مجدد پایگاه داده دارد.

۳- الگوریتم های مبتنی بر متد بازسازی

در این روش ها برای ساخت درخت به بازسازی شاخه های درخت نیاز می باشد.

۳-۱- الگوریتم CP-tree

در سال ۲۰۰۹، Tanbeer و همکارانش الگوریتم CP-tree را معرفی کردند[۱۴] . در این الگوریتم از یک لیست به نام I-list جهت تسهیل پیمایش درخت استفاده می شود و ایتدا یک بلوک از داده ها در درخت درج می شود و سپس درخت بازسازی می شود و به همین ترتیب تا پایان پایگاه داده بلاک به بلاک داده ها در درخت درج و سپس بازسازی می شوند. در مرحله ی درج آیتم های هر تراکنش بر اساس یک ترتیب منطقی وارد درخت می شوند؛ سپس در مرحله بازسازی ابتدا I-list بر اساس ترتیب نزولی فراوانی آیتم ها مرتب شده و بر اساس آن، درخت بازسازی می شود. برای مرتب کردن مسیرهای نا مرتب در درخت از متد مرتب سازی شاخه ای استفاده می شود.

محدودیت:

- درخت اولیه ممکن است غیرفشرده که قطعا زمان بازسازی زیادخواهد داشت و تعداد کمی از شاخه های موجود در درخت کاملا مرتب خواهند بود.
- این الگوریتم به دلیل اینکه بعد از درج هر بلاک از داده ها در درخت، درخت را بازسازی می کند بنابراین برای ساخت درخت از بازسازی های متعددی استفاده می کندکه بسیار زمانبر است.

۳-۲- الگوریتم IP-tree

در سال ۲۰۱۰، Zhou و همکارانش الگوریتم IP-tree را معرفی کردند [۱۵] ،که با یکبار اسکن پایگاه داده الگوها را استخراج می نماید. ابتدا با یک اسکن پایگاه داده تراکنش ها مطابق ترتیب حروف الفبا در درخت درج می شوند وسپس براساس ترتیب نزولی

[۲] Fast Updated FP-tree

فراوانی آیتم ها در F-list که لیست آیتم ها می باشد بازسازی می شود.

محدودیت ها:

درخت ساخته شده ی اولیه غیرفشرده می باشد لذا بازسازی درخت بسیار زمان بر خواهد بود.

۳-۳- الگوریتم SPO-tree

در سال ۲۰۱۱، Koh و همکارانش الگوریتم SPO-tree را پیشنهاد کردند [16]. ساخت درخت در الگوریتم SPO-tree شامل دو مرحله درج و بازسازی می شود، در فاز درج آیتم ها طبق ترتیب نزولی فراوانی آیتم ها وارد درخت می شوند و در فاز بازسازی در صورتی که نسبت مجموع فاصله تصحیح مطلق به مجموع حداکثر فاصله تصحیح بیشتر از کمترین فاصله تصحیح تعریف شده شود (طبق فرمول SPO) درخت یک بار بازسازی می شود .در این الگوریتم تراکنش ها قبل از درج بر اساس ترتیب نزولی فراوانی آیتم هادر I-list مرتب می شوند.

محدودیت :

به محاسبات زیادی نیاز داردوزمانگیر است که باعث کاهش سرعت استخراج الگو.ها می شود.

۳-۴- الگوریتم DFP-tree

در سال 2013 همدانیان و همکارانش الگوریتم DFP-tree را معرفی کردند [۱۷]، برای ساخت درخت DFP-tree از یک لیست موسوم به Last-Sup استفاده می شود که شامل آخرین فراوانی آیتم های موجود در تراکنش هاست. ابتدا تراکنش اول بر اساس یک ترتیب منطقی مانندترتیب حروف الفبا وارد درخت می شود، برای تراکنش های بعدی قبل از درج هر تراکنش، ابتدا فراوانی آیتم های موجود در Last-Sup به روز رسانی شده و سپس تراکنش مربوطه بر اساس این لیست مرتب و در درخت درج می شود. پس از یک اسکن پایگاه داده، بازسازی این درخت با کمک روشBranch-Sorting انجام می شود .در این روش کلیه مسیرهای موجود بررسی شده و در صورتی که ترتیب آیتم ها مطابق با ترتیب نزولی فراوانی آیتم ها نباشد مسیر مربوطه از درخت حذف وپس از مرتب سازی در یک آرایه موقت، مجدداً در درخت درج می شود.

۴- نتیجه‌گیری

در جدول ۱ الگوریتم های بررسی شده در این مقاله مورد مقایسه قرار گرفته اند. همان طور که در جدول ۱ نیز مشاهده می شود تمامی الگوریتم ها به جز الگوریتم FUFP با یک اسکن پایگاه داده، الگوهای پرتکرار را استخراج می نمایند. الگوریتم FUFP به دلیل اینکه در بدترین حالت مجبور به اسکن دوباره ی پایگاه داده است در پایگاه داده های بزرگ پرهزینه عمل میکند. ضمن اینکه درخت غیرفشرده ای نیز تولید میکند که موجب زمانبر شدن استخراج الگوهای پرتکرار در آن می شود. همچنین CanTree نیز به دلیل ایجاد درخت غیرفشرده، پروسه ی استخراج الگوهای بسیار

الگوریتم	سال انتشار	تعداد اسکن پایگاه داده	شامل آیتم های	اندازه درخت نهایی	متدبازسازی
CanTree	۲۰۰۷	۱	پرتکرار و غیرپرتکرار	غیرفشرده	ندارد
FUFP	۲۰۰۸	در بدترین حالت ۲	پرتکرار و غیرپرتکرار	غیرفشرده	ندارد
CP-tree	۲۰۰۹	۱	پرتکرار و غیرپرتکرار	فشرده	دارد
IP-tree	۲۰۱۰	۱	پرتکرار و غیرپرتکرار	فشرده	دارد
SPO-tree	۲۰۱۱	۱	پرتکرار و غیرپرتکرار	فشرده	دارد
DFP-tree	۲۰۱۳	۱	پرتکرار و غیرپرتکرار	فشرده	دارد

زمانبری دارد. الگوریتم های CP-tree و SPO-tree ممکن است بازسازی های متعددی داشته باشند که پروسه ی ساخت درخت را بسیار زمانبر میکند. در عوض الگوریتم های IP-tree و DFP-tree تنها با یک بازسازی، درخت فشرده ای ایجاد میکنند که زمان ساخت درخت را کاهش میدهد. هر چند IP-tree به دلیلی اینکه قبل از بازسازی بر اساس متد CanTree درخت را میسازد؛ درخت ساخته شده ی آن قبل از بازسازی به حافظه ی زیادی برای نگهداری نودها نیاز دارد.

مراجع

[1] He, Zengyou, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. "*A frequent pattern discovery method for outlier detection*." In International Conference on Web-Age Information Management, pp. 726-732. Springer Berlin Heidelberg, 2004.

[2] Yu, Jeffrey Xu, Zhihong Chong, Hongjun Lu, Zhenjie Zhang, and Aoying Zhou. "*A false negative approach to mining frequent itemsets from high speed transactional data streams.*" Information Sciences, vol. 176, no. 14 pp. 1986-2015, 2006.

[3] Lee, A.J. and C.-S. Wang, "*An efficient algorithm for mining frequent inter-transaction patterns",* Information Sciences, vol. 177, no.17, pp. 3453-3476, 2007.

[4] Hu, T., et al., "*Discovery of maximum length frequent itemsets",* Information Sciences, vol. 178, no. 1, pp. 69-87, 2008.

[5] Lee, A.J., Y.A. Chen, and W.C. Ip, "*Mining frequent trajectory patterns in spatial–temporal databases",* Information Sciences, vol. 179, no. 13, pp. 2218-2231, 2009.

[6] Tanbeer, S.K ,.et al., '*Sliding window-based frequent pattern mining over data streams",* Information Sciences, vol. 179, no. 22, pp. 3843-3865, 2009.

[7] Yun, U. and K.H. Ryu, "*Approximate weighted frequent pattern mining with/without noisy environment",* Knowledge-Based Systems, vol. 24, no. 1, pp. 73-82, 2011.

[8] Lee, G., U. Yun, and K.H. Ryu, "*Sliding window based weighted maximal frequent pattern mining over data streams",* Expert Systems with Applications, vol. 41, no. 2, pp. 694-708, 2014.

[9] Xun, Y., J. Zhang, and X. Qin, "*Fidoop: Parallel mining of frequent itemsets using mapreduce",* IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, no. 3, pp. 313-325, 2016.

[10] Agrawal, R. and R. Srikant, "*Fast algorithms for mining association rules",* in *Proc. 20th int. conf. very large data bases, VLDB*. 1994.

[11] Han, J., J. Pei, and Y. Yin, "*Mining frequent patterns without candidate generation",* in *ACM Sigmod Record*. 2000.

[12] Leung, C.K.S., et al., "*CanTree: a canonical-order tree for incremental frequent-pattern mining",* Knowledge and Information Systems, vol. 11, no.3, pp. 287-311, 2007.

[13] Hong, T.-P., C.-W. Lin, and Y.L. Wu, "*Incrementally fast updated frequent pattern trees",* Expert Systems with Applications, vol. 34, no. 4, pp. 2424-2435, 2008.

[14] Tanbeer, S.K., et al., "*Efficient single-pass frequent pattern mining using a prefix-tree",* Information Sciences, vol. 179, no. 5, pp. 559-583, 2009.

[15] Zhou, M. and T. Wang, "*Improved pattern tree for incremental frequent-pattern mining",* Transactions of Tianjin University,vol. **16**, pp. 129-134, 2010.

[16] Koh, Y.S. and G. Dobbie, "*SPO-tree :efficient single pass ordered incremental pattern mining",* in *International Conference on Data Warehousing and Knowledge Discovery*, Springer, 2011.

[17] Hamedanian, M., M. Nadimi, and M. Naderi, "*An Efficient Prefix Tree for Incremental Frequent Pattern Mining",* International Journal of Information, 2013.