باسمه تعالی

نگاهی بر داده کاوی و کشف قوانین وابستگی

چکیده:

با افزایش سیستمهای کامپیوتر و گسترش تکنولوژی اطلاعات , بحث اصلی در علم کامپیوتر از چگونگی جمع آوری اطلاعات به نحوه استفاده از اطلاعات منتقل شده است . سیستمهای داده کاوی ,این امکان را به کاربر می دهند که بتواند انبوه داده های جمع آوری شده را تفسیر کنند و دانش نهفته در آن را استخراج نمایند .

داده کاوی به هر نوع کشف دانش و یا الگوی پنهان در پایگاه داده ها اطلاق می شود . امروزه داده کاوی به عنوان یکی از مهمترین مسائل هوش مصنوعی و پایگاه داده ، محققان بسیاری را به خود جذب کرده است . در این تحقیق ابتدا نگاه کلی بر داده کاوی ، استراتژیهای داده کاوی و... داریم ، سپس مسأله کشف قوانین وابستگی در پایگاه داده را به تفضیل بررسی کردیم و نگاهی به الگوریتمهای موجود برای آن داشتیم . سپس مسأله کشف قوانین وابستگی در پایگاه داده های پویا را مورد بحث قرار دادیم و الگوریتم های ارائه شده مربوطه را مطرح کردیم .

مقدمه :

هدف از این ارائه و تحقیق بررسی روشهای مطرح داده کاوی است .داده کاوی هر نوع استخراج دانش و یا الگواز داده های موجود در پایگاه داده است که این دانشها و الگوها ضمنی و مستتر در داده ها هستند ,از داده کاوی می توان جهت امور رده بندی (Classification) و تخمین (Estimation) ,پیش بینی (Prediction) و خوشه بندی (Clustering)استفاده کرد .داده کاوی دارای محاسن فراوانی است . از مهمترین آن محاسن کشف کردن دانش نهفته در سیستم است که به شناخت بهتر سیستم کمک می کند .به عنوان مثال می توان به استفاده ترکیبی از روش خوشه بندی جهت تخصیص بودجه به دسته های مختلف از کتب اشاره کرد .

سیستمهای داده کاوی تقریبا از اوایل دهه ۱۹۹۰ مورد توجه قرار گرفتند . علت این امر نیز آن بود که تا آن زمان سازمانها بیشتر در پی ایجاد سیستمهای عملیاتی کامپیوتری بودند که به وسیله آنها بتوانند داده های موجود در سازمان خود را سازماندهی کنند . پس از ایجاد این سیستمها ,روزانه حجم زیادی از اطلاعات جمع آوری میشد که تفسیر کردن آنها از عهده انسان خارج بود . به همین دلیل , نیاز به تکنیکی بود که از میان انبوه داده معنی استخراج کند و داده کاوی به همین منظور ایجاد و رشد یافت .

بنابر این هدف اصلی از داده کاوی ,کشف دانش نهفته در محیط مورد بررسی است که این دانش می تواند شکلهای گوناگونی داسته باشد . دانش استخراج شده می تواند به فرم الگوهای موجود در داده ها باشد که کشف این الگوها منجر به شناخت بهتر

سیستم نیز می شود . الگوهای استخراجی عموما بیانگر روابط بین ویژگیهای سیستم هستند بعنوان مثال در سیستم تجاری یک الگو می تواند بیانگر رابطه بین نوع کالا و میزان تقاضای آن باشد .

در این تحقیق داده کاوی مورد بحث قرار می گیرد . علل استفاده از داده کاوی و منابعی که داده کاوی بر روی آنها اعمال می شود ,علاوه بر این خلاصه ای از روشهای رایج داده کاوی ارائه شده است . تکنیکهای داده کاوی و قوانین وابستگی و الگوریتمهای موجود (Apriori , Aprior TID, Partition, Eclat ,Max Eclat , Vector) و الگوریتم با ساختار Trieو fp grow و الگوریتمهای کاهشی مورد بررسی قرار می گیرند و در هر مورد مثالها ,موارد کاربرد ,تکنیکها و نقاط قوت و ضعف مورد بررسی قرار گرفته اند .

Data mining (داده کاوی)

تعریف :

Data Mining represents a process developed to examine large amounts of data routinely collected. The term also refers to a collection of tools used to perform the

process. Data mining is used in most areas where data are collected-marketing, health, communications, etc.

داده کاوی فرآیند بکارگیری یک یا چند تکنیک آموزش کامپیوتر، برای تحلیل و استخراج داده های یک پایگاه داده می باشد.در واقع هدف داده کاوی یافتن الگوهایی در داده هاست.

دانش کسب شده از فرآیند داده کاوی بصورت مدل یا تعمیمی از داده ها نشان داده می شود.

چندین روش داده کاوی وجود دارد با این وجود همه روشها " آموزش بر مبنای استنتاج " را بکار می برند.

آموزش بر مبنای استنتاج، فرآیند شکل گیری تعاریف مفهوم عمومی از طریق مشاهده مثالهای خاص از مفاهیمی که آموزش داده شده اند، است.

مثال زیر نمونه ای از دانش بدست امده از طریق فرایند اموزش بر مبنای استنتاج است:

آیا تا کنون فکر کرده اید، فروشگاههای بزرگ اینترنتی در mail های خود به مشتریان از چه تبلیغاتی استفاده می کنند؟ و آیا این تبلیغات برای همه مشتریان یکسان است؟

پاسخ این است که از روی دانش کسب شده از اطلاعات خرید افراد و نتیجه گیری از این دانش، این کار را انجام می دهند.مثلا در نظر بگیرید یک قانون در پایگاه داده بصورت زیر استخراج می شود:

دقت = ٪۸۰ : سیگار می خرند ^ نان می خرند ← کسانی که شیر می خرند

از روی این قانون فروشگاه می تواند به تمام کسانی که شیر می خرند تبلیغات سیگار و انواع نان را نیز بفرستد.همچنین این قانون در چیدن قفسه های فروشگاه نیز بی تاثیر نخواهد بود.

{شیر و نان و سیگار در قفسه های کنار هم چیده شوند}

کشف دانش در پایگاه داده [1]

KDD یا کشف دانش در پایگاه داده اصطلاحی است که مکررا بجای داده کاوی بکار می رود. از نظر تکنیکی، KDD کاربردی از روشهای علمی داده کاوی است.

بعلاوه برای انجام داده کاوی فرایند KDD شامل :

۱- یک روش برای تهیه داده ها و استخراج داده ها ،

۲- تصمیم گیری درباره عملی که پس از داده کاوی باید انجام شود ، می باشد.

[1].Khowledge Discovery in DB (KDD)

آیا داده کاوی برای حل مسائل ما مناسب است؟

تصمیم گیری در مورد اینکه آیا داده کاوی را به عنوان استراتژی حل مساله بکار ببریم یا نه، یک مساله دشوار است.

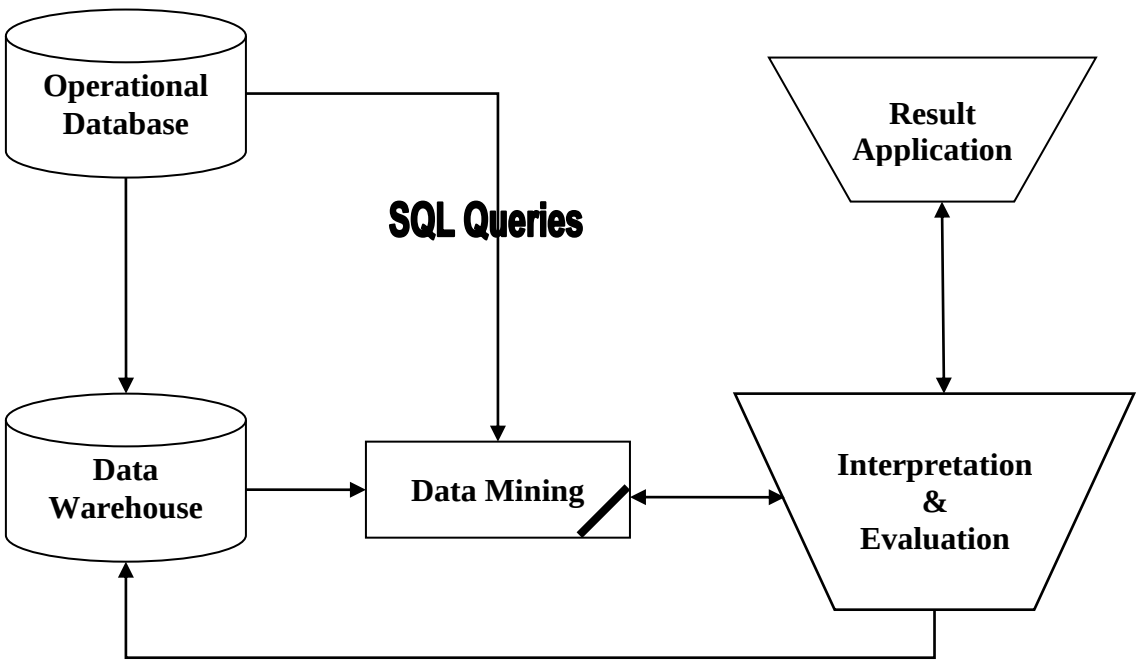
اما به عنوان نقطه شروع چهار سؤال عمومی را باید در نظر بگیریم :

۱. آیا به وضوح می توانیم مساله را تعریف کنیم ؟
۲. آیا بطور بالقوه داده با معنی وجود دارد ؟
۳. آیا داده ها شامل " دانش پنهان" هستند یا فقط برای هدف گزارشگری مناسبند ؟
۴. آیا هزینه پردازش داده (برای داده کاوی) کمتر از سود حاصل از دانش پنهان بدست آمده از پروژه داده کاوی است ؟

یک مدل پردازش داده کاوی ساده :

در یک دید کلی ، ما می توانیم داده کاوی را به عنوان یک فرآیند چهار مرحله ای تعریف کنیم :

۱. جمع آوری یک مجموعه از داده ها برای تحلیل
۲. ارائه این داده ها به برنامه نرم افزاری داده کاوی
۳. تفسیر نتایج
۴. بکارگیری نتایج برای مساله یا موقعیتهای جدید



شکل فوق یک دیاگرام از فرآیند داده کاوی را نشان می دهد.

- جمع آوری داده ها :

فرآیند داده کاوی احتیاج به دسترسی به داده ها دارد. داده ممکن است در تعدادی رکورد، در چندین فایل پایگاه داده ذخیره شود و یا ممکن است داده فقط شامل چند صد رکورد در یک فایل ساده باشد.

با توجه به اینکه معمولا داده های واقعی شامل چندین هزار رکورد می باشند، اولین گام در داده کاوی تهیه زیر مجموعه مناسبی از داده برای پردازش است. گاهی این مرحله احتیاج به تلاش انسانهای بسیاری دارد. در کل سه راه متداول برای دستیابی فرآیند داده کاوی به داده وجود دارد :

۱. ذخیره داده در " انبار داده [۱] "
۲. ذخیره داده در پایگاه داده رابطه ای
۳. ذخیره داده در فایل ساده

- داده کاوی :

همانطور که در شکل مشخص است مرحله بعد داده کاوی است. با این حال قبل از ارائه داده به ابزار داده کاوی ، چندین انتخاب داریم:

۱. یادگیری باید تحت کنترل باشد یا بدون کنترل ؟
۲. کدام نمونه ها در داده ها ی جمع آوری شده برای ساخت مدل بکار میروند و کدامها برای تست مدل ؟
۳. کدام صفتها از صفتهای موجود انتخاب می شوند ؟

و ....

- تفسیر نتایج :

در این مرحله خروجیهای مرحله داده کاوی آزمایش می شوند تا مشخص شود که آیا این نتایج قابل استفاده و جالب هستند یا نه؟ همانطور که در شکل می بینیم اگر نتایج بهینه نباشد می توانیم فرآیند داده کاوی را با صفات و نمونه های جدید تکرار کنیم. همچنین ما می توانیم به " انبار داده " مراجعه کنیم و فرآیند استخراج دانش را تکرار کنیم.

ـ بکارگیری نتایج :

[1] . Data Warehouse

هدف نهایی ما بکارگیری نتایج برای موقعیتهای جدید است. به عنوان مثال دانشی که در یک پایگاه داده فروشگاه بیان می کند کسانی که مجله ورزشی می خرند همچنین سیگار هم می خرند؛ در شکل گیری استراتژیهای فروشگاه در چیدن قفسه ها ، تهیه کاتالوگ ها و ... تاثیر می گذارد.

استراتژیهای داده کاوی :

همانطور که در شکل زیر می بینیم استراتژیهای داده کاوی بطور کلی می توانند به دو دسته " تحت کنترل " یا " بدون کنترل " تقسیم می شوند. آموزش تحت کنترل مدلهایی را با بکارگیری صفات ورودی برای تشخیص مقدار صفت خروجی می سازد. حتی برخی از الگوریتمهای " آموزش تحت کنترل" امکان تشخیص چندین صفت خروجی را به ما می دهند. به صفات خروجی ، صفات وابسته نیز

می گوییم. زیرا مقدار آنها به مقدار یک یا چند صفت ورودی بستگی دارد. به همین ترتیب به صفات ورودی، صفات مستقل نیز می گوییم.

هنگامی که " آموزش بدون کنترل " را بکار می بریم تمامی صفات ورودی هستند و صفت خروجی نداریم.

آموزش تحت کنترل با توجه به اینکه صفات خروجی مقوله ای هستند یا عددی و آیا مدلهای ایجاد شده برای مشخص کردن موقعیت کنونی ایجاد شدند یا پیش بینی خروجیهای آینده ، به چندین قسمت تقسیم می شوند. (منظور از صفات مقوله ای ، صفاتی هستند که مقدار آنها تعداد محدود و مشخصی است، مثل صفاتی که مقدار آنها Boolean است که دو مقدار {true, false} دارد).

طبقه بندی[1] :

طبقه بندی احتمالا از همه استراتژیهای داده کاوی قابل درک تر است. طبقه بندی سه خصوصیت دارد :

۱. آموزش تحت کنترل است.
۲. متغیر وابسته ، مقوله ای است.
۳. تاکید بر روی ساخت مدلهایی است که قادر به اختصاص نمونه های جدید به یکی از کلاسهای تعریف شده باشند.

تخمین[2] :

[1] . Classification
[2] . Estimation

مشابه طبقه بندی ، هدف یک مدل تخمین نیز مشخص کردن مقدار برای یک صفت خروجی است؛ اما بر خلاف طبقه بندی صفات خروجی برای مساله تخمین، عددی است بجای مقوله ای .

بعنوان یک مثال برای تخمین ، پایگاه داده ای را در نظر بگیرید که هر رکورد آن اطلاعاتی را راجع به شخصی دارد مثل : محل زندگی، غذای روزانه در اغلب روزها، نوع ماشین شخصی ، درآمد ماهانه و ....

هدف الگوریتم تخمین در این مثال ایجاد مدلی برای تشخیص درآمد ماهانه نمونه های جدید (رکوردهای جدید) می باشد.{که بقیه صفات آنها بجز درآمد ماهانه مشخص است}.

بیشترتکنیکهای تحت کنترل قادرند که یا مسائل طبقه بندی را حل کنند یا تخمین ، اما نه هردورا.

پیش گویی Perdiction :

تشخیص تفاوت بین پیش گویی و طبقه بند ی یا تخمین کار ساده ای نیست. با این حال هدف یک مدل پیش گویی ، برخلاف طبقه بندی یا تخمین، بجای مشخص کردن رفتار کنونی، مشخص کردن خروجیهای آینده است. بیشتر روشهای داده کاوی که برای طبقه بندی یا تخمین مناسبند، برای ساخت مدلهای پیش گویی نیز بکار میروند. عملا این طبیعت داده است که مشخص می کند یک مدل برای تخمین مناست است یا طبقه بندی ویا پیش گویی.

دسته بندی بدون کنترل Unsupervised Clustering:

در دسته بندی بدون کنترل، ما دیگر صفات خروجی نداریم که ما را در فرآیند یادگیری راهنمایی کند، در عوض برنامه مربوطه ساختارهای دانش را با بکارگیری معیارهای " کیفیت دسته " برای گروه بندی داده ها به دو یا چند کلاس (دسته)، بدست می آورد.

یک هدف اساسی دسته بندی بدون کنترل، کشف ساختارهای مفهومی در داده است.

کاربردهای متداول دسته بندی بدون نظارت عبارتند از :

- مشخص می کند که آیا ارتباطات با معنی در شکل مفاهیم می تواند در داده ما پیدا شود یا نه ؟
- کارآیی روش آموزش تحت کنترل را مشخص می کند.
- بهترین صفات ورودی برای آموزش تحت کنترل را مشخص می کند.
- شناسایی از حد خارج شده ها (outlier)

تحلیل سبد بازاری Market Basket Analyse :

هدف این مرحله پیدا کردن ارتباطات جالب میان محصولات (خرده فروشی) است. خروجی این مرحله به فروشندگان کمک می کند تا بهتر بتوانند قفسه ها را بچینند یا کاتالوگها را تنظیم کنندو نیز در ایجاد استراتژیهای فروشگاه نیز کارا است.

مثالی از دانش این مرحله به فرم زیر است (در یک فروشگاه)

سیگار می خرند ⟵ کسانی که قهوه می خرند

تکنیکهای داده کاوی تحت کنترلSupervised Data Mining:

تکنیکهای داده کاوی برای بکارگیری استراتژی داده کاوی برای یک مجموعه داده بکار می رود. یک تکنیک داده کاوی از دو قسمت تشکیل شده است:

۱. الگوریتم.

۲. ساختار دانش مربوطه مثل درخت یا یک مجموعه قوانین درخت تصمیم که در قسمتهای قبلی توضیح دادیم.

در اینجا چندین روش دیگر برای داده کاوی نظارت شده ارائه می دهیم :

۱. شبکه عصبی :

یک شبکه عصبی مجموعه ای از نودهای به هم پیوسته است که طراحی می شوند تا رفتار مغز انسان را شبیه سازی کنند.

چون مغز انسان از بیلیونها عصب تشکیل شده و شبکه های عصبی کمتر از صد نود دارند مقایسه یک شبکه عصبی و رفتار مغز کمی غیر متعارف است. با این وجود شبکه های عصبی با موفقیت ، برای حل مسائل بکار برده می شوندو برای داده کاوی نیز کاملا ابزار مناسبی است .

شبکه های عصبی در شکلها و فرمهای گوناگونی وجود دارند و هم برای آموزش تحت کنترل و هم دسته بندی بدون کنترل بکار می روند. درهمه موارد ، مقادیر ورودی برای شبکه عصبی باید عددی باشند. شبکه feed-forward یک نوع شبکه عصبی مناسب برای مسائل آموزش تحت کنترل می باشد.

۲. برگشت آماری[1] :

برگشت آماری یکی از روشهای آموزش تحت کنترل است که یک مجموعه از داده های عددی را توسط ایجاد معادلات ریاضی مرتبط با یک یا چند صفت ورودی به یک صفت خروجی عددی نسبت می دهد.

[1] . Statistical Regression

یک مدل " برگشت خطی " توسط یک صفت خروجی که مقدارش بوسیله :

" جمع مقادیر صفت های ورودی × یک وزن مشخص " مشخص می شود.

مثلا اگر یک پایگاه داده شامل صفات ورودی A , B, C , D و صفت خروجی E باشد، رابطه زیر

می تواند یک مدل برگشت خطی باشد :

$$E = 0.5\ C - 0.2\ B + A + 0.32$$

میبینیم که E صفت خروجی است که مقدارش توسط ترکیب خطی صفات A , B , C تعیین می گردد.

همانند شبکه عصبی ، در این روش نیز همه ورودیها باید عددی باشند و در صورتیکه داده ها در پایگاه داده مقوله ای باشند باید آنها را به داده های عددی تبدیل کنیم.

۳. قوانین وابستگی[۲] :

به تفصیل در بخشهای بعد مورد بحث قرار می گیرد.

قوانین پیوستگی:

یکی از مهمترین بخشهای داده کاوی، کشف قوانین وابستگی در پایگاه داده است.این قوانین، لزوم وقوع برخی صفات(آیتم ها) را در صورت وقوع برخی دیگر از آیتمها، تضمین می کند.

برای روشن شدن مطلب یک فروشگاه خرده فروشی را در نظر بگیرید. مشخصات اجناس خریده شده توسط هر مشتری در یک رکورد پایگاه داده ذخیره می شود.به هر رکورد یک شناسه (TID) نسبت داده می شود.فرض کنید که مجموعه I شامل تمام آیتمها(اجناس) فروشگاه باشد. اگر $x,y \subset I$ و $x \cap y = \emptyset$ آنگاه $x => y$ یک قانون وابستگی است که بیان میکند اگریک مشتری اجناس مجموعه x را بخرد، اجناس مجموعه y را هم می خرد. این چنین قوانین، تأثیر مهمی در تایین استراتژیهای فروش، بخش بندی مشتریان، تنظیم کاتالوگها و... دارد. همچنین کشف قوانین وابستگی، کاربردهای بسیاری در علوم مختلف نیز دارد.

تعریف مسأله:

مجموعه آیتم: به هر زیر مجموعه از مجموعه آیتمها (I) ' یک مجموعه آیتم ' میگوییم.

در بیشتر الگوریتمها مساله کشف قوانین پیوستگی به دو زیر مساله تقسیم می شود:

۱.پیدا کردن تمامی زیر مجموعه های مجموعه I [مجموعه آیتمها] که تکرار (وقوع) آنها در پایگاه بیشتر از یک حد تایین شده است.

به مجموعه آیتمهایی که تعداد وقوع آنها در پایگاه بزرگتر(یا مساوی)حد تایین شده است

---

[۲] . Association Rules

' مجموعه آیتمهای بزرگ'، وبه بقیه' مجموعه آیتمهای کوچک' می گوییم.

۲.بکارگیری مجموعه آیتمهای بزرگ برای تولید قوانین مطلوب.

تعریف:

پوشش[2]: مجموعه I شامل تمام آیتمها و مجموعه آیتم x را در نظر بگیرید ، می گوییم پوشش x در پایگاه داده برابر است با ℓ اگر و فقط اگر تعداد وقوع مجموعه آیتم x در پایگاه داده برابر با ℓ باشد.

Support(x)=ℓ

درجه اطمینان [3]:مجموعه I شامل تمامی اقلام و مجموعه آیتمهای x و y مفروضند. درجه اطمینان قانون x=>yبرابر است با : x∩y=ø

$$Conf(x=>y) = \frac{support(xUy)}{support(x)}$$

الگوریتم Apriori : این الگوریتم(Agrawal & Srikant ,1994) برای تولید مجموعه اقلام بزرگ به این ترتیب عمل می کند:

ابتدا با یک دور خواندن پایگاه داده مجموعه اقلام بزرگ یک عضوی (1-itemset)را مشخص می کنیم.[مجموعه اقلام ۱ عضوی که تعداد تکرار آنها در DB از حد تایین شده(minsup) بیشتر است.]

سپس با استفاده ازمجموعه اقلام بزرگ یک عضوی، مجموعه اقلام دو عضوی را ایجاد می کنیم و برای تایین پوشش مجموعه اقلام دو عضوی یک بار دیگر کل پایگاه داده را می خوانیم تا مجموعه اقلام بزرگ دو عضوی را تایین کنیم.

به همین ترتیب با استفاده از مجموعه اقلام بزرگ دو عضوی مجموعه اقلام سه عضوی را ایجاد کرده و با خواندن دوباره پایگاه داده پوشش هر مجموعه قلم سه عضوی را مشخص کرده و مجموعه اقلام بزرگ سه عضوی تولید می شوند و این کار را برای مجموعه های ۴عضوی و ... انجام میدهیم تا مرحله ای که هیچ مجموعه آیتم بزر الگوریتم:

```
L1= { larg-1-itemset }
for ( k=2; Lk-1 ≠0;k+1 ) do
 begin
    C k=apriori – gen(Lk-1 ) // ساخت زیر مجموعه های k عضوی با استفاده از اقلام بزرگ k-۱ عضوی
    for all transaction lεD do
      begin
            C t=subset(Ck,l);// تست اینکها کدام مجموعه آیتمهای k عضوی در تراکنش رخ دادند.
             for all candidate cεCt do
                                c.count++;
      end
   Lk={ cεCk l c.count ≥minsup}
end;
Answer=Uk Lk
```

---

[2] .support
[3] .confidence

(تبصره : اگر یک مجموعه آیتم بزرگ باشد[تکرارش درپایگاه داده بیشتر از minsupباشد] تمامی زیرمجموعه های آن نیز بزرگ هستند.)

چون هدف، تولید مجموعه اقلام بزرگ است، در الگوریتم apriori در هر مرحله پس از ایجاد مجموعه اقلامk عضوی از مجموعه اقلام بزرگ k-1 عضوی قبل از خواندن پایگاه داده برای تشخیص پوشش مجموعه اقلام k عضوی، ابتدا باید برای هر مجموعه قلم ببینیم آیا زیر مجموعه k-1 عضوی اش بزرگ هستند یا نه، اگر حتی یک زیر مجموعه k-1 عضوی اش هم بزرگ نباشد، آن مجموعه قلم k عضوی نمی تواند بزرگ باشد.(طبق قضیه) و آن را حذف می کنیم.

برای روشن تر شدن الگوریتم به مثال زیر توجه کنید:

minsup=3

Database:

TID	Items
100	1 3 4 5
200	2 3 5
300	1 2 3 5
400	2 5
500	1 3 4 5

گام۱:ایجاد مجموعه اقلام ۱ عضوی:

L 1 مجموعه آیتمهای بزرگ:

{1}=3
{2}=3
{3}=4
{5}=5

مجموعه آیتمهای ۱ عضوی:

{1}=3
{2}=3
{3}=4
{4}=2
{5}=4

گام۲: ایجاد مجموعه آیتمهای دو عضوی با استفاده از مجموعه آیتمهای بزرگ ۱ عضوی:

L2مجموعه آیتمهای بزرگ دو عضوی:

{1,3}=3
{2,5}=3
{3,5}=3
{1,5}=3

مجموعه آیتمهای ۲ عضوی:

{1,2}=1
{1,3}=3
{1,5}=3
{2,3}=2
{2,5}=۳
{3,5}=4

گام۳:ایجاد مجموعه آیتمهای سه عضوی با استفاده از مجموعه آیتمهای بزرگ دو عضوی:

L3مجموعه آیتمهای بزرگ سه عضوی:

{1,3,5}=3

مجموعه آیتمهای ۳عضوی:

{1,3,5}=3

Answer=L1 U L2 U L3={{1} {2} {3} {5} {1,3} {2,5} {3,5} {1,5} {1,3,5}}

الگوریتم Aprior TID :

در این الگوریتم (Agrawal & Srikant ,1994)در ابتدا فضای جستجو پایگاه داده اولیه است که هر تراکنش آن را به عنوان مجموعه ای از مجموعه قلم های تک عضوی می بینیم:

به این معنی که تراکنش (100 l 1,2,3) به صورت (100 l{1}{2}{3})در نظر گرفته می شود

سپس همانند الگوریتم apriori مجموعه اقلام ۱ عضوی را ایجاد کرده و تعداد تکرار آنها را در پایگاه می شماریم و مجموعه اقلام بزرگ ۱ عضوی را مشخص می کنیم.

همانند الگوریتمapriori با استفاده ازعناصر مجموعه آیتمهای ۱عضوی بزرگ، مجموعه آیتمهای دو عضوی را ایجاد می کنیم.(چون هر مجموعه آیتم k عضوی از ترکیب دو مجموعه k-1 عضوی تشکیل شده است برای شمردن تعداد تکرار یک مجموعه kعضوی در پایگاه می توان در هر تراکنش به دنبال مجموعه آیتمهای k-1 عضوی تولید کننده آن مجموعه آیتم، گشت. یعنی مثلا برای شمردن تکرار مجموعه آیتم {1,2,3}در هر تراکنش باید ببینیم آیا (1,2)و(1,3)در مجموعه هست یا نه.)

سپس برای هر تراکنش TIDمربوطه را به همراه تمامی مجموعه آیتمهای kعضوی که در تراکنش هست[تولید کننده های k-1عضوی اش در تراکنش هست] به عنوان تراکنش پایگاه جدید اضافه می کنیم.(برای ا ستفاده مجموعه آیتمهای k+1)

$\hat{C}k \rightarrow$ *پایگاه جدید*

وتراکنشهایی که هیچ مجموعه آیتم kعضوی را پوشش ندهند به پایگاه جدید راه پیدا نمی کنند.سپس مجموعه اقلام k+1عضوی را با استفاده از مجموعه اقلام kعضوی بزرگ ایجاد می کنیم ودر پایگاه داده جدید به دنبال تعداد تکرارهای این مجموعه اقلام می گردیم.[در واقع برای هر مجموعه آیتمk+1عضوی در هر تراکنش جستجو می کنیم ببینیم آیا دو مجموعه kعضوی تولید کننده اش در تراکنش است یا نه.]

سپس TIDاین تراکنش به همراه تمامی مجموعه آیتمهای k+1عضوی که پوشش میدهد به پایگاه داده جدید دیگری اضافه می کنیم، $\hat{C}k+1 \rightarrow$ پایگاه داده جدید

و این کار را به ازای تمامی تراکنشها تکرار می کنیم.

عملیات بالا را زمانی که پایگاه داده جدید ایجاد شده ($\hat{C}k$)تهی نباشد ادامه می دهیم.

الگوریتم partition :

این الگوریتم (Savasere &Navathe & Omiecinski,1995) برای بدست آوردن مجموعه آیتمهای بزرگ از دو قسمت تشکیل شده است. در قسمت اول این الگوریتم پایگاه داده را به چندین بخش مجزا از نظر منطقی تقسیم می کند وهمهٔ مجموعه آیتمهای بزرگ محلی (مجموعه آیتمهایی که تعداد تکرار آنها از minsupp محلی که برای هر بخش در نظر گرفته شده بیشتر است) برای هر بخش بطور مجزا تولید می شود .

سپس در انتهای مرحله یک ، همه مجموعه اقلام بزرگ در هر بخش باهم مجموعه اقلام بزرگ بالقوه در سطح کل پایگاه داده را تشکیل می دهند (مجموعه Σ). درقسمت دوم، پوشش (تعداد تکرار) هر مجموعه آیتم در مجموعه اقلام بالقوه (Σ) در کل پایگاه داده شمارش می شود و مجموعه اقلام بزرگ واقعی محاسبه می شوند .

توجه: برای هر بخش یک minsupp محلی در نظر گرفته می شود و یک minsupp کلی برای کل پایگاه داده نیز داریم.

نحوه تولید مجموعه اقلام بزرگ در هر بخش به صورت زیر است:

۱- ابتدا به ازای هر آیتم a در مجموعه I(آیتم ها) ، ID تراکنش هایی در بخش مربوطه از DB که شامل آیتم a می باشند را مشخص می کنیم.

بخش۱ $t(a)=\{100,300\}$ بخش۲ $t(a)=\{400,500,601\}$

سپس همانند الگوریتم apriori، ابتدا مجموعه اقلام یک عضوی بزرگ را ایجاد می کنیم . سپس از روی آنها مجموعه اقلام بزرگ دو عضوی را ایجاد میکنیم و ...

با این تفاوت که در این الگوریتم برای محاسبه پوشش (تعداد وقوع) هر مجموعه آیتم مثل

$A=\{i1,i3,i5\}$ $i1,i3,i5 \in I$

تعداد اعضای مجموعه $t(A)=t(i1)\cap t(i3)\cap t(i5)$ را می شماریم (به جای خواندن دوبارهDB)، که t(x) مجموعه ID تراکنش های شامل x است (در همان بخش).

پس از اینکه تمام مجموعه اقلام بزرگ(مجموعه اقلامی که تعداد تکرار آنها در این بخش از DB بزرگتر از minsupp محلی است) را بدست آوردیم همین کار را برای بخش های دیگر تکرار می کنیم .

در صورتیکه مجموعه اقلام بزرگ برای بخش i را ، $\sum i$ بنامیم، مجموعه $\sum$ به صورت زیر ایجاد می شود:

$$\sum=\sum 1\cup\sum 2\cup\ldots\cup\sum n$$

حال مجموعه Σ شامل مجموعه اقلام بزرگ بالقوه است .اکنون یک بار دیگر پایگاه داده را می خوانیم وپوشش هر کدام از مجموعه اقلام کاندید را در کل پایگاه داده مشخص می کنیم.

در صورتیکه پوشش A ازminsup،عمومی بزرگتر باشد .یک مجموعه اقلام بزرگ است.

تعداد بخش ها در پایگاه به گونه ای تعیین شده که هر بخش به همراه ساختمان داده های مورد نیاز در حافظه جای گیرد .

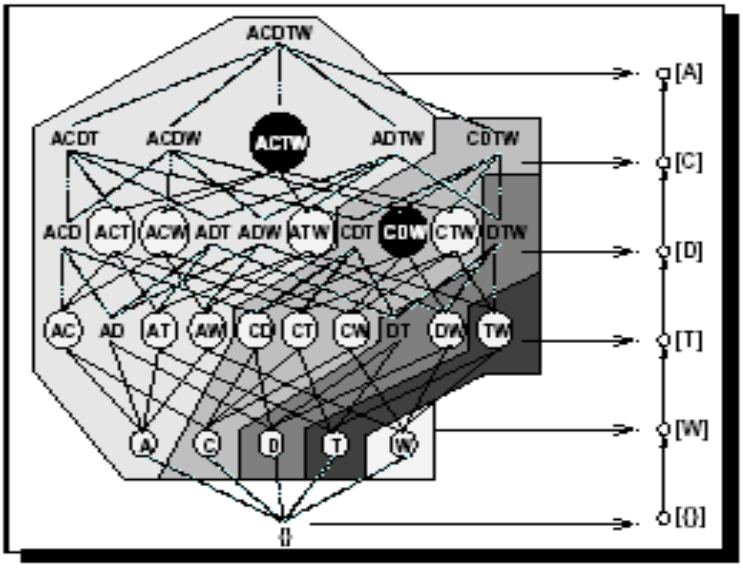
الگوریتم های MaxEclat,Eclat :

در این الگوریتم ها(Zaki,1997) نیز همانند partition ابتدا مجموعه t(ai) به ازای هر ai متعلق I (مجموعه اقلام کلی)تولید می شود.(t(ai) شناسه تراکنشهایی را دارد که شامل aiهستند.)

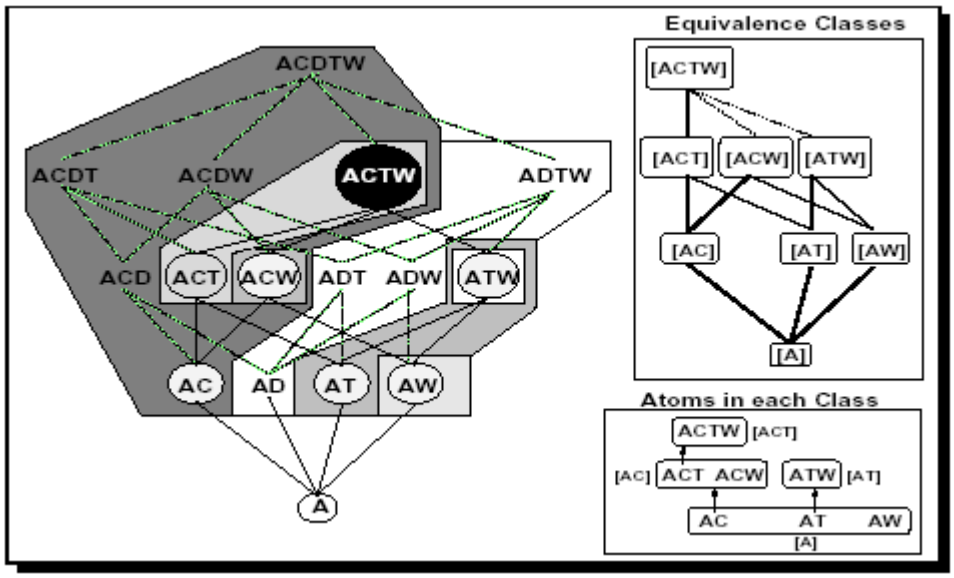
ولذا پس از ایجاد هر مجموعه آیتم k عضوی (از مجموعه های k-1 عضوی) مثل A={a1,a2,a3} برای محاسبه پوشش A در پایگاه داده تعداد اعضای مجموعه $t(A)=t(a1)\cap t(a2)\cap t(a3)$ را شمارش می کنیم.

با توجه به اینکه تعداد مقادیر میانی با انجام این کار (اشتراک ها) بسیار زیاد می شود، فضای حافظه اصلی جوابگو نیست و لذا راه حل زیر ارائه شد.

با توجه به تعاریف ریاضی، مجموعه تمام زیر مجموعه های یک مجموعه (مجموعه توانی) ، یک شبکه را تشکیل میدهند.این الگوریتم ، این شبکه را به چندین زیر شبکه تقسیم میکند و برای هر زیر شبکه بطور مجزا مجموعه آیتمهای بزرگ را تولید می کند. در این صورت ساختمان داده های هر زیر شبکه به راحتی در حافظه اصلی جای می گیرد .



در شکل زیر ، زیر شبکه مربوط به آیتم [A] را مشاهده می کنیم که شامل اعضایی است که با A شروع می شوند. به نودهایی که مستقیما با [A] در ارتباطند 'اتم'ها می گوییم.



درهرزیرشبکه با داشتن اتمهای مربوطه میتوان استراتژیهای پایین به بالا (partition,apriori) ویا بالا به پایین را برای جستجوی مجموعه اقلام بزرگ بکار برد.(در شکل جستجوی پایین به بالا را مشاهده می کنیم).

```
Bottom-Up(S):
for all atoms A_i ∈ S do
    T_i = ∅;
    for all atoms A_j ∈ S, with j > i do
        R = A_i ∪ A_j;
        L(R) = L(A_i) ∩ L(A_j);
        if σ(R) ≥ min_sup then
            T_i = T_i ∪ {R}; F_|R| = F_|R| ∪ {R};
    end
end
for all T_i ≠ ∅ do Bottom-Up(T_i);
```

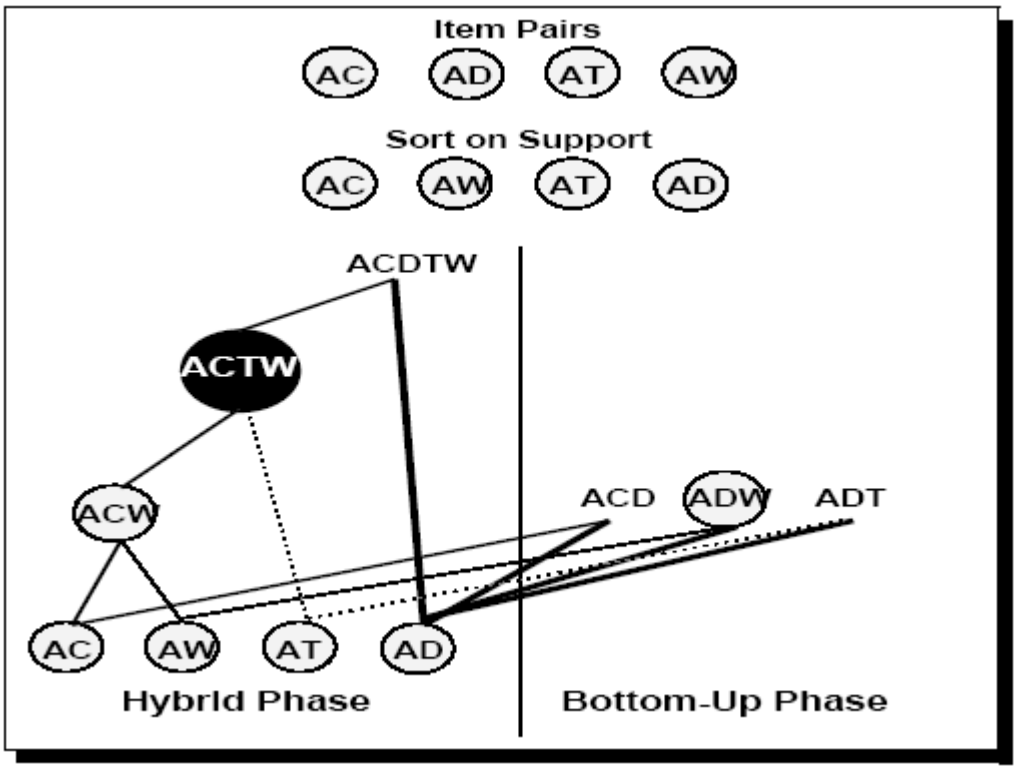
S در الگوریتم فوق مجموعه اتمهای زیر شبکه است و F مجموعه اقلام بزرگ نهایی است و Ti مجموعه اقلام بزرگ جدید هستند.(یک سطح بالاتر).

در روش بالا به پایین در هر زیر شبکه از بزرگترین مجموعه آیتم در آن زیرشبکه شروع می کنیم و به سمت اتمها پیش می رویم. هر گاه یک نود مجموعه قلم بزرگ را مشخص کند دیگر برای آن نود فرزندانش چک نمی شوند زیرا مطابق قضیه قبلی (زیر مجموعه های بزرگ هم بزرگند)، تمام زیر مجموعه هایش مجموعه اتم های بزرگ هستند.

```
Top-Down(S):
R = ⋃{A_i ∈ S};
if R ∉ F_|R| then
    L(R) = ⋂{L(A_i) | A_i ∈ S};
    if σ(R) ≥ min_sup then
        F_|R| = F_|R| ∪ {R};
    else
        for all Y ⊂ R, with |Y| = |R| − 1
            if Y ∉ HT then
                Top-Down({A_j | A_j ∈ Y});
                if σ(Y) < min_sup then HT = HT ∪{Y};
        end
```

Pseudo-code for Top-Down Search

بجز روش بالا به پایین و پایین به بالا روش دیگری به نام hybrid که ترکیب دو روش بالا به پایین و پایین به بالا می باشد نیز وجود دارد که این روش را در شکل زیر نشان دادیم.



ابتداAWو AC ترکیب می شوند(ACW) .اگر(ACW)مجموعه آیتم بزرگ باشد با عنصر اتم بعدی (AT)ترکیب میشود و(ACTW)را ایجاد می کند.دوباره اگر این مجموعه آیتم بزرگ باشد با اتم بعدی (AD)ترکیب میشود و تشکیل (ACDTW) را می دهد و چون این مجموعه آیتم بزرگ نیست مجموعه آیتم قبلی (ACTW)یک مجموعه آیتم maximal است.

حال اگر مجموعه تمام اتم هایی که دراین مرحله شرکت داشتند را S1 بنامیم و بقیه اتم ها را S2 , به ازای اولین عضو S2 مثل ai,ai را با تمام اتم های S1 ترکیب کرده و اتم های جدید را به دست می اوریم و برای آنها جستجوی پایین به بالا را تکرار می کنیم و مجموعه آیتم های بزرگ را به دست می آوریم .سپس ai هم به S1 رفته و از S2 خارج میشود .حال دوباره به ازای اولین عنصر S2 کارهای قبل را تکرار می کنیم.(ترکیب اولین عنصرS2 با اتم های S1 و فراخوانی جستجو پایین به بالا برای آنها)

```
Hybrid(S sorted on support):
R = A_1; S_1 = {A_1};
for all A_i ∈ S, i > 1 do /* Maximal Phase */
   R = R ∪ A_i; L(R) = L(R) ∩ L(A_i);
   if σ(R) ≥ min_sup then
      S_1 = S_1 ∪ {A_i}; F_|R| = F_|R| ∪ {R};
   else break;
end
S_2 = S − S_1;
for all B_i ∈ S_2 do /* Bottom-Up Phase */
   T_i = {X_j | σ(X_j) ≥ min_sup, L(X_j) = L(B_i) ∩ L(A_j), ∀A_j ∈ S_1};
   S_1 = S_1 ∪ {B_i};
   if T_i ≠ ∅ then Bottom-Up(T_i);
end
```

Pseudo-code for Hybrid Search

با توجه به استراتژیهای بالا الگوریتم های زیر را داریم:

Eclat: روش پایین به بالا را برای هر زیر شبکه اعمال می کند.

Maxeclat : روش ترکیب hybridرا برای هرزیر شبکه اعمال می کند.

الگوریتم با ساختار trie :

در این الگوریتم(Amir & Feldman & Kashi,1998) از یک ساختار داده بنام trie برای تولید مجموعه اقلام بزرگ استفاده می کنیم . به کار گیری این ساختارها ما را قادر می سازد تا مجموعه اقلام بزرگ را به طور کارا تولید کنیم.(حتی با وجود minsupp های بسیار کوچک)

رشته S=S[1]...S [n]را در نظر بگیرید برای افزودن این رشته به درخت trie ابتدا از ریشه درخت trie شروع می کنیم (current-node= root) و نگاه می کنیم , بینیم آیا یالی از current-node با بر چسپ S [1] وجود دارد یا خیر .

اگر چنین یالی وجود داشت, از طریق آن یال به نود فرزند می رویم و (نود فرزند=current node) سپس دوباره اعمال بالا را برای current-node جدید و کاراکتر بعدی (S[2]) تکرار می کنیم.

در غیر این صورت (یالی با بر چسب S[1] وجود نداشت), یالی با برچسب S[1] از current-node ایجاد می کنیم و فرزند ایجاد شده از طریق این یال , current-node مرحله بعدی است .

حال باید کار را برای S[2] و current-node جدید تکرار کنیم .

این کار را آنقدر تکرارکنیم تا رشته تمام شود, در صورتیکه تمام رشته را چک کردیم و کار را برای S[n] و current-node مربو طه اش چک کردیم شمارنده مربوط به نود فرزند (Current نود آخر از طریق یال S[n]) رامی افزاییم.

الگوریتم مربوطه:

در این الگوریتم فقط یک دور پایگاه داده را می خوانیم و به ازای هر تراکنش مثل T, تمام زیر مجموعه های T را به درخت (trie) می افزاییم .

مثلا برای {1,2,5} زیر مجموعه های {1},{2},{5},{1,2},{1,5},{2,5},{1,2,5} یکی یکی به درخت trie افزوده می شوند.

در نهایت شمارنده مربوطه به هر نود در درخت (trie) تعداد وقوع مجموعه آیتم نود مربوطه را در بر دارد.

نکته :این الگوریتم برای حالات خاصی که طول هر تراکنش بسیار کم است در نظر گرفته شده است و در هنگامی که طول تراکنش ها کم باشد, می توان نتیجه گرفت که طول مجموعه اقلام بزرگ نیز از یک حدی (طول M) بیشتر نخواهد بود,لذا برای هر تراکنش فقط زیر مجموعه های ۱ تا M عضوی را تولید وبه درخت اضافه می کنیم.

الگوریتم fp-grow :

در این الگوریتم (Han &Pei &Yin,2000)با استفاده از ساختاری فشرده به نام frequent pattern tree(fp-tree) کل فضای پایگاه داده را به فضای کمی در حافظه اصلی نگاشت می کنیم و موقع شمردن پوشش مجموعه آیتم ها درپایگاه داده از fp- tree استفاده می کنیم.به این ترتیب زمان ناشی از خواندن پایگاه داده (DB) در دفعات کاهش می یابد.

تعریف:

۱_ یک fp- tree یک درخت است که ریشه آن null است و یک مجموعه از زیر درخت های پیشوندی آیتم به عنوان فرزندان ریشه هستند, یک جدول سرایند آیتم های بزرگ هم وجود دارد.

۲_هر نود در درخت fp- tree شامل سه مولفه است:

الف: نام آیتم: آ یتمی که نود مشخص می کند.

ب: تعداد : تعداد تراکنش در پایگاه داده, که شامل بخشی از شاخه درخت از ریشه تا آن نود هستند.

ج: link : اشاره گری به نود بعدی در درخت که شامل همین آیتم است , یا مقدارnull دارد.

۳ _ هر ورودی در جدول سرایند از دو فیلد تشکیل شده:

۱-نام آیتم

۲- اشاره گری به اولین نود حامل آیتم

ساخت fp- tree :

برای ساخت fp tree ازپایگاه داده , ابتدا یک دور پایگاه داده را می خوانیم تا پوشش هر آیتم $(\forall a \in I)$ مشخص شود .(Supp (a)) .

سپس قبل از افزودن هر تراکنش به درخت fp tree , مجموعه آیتم های آن تراکنش را, ابتدا بر اساس supp (پوشش) مرتب می کنیم و سپس به درخت می افزاییم.

الگوریتم ساخت fp- tree :

ورودی : یک پایگاه داده شامل تراکنش ها و یک حد مینیمم برای پوشش ($\geq$ یا minsupp)

خروجی : درخت fp tree مربوطه.

الگوریتم :

۱- یکبار پایگاه داده را بخوانید.مجموعه F شامل آیتم های بزرگ(با تکرار کافی در پایگاه داده) و پوشش هر یک را بدست آورید.سپس F مرتب کنید و L بنامید.

۲_ ریشه fp-tree (T)را null بگذارید و برای هر تراکنش در پایگاه داده, به ترتیب زیر عمل کنید:

ابتدا آیتم های بزرگ در تراکنش را انتخاب نمایید و مطابق مجموعه L مرتب کنید(به ترتیب پوشش) فرض کنید مجموعه آیتم های بزرگ مرتب شده حاصل از این تراکنش به صورت [p|P] که p اولین آیتم در این مجموعه است و P بقیه آیتم هاست , تابع insert tree ([p|P] ,T) را فراخوانی کنید.

این تابع به شکل زیر کار میکند که اگر T فرزندی مثل N دارد که

N.itemname=p.itemname آنگاه شمارنده (count) نود N یکی افزایش می یابد و گرنه یک نود جدید N می سازیم و شما رنده آن را یک می کنیم .

Link پدربرای این نود به T اشاره می کندو node-link این نود به نودی با همین item-name در همین درخت اشاره می کندو اگر P تهی نیست insert-tree(P,N) را فراخوانی کنید.

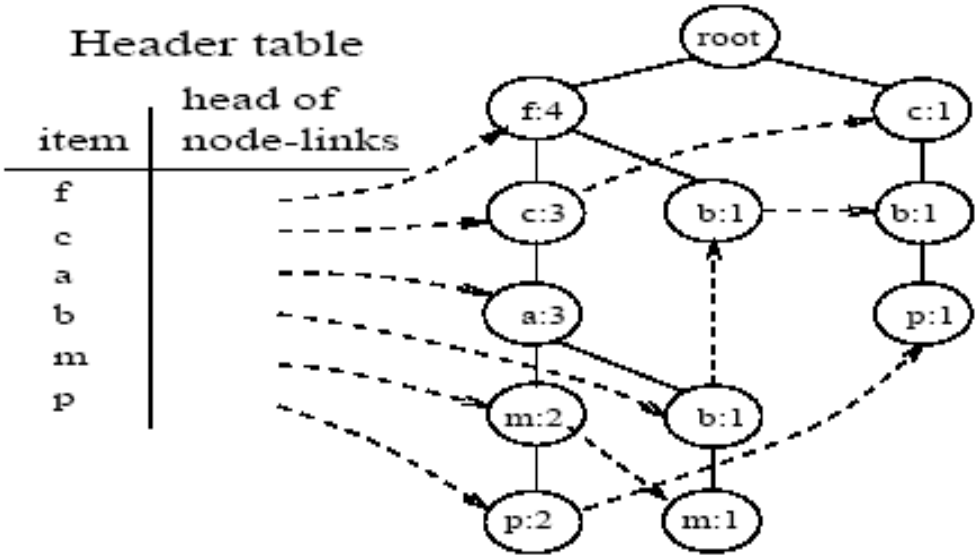
شکل صفحه بعد مثالی را نشان می دهد که fp tree برای جدول ۲ رسم شده است و header table نیز آدرس اولین نود را برای هر آیتم بزرگ را دارد .

الگوی شرطی:

برای روشن مفهوم فوق (الگوی شرطی) شکل را در نظر بگیرید.آیتم p را در نظر بگیرید.مطابق شکل شاخه های پیشوند های p در درخت عبارتند از <f:4,c:3,a:3,m:2> و زیر شاخه <c:1,b:1> . به این معنی که مجموعه آیتم های پیشوندی در پایگاه داده عبارتند از <c:1,b:1>,<f:2,c:2,a:2,m:2> که به این دو الگوی های شرطی میگوییم.

TID	Items Bought	(Ordered) Frequent Items
100	f, a, c, d, g, i, m, p	f, c, a, m, p
200	a, b, c, f, l, m, o	f, c, a, b, m
300	b, f, h, j, o	f, b
400	b, c, k, s, p	c, b, p
500	a, f, c, e, l, p, m, n	f, c, a, m, p

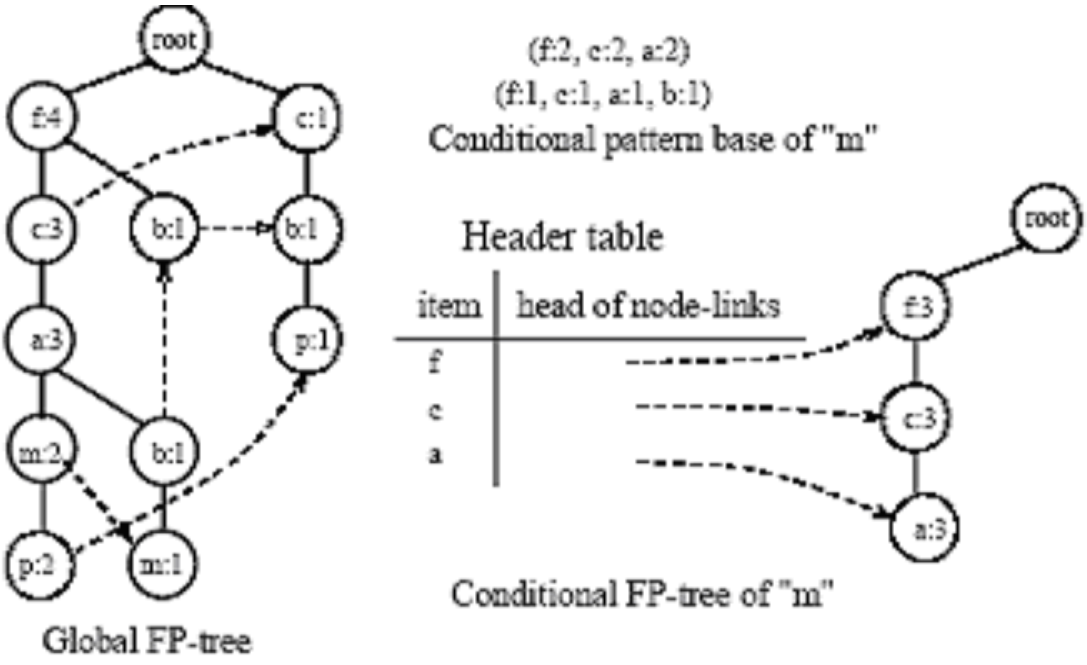
الف



Fp-tree شرطی :

در صورتیکه با الگوهای شرطی, از نو یک fp-tree درست کنیم و نود مربوط به آیتم هایی که تعداد تکرار کمتر از Σ دارند را حذف کنیم, درخت حاصل fp-tree شرطی نامیده میشود.و به صورت(fp-tree|p) نمایش داده می شود (که p آیتم مربوطه است)

(fp-tree|m) در شکل زیر مشخص است.



A conditional FP-tree built for m, i.e, "FP-tree | m"

الگوریتم ایجاد مجموعه آیتم های بزرگ بوسیله درخت fp-tree :

روش کار به این صورت است که ابتدا به ازای هر آیتم بزرگ مثل{a} کارهای زیر را تکرار می کنیم :

۱- temp =λ

۲-اگر $\sup(a) > \Sigma$,a∪temp را به مجموعه آیتم های بزرگ اضافه کن .

T1=(fp-tree|a) را در درخت ایجاد میکنیم و(temp=a) .

دوباره درداخل این fp-tree جدید (T1) به ازای هرآیتم بزرگ دراین fp-tree مثل 'a در صورتیکه $\sup(a') > \Sigma$ (تکراردرخت T1) ,temp ∪{a'} به مجموعه آیتم های بزرگ اضافه می شودو دوباره همین کار را برای (T2 =(T1|a'حاصل از درخت T1 تکرار می کنیم.

(مقدار temp قبل از فراخوانی تابع (T1|a') برابرست با :temp=a' ∪ temp')

الگوریتم برداری :

برای هر آیتم یک بردار مشخصات (ویژگی) فشرده به همراه یک رکورد ویژگی ساخته می‌شود. این ساختار فقط یکبار در هنگامی که پایگاه داده برای اولین بار خوانده می‌شود، ساخته می‌شود. سپس برای پوشش محاسبه هر مجموعه آیتم, از بردارهای فوق(به جای پایگاه داده) استفاده میشود.

لذا دسترسی مجدد به پایگاه داده اولیه نیازی نیست و محاسبات نیز بسیار سریعتر از الگوریتمهای قبل می‌باشد.

الگوریتم ارائه شده:

ایده‌ی این روش، استخراج ویژگی‌های آیتم‌های مشخص شده[1] و ذخیره آنها در یک بردار فشرده است. پس از اینکه بردار فشرده این آیتمها ساخته شود همه محاسبات برای مشخص شدن پوشش (تعداد تکرار در پایگاه داده) هر مجموعه آیتم از طریق عملیا ت AND بین بردارها صورت می‌گیرد بجای دسترسی مجدد به پایگاه داده.

(بردار فشرده‌ی یک آیتم مشخص می‌کند که این آیتم در کدام تراکنش‌ها رخ داده است)

در الگوریتم(Tseng,2000) دو پایگاه داده داریم بنامهای Support DB و feature DB:

feature DB : بردارهای مربوط به آیتم‌ها را مشخص می‌کند. (هرخانه آن بردار مربوط به یک آیتم است)

Support DB: پوشش هر آیتم را در بر دارد.

هنگامی که کاربر یک تقاضا مبتنی بر کشف قوانین وابستگی بین آیتمهای IT (که $IT \subset I$) با حداقل پوشش minsupp و حداقل درجه اطمینان minconf می دهد، در این روش ابتدا چک می‌کنیم ببینیم , آیا feature DB , Support DB وجود دارند یا نه در صورتیکه وجود نداشته باشند آن ها را می‌سازد. ساخت Support , feature DB DB احتیاج به یک دور خواندن کل پایگاه داده دارد.در طول خواندن پایگاه داده، Feature DB ، Support DB برای آیتمهای مشخص شده توسط کاربر نیز ساخته می‌شود. پس از اینکه Support DB, Feature DB ساخته شدند در دیسک قرار می گیرند.

لذا, L_1 را نیز با دسترسی مستقیم به Support DB مشخص کنیم, بجای خواندن کل پایگاه داده.

پس از مشخص شدن L_1 ، بردار هر مجموعه آیتم مورد نیاز نیز با عملیات AND به سرعت انجام می‌شود.

1) IF (*SupportDB* is empty)

Scan the database to build *SupportDB* and determine L_1, and build *Feature DB* for interested items;

ELSE

Retrieve *SupportDB* and determine L_1 and build *FeatureDB* for non-existent items;

1-آیتم هایی که توسط کاربر مشخص می شود.

2) k=2
3) Generate candidate large itemset C_k from L_{k-1};

4) IF C_k is not empty
Optain the count for each itemset in C_k by bitwise AND operators on the items' *feature vectors* and get L_k;

$k = k + 1;$
go to step 3;
5) Answer= union of all L_k;

مرحله ساخت بردار v_i برای آیتم d به صورت زیر است:

بردار v_i یک بردار است به طول تعداد تراکنشهای پایگاه داده. در صورتیکه آیتم d در یک تراکنش با شناسه‌ی j باشد $v_i[j]=1$ در غیر اینصورت $v_i[j]=0$. (فرض بر این است که شناسه‌ها در پایگاه داده از 1 تا n-1 طول پایگاه داده" است) مثلاً اگرآیتم a در یک پایگاه داده‌ی 10 عضوی فقط در رکوردهای 10,5,4 رخ دهد, داریم:

$v_a =$

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
0	0	0	1	1	0	0	0	0	1

به این ترتیب با داشتن بردار هر آیتم برای مشخص کردن بردار هر مجموعه آیتم مثل $X=\{i_1,i_2,...,i_n\}$ داریم :

$$V_x = V_{i_1} and V_{i_2} ... and V_{i_n}$$

بدیهی است که تعداد ۱ ها در بردار هر مجموعه آیتم برابر پوشش آن مجموعه آیتم است.

به این ترتیب تمام مجموعه آیتم‌های بزرگ تولید می‌شوند.

واضح است که در یک بردار مقادیر زیادی صفر وجود دارد که فضای زیادی را اشغال می‌کنند. برای رفع این مشکل می‌توان در صورتیکه چندین (بیشتر از 8) 0 پشت سر هم وجود داشت تعداد آنها را بشماریم و در یک index شمارنده ذخیره کنیم. با ذخیره بردار هر آیتم دسترسی به پایگاه داده به حداقل می‌رسد.

یک راه حل دیگر اینست که برای هر آیتم فقط شماره رکوردهایی را که آیتم در آنها رخ داده را ذخیره کنیم یعنی به عنوان مثال اگر آیتم a در رکوردهای شماره 220,201,100 رخ داده بردار آن به شکل زیر است:

$V_a = 100,201,220$

در اینصورت برای هر مجموعه آیتمX بردار V_x به شکل زیر است:

$$X=\{i_1,i_2,...,i_m\}$$
$$V_x = V_{i_1} \cap V_{i_2} \cap ... \cap V_{i_m}$$

ساخت و هرس کردن مجموعه آیتمهای کاندید بزرگ (C_k) از L_{k-1} همانند الگوریتم Aprioriاست با این تفاوت که برای شمارش پوشش هر مجموعه آیتم این کار از طریق عملیات AND یا اشتراک انجام می‌دهیم.

و تنها برای محاسبه‌ی بردارهای هر آیتم, پایگاه داده را در ابتدای کار یک دور می‌زنیم.

یک الگوریتم جدید برای پایگاه داده های پویا :

پویایی پایگاه داده را از دو جنبه می توان بررسی کرد :

۱. هنگامی که داده ها تغییر کنند (یک رکورد تغییر کند) یا داده ها از پایگاه داده حذف شوند .

۲. هنگامی که داده ای (رکورد جدید) به پایگاه داده اضافه شود .

داده کاوی که با نوع اول از پویایی پایگاه داده تطابق دارد را داده کاوی کاهشی و داده کاوی که با نوع دوم پویایی تطابق دارد را داده کاوی افزایشی می نامیم.

الگوریتم (Zhang &Etal,2003) از نوع الگوریتم کاهشی است . یعنی ورودی را بررسی میکنیم که میخواهیم داده کاوی را در یک پایگاه داده که داده هایش دائما حذف شده و یا تغییر کرده اند انجام میدهیم و قاعدتا انتظار داریم که الگوریتم در فواصل مختلف زمانی که پایگاه داده مورد تغییر (یا حذف) قرار می گیرد با بکارگیری الگوهای حاصل از داده کاوی قبلی از هزینه زمانی خوبی برخوردار باشد .

متاسفانه تحقیقی در مورد کشف قوانین وابستگی هنگامی که داده ها از پایگاه داده حذف می شوند صورت نگرفته است . در حالیکه حذف ،یکی از عملیات اساسی در پایگاه داده است .

لذا کاربر باید دوباره الگوریتم داده کاوی را برای پایگاه داده جدید با تولید تمام الگوها از اول بکار گیرد که این کار بسیار پر هزینه است .

نگهداری قوانین وابستگی :

فرض کنید DB پایگاه داده اصلی (قبل ار تغییر) باشد ، db نیز یک مجموعه داده باشد که تصادفا از DB حذف میشوند و DB-db بقیه پایگاه داده ی اصلی باشد .

|DB| ، |db| و |DB-db| به ترتیب اندازه ی پایگاه داده اصلی ، پایگاه داده ی حذف شده و پایگاه داده ی حاصل از حذف داده ها از DB باشد . همچنین فرض کنید S0 حداقل پوشش تعیین شده توسط کاربر باشد .

L ، L' و L" مجموعه ی مجموعه آیتمهای بزرگ در DB ، db و DB-db می باشد . فرض کنید که مجموعه L و پوشش هر مجموعه آیتم بزرگ در آن مشخص است .

بعد از یک دوره زمانی برخی از داده ها ی غیر مفید از پایگاه داده اصلی حذف می شوند و پایگاه داده ی حذف شده ی db را تشکیل میدهند .

فرض کنید یک مجموعه آیتم مثل X موجود باشد با پوشش Supp X . X یک مجموعه آیتم بزرگ است اگر ≥ Supp minSupp

بنابراین یک مجموعه آیتم مثل X میتواند در پایگاه داده جدید (DB-db) بزرگ نباشد .

الگوریتم کاهشی :

در این الگوریتم با داشتن L (مجموعه آیتمهای بزرگ DB) و S0 (حداقل پوشش) ، مجموعه آیتمهای بزرگ در پایگاه داده ی جدید (DB-db) را که در L" قرار میدهیم ، بدست می آوریم .

این الگوریتم به شکل زیر آمده است :

```
Compute S' ;  /* S' is explained later in this section and also in Section 4.1. */

Mine db with minimum support S' and put the frequent itemsets into L' ;

For each itemset in L do
        For each itemset in L' do
          Identify the frequent itemsets in DB-db, eliminate them          from L' and
store to L" ;
Return L";
End procedure ;
```

در این الگوریتم برای S' مقادیر مختلفی وجود دارد . بطور کلی S' = m.S0 که 0<m<1 است .

مشخص شدن m یک مساله قابل بحث و تحقیق است .

نتیجه گیری:

هدف اصلی از داده کاوی ,کشف دانش نهفته در محیط مورد بررسی است که این دانش می تواند شکلهای گوناگونی داسته باشد . دانش استخراج شده می تواند به فرم الگوهای موجود در داده ها باشد که کشف این الگوها منجر به شناخت

بهتر سیستم نیز می شود . الگوهای استخراجی عموما بیانگر روابط بین ویژگیهای سیستم هستند. همانطور که دیدید در این تحقیق ما الگوریتم های مختلف برای کشف قوا نین وابستگی در پایگاه داده را, مورد بررسی قرار دادیم. از بین الگوریتم های گفته شده Vector,Partition,Apriori از مهمترین الگوریتم ها می باشند.

الگوریتم های trie,fp-grow نیز در موارد خاص بسیار کارا هستند.

در تمام الگوریتم های بحث شده پوشش حداقل ثابت بوده و فقط یک مقدار به عنوان پوشش حداقل داشتیم.الگوریتم هایی(Lee &Hong,2005) برای مواقعی که برای هر آیتم یک پوشش حداقل,در نظر می گیریم نیز طراحی شده است که از گنجاندن آنها در اینجا خودداری کردیم.

همچنین الگوریتم هایی برای مواردی که در پایگاه داده, به ازای هر مشتری علاوه بر اجناس خریداری شده تعداد هر جنس را هم داریم , موجود می باشد.

اخیرا نیز الگوریتم های فازی نیز برای کشف قوانین وابستگی در پایگاه داده, طراحی شده است .

مراجع:

1)A.amiri,R.Feldman and R.Kashi. A new and versatile method for association generation. Information Systems, vol. 22,no. 6, pp. 333-347, 1999.

2)J.Hipp,U.Guntzer and G.Nakhaeizadeh.Algorithms for Association Rules data Mining _ a General survey and comparision. Source, ACM SIGKDD Explorations Newsletter,2(1):58–64, July 2000.

3) Y.Lee ,T.Hong and W.Lin.Mining association rules with multiple mininmum supports using minimum constraints . International Journal of Approximate Reasoning 40(2005) 44-54.

4)S.Zhang ,X.Wn , J.Zhang and C.Zhang. A Decremental Algorithm for Maintaining Frequent Itemsets in Dynamic Database. DaWaK 2005, LNCS 3589, pp. 305–314, 2005.

5) S.Tseng.An Efficient Method for Mining Association Rules With Item Constraint . Technical Report No. CSD-99-1089,2000.

6) P.Hsu,Y.chen and C.ling.Algorithms for mining associations rules in bag database . INFORMATION SCIENCES 166(2004) 31-47.

7)Y.GAO,J.MA and L.Ma.

A New Algorithm for mining fuzzy association rules.preceedings of third international confrance of machine learning and cybmetic, shanghai,26_29 Agust 2004 .

اطلاعات مورد نیاز نیز با استفاده از اینترنت تهیه شده است .کلید واژه های مورد استفاده عبارتند از :

Data Mining , Graph Data Base , Graph Search Algorithms

سایتهایی که با کلید واژه های فوق نمایش داده می شوند تعداد نسبتا زیادی را تشکیل می دهند پس از مطالعه سایتها ,سایتهای زیر قابل استفاده تشخیص داده شدند و مورد استفاده قرار گرفتند :

citeseer.nj.nec.com/gting94graphdb.html: صفحه ابتدایی GraphDBاست که نحوه پیاده سازی گراف در بانک اطلاعاتی را توضیح می دهد .

db.cs.sfu.ca/DBMiner : سایتی است حاوی اطلاعاتی در مورد معماری نرم افزار داده کاوی و نمونه یک نرم افزارتولید شده در این زمینه که بر روی انبار داده عمل داده کاوی را انجام می دهد .

db.cs.sfu.ca/GeoMiner : سایتی است حاوی اطلاعاتی در مورد داده کاوی و یک پروژه در زمینه داده کاوی فضایی .

www.mecani.org : سایتی است حاوی اطلاعاتی در مورد بانکهای اطلاعاتی.

www.sigmod.com : سایتی است حاوی اطلاعاتی در مورد داده کاوی از بانکهای رابطه ای و فضایی.

ftp.fas.sfu.ca : این سایت حاوی معماری و الگوریتمهای متعددی برای سیستمهای داده کاوی است .

هرگونه سوال در زمینه Data Mining را می توانید با این email نیز در میان بگذارید .

training@the-modeling-agency.com