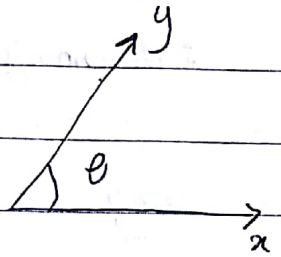
مثلاً $\frac{1}{10}$ متن شده apple. حال یک فرمول پیدا کرده که شباهت بین دو vector را بفهمیم.

apple در متن ۱ ۱۰ بار تکرار شده و در متن دوم 30.

	Apple	Microsoft	obama	Electro
D1	10/30	20/30	0	0
D2	30/90	60/90	0	0



برای پیدا کردن شباهت چندین مورد داریم:

۱- Cos_Similarity : مثلاً داریم:

اگر: $\theta = 90$, $\cos\theta = 0$ → عمود بر هم → شبیه نیستند

اگر: $\cos\theta = 1$, $\theta = 0$ → بر هم منطبق اند → شبیه اند

تصویر هندسی اندازه شباهت:

$\cos(x, y)$ می‌شود شباهت بینشان

برای مقایسه Document ها استفاده از cos رایج تر است.

به شرطی می‌توان از cos استفاده کرد که از قبل داده‌ها را نرمال سازی کنیم.

$$\cos(d_1, d_2) = \frac{d_1 * d_2}{\|d_1\| \|d_2\|}$$

۲- distance : مقداری که می‌فهمیم دو object چقدر متفاوت اند.

هرچه به صفر نزدیک باشد، شبیه‌ترند.

برای پیدا کردن فاصله چندین روش داریم:

الف) اقلیدسی:

$$\text{فاصله در اقلیدسی} = \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\dots}$$

ب) منهتن:

$$\text{فاصله منهتن} = |x_{i1}-x_{j1}| + |x_{i2}-x_{j1}| + \dots$$

ج) مینکوفسکی:

$$\text{فاصله در مینکوفسکی} = \left(|x_{i1}-x_{j1}|^h + |x_{i2}-x_{j2}|^h + \dots\right)^{1/h}$$

د) Hamming distance:

تعداد بیت هایی که با هم تفاوت دارند (در دو کد)

چطوری شباهت بین String ها را پیدا می‌کنیم؟ در آنالیز DNA کاربرد دارد

Edit distance: تعداد Delete و Insert هایی که با هم متفاوت هستند که باید شبیه هم باشند.

فصل ۶ کتاب: پیدا کردن قوانین انجمنی و الگوهای پرتکرار

مفاهیم: الگوی پرتکرار: مثلاً در یک فروشگاه کالایی است که بیشتر از همه خریداری شده اند.

چرا الگوهای پرتکرار را باید داشته باشیم؟

یکی از سوالات ← بعد از چه کالایی، چه کالایی می‌خریم. یعنی توالی در خرید کالا
مثلاً می‌دانیم هرکس کامپیوتر بخرد، بعد از آن پرینتر می‌خرد.
پس میایم یک تخفیف برای پرینتر می‌ذاریم. که همراه خریداری بشوند.

۱- توالی در خرید آیتم ها

۲- چه آیتم هایی با هم خریداری شده اند. (مثلاً شیر و ماست با هم خریده شده اند پس اینا را کنار هم می‌چینیم)

۳- DNA تاثیر یک چیز بر روی چیز دیگر

مثلا یک دارو روی DNA های مختلف چه تأثیری نداشته

یا مثلا چیدمان فروشگاه چه تأثیری در میزان فروش داشته.

۴- طبقه بندی اسناد وب   مثلا این خبرگذاری ها باید کنار هم باشند.

مثلا هر کسی اخبار ایران را سرچ کرده اخبار افغانستان هم سرچ کرده پس این دو باید کنار هم باشند.

جاهای زیادی کاربرد دارد مثل سیستم فروش و بازاریابی، علم پزشکی، پیمایش صفحات وب ← در همه ی این ها نیاز داریم الگوهای پرتکرار را پیدا کنیم

سوال: اگر کسی کامپیوتر خرید، بعد از خرید آن به چه احتمالی آنتی ویروس می خرد؟

قانون: باید یک قانون را پیدا کنیم که این اتفاق افتاده.

$$Computer \Rightarrow Software \; [support = 2\%, \; confidence = 60\%]$$

$$A \Rightarrow B$$

باید روابط بسازیم.

فرض کنیم فروشگاه n آیتم به شکل زیر دارد که می خواهیم یک کالای A بخریم.
بعد از خرید A، حتما B را بخریم که اشتراک این دو تهی است.
هیچ وقت سمت راست و چپ نباید اشتراک داشته باشند.

T: مجموعه آیتم ها   I: هر item

$A \Rightarrow B$

$A \cap B = \emptyset$

$T = \{I_1, I_2, \ldots, I_n\}$

page: ( )

Subject:
Year : Month : Day : ( )

Support : (درجه پشتیبان) فرض کنید قانون در مجموعه تراکنش D با S که

support است. در این صورت S درصدی از تراکنش D است که

حاوی A و B است و با احتمال P(AUB) نمایش می‌دهند

support(A⇒B) ⇒ AUB

یعنی جاهایی که دو کالا با هم خریده شده اند.

Confidence : اگر طرف اول true باشد (یعنی A رخ بدهد) چه قدر احتمال دارد

که طرف دوم نتیجه درست باشد.

$$P(B|A) = \frac{P(AUB)}{P(A)}$$

یعنی اگر A رخ داد چه قدر احتمال دارد B رخ دهد؟

این support که می‌گیم min حالت است.

پس الان که min=۲٪ یعنی خیلی جاها A و B با هم اند پس خیلی قانون داشتیم گرفته‌ایم.

اما وقتی می‌گیم ۹۸٪ ساپورت باشد. پس داریم قانون را خیلی سخت می‌گیریم

یعنی خیلی کم پیش می‌آید که می‌گیم ۹۸٪ باید این دو باهم خریداری شده باشند.

اگر قانون هیچ چیزی پیدا نکند پس این قانون به درد نمی‌خورد.

یا مثلاً قانون را این بگیریم کسانی که امروز یک کالا خریده اند. این احتمال خیلی زیاد

است. ← از آن چه می‌خواهیم بدست آوریم ← هیچی به درد نمی‌خورد!

پس زیاد ریتا خارج کردن و کم ریتا خارج کردن به درد نمی‌خورند.

یه چیز متوسط می‌خواهیم نه زیاد سختگیرانه که چیزی پیدا نشه و نه زیاد آزادی بدیم.

ItemSet مجموعه آیتم ها

Subject:
Year : Month : Day : ( )

page: ( )

Item

K-Itemset اگر Itemset ها K عضو داشته باشند

فراوانی (تکرار) : نرخ تکرار یک Item در مجموعه دیتاست.

Support
- → Absolute Support تعداد رخدادهایی که A و B همزمان با هم رخ می‌دهند.
- → Relative Support احتمال اینکه A و B با هم رخ می‌دهد

Relative با confidence فرق دارد چون confidence میگه اولی حتماً رخ داده و ...

$$\text{Relative} \rightarrow \text{احتمال} = \frac{\text{تعداد رخ دادگی A و B}}{\text{تعداد کل رخ دادها}}$$

مثال : فرض کنید یک فروشگاه چنین تراکنش دارد.

TID	
10	Coke, nuTs, diaper
20	Coke, Coffe, diaper
30	Coke, diaper, egg
40	Nut, Egg, milk
50	Nut, coffee, diaper, egg, milk

min-support=2 قرار می‌دهیم
پس باید بیشتر از ۲ بار آمده باشد تا ما بنویسیمش.
الان می‌خواهیم 1-Itemset های را پیدا کنیم که پرتکرارند:

1-Itemset = {Coke} {nut} {diaper} {coffee} {egg} {milk}

پس این روش خیلی کارایی دارد.
اما اگر minSupport = 4 باشد.
که هیچ چیزی پیدا نمی‌شود حتی برای 1-Itemset پس خیلی سختگیرانه است.

2-Itemset = {Coke, diaper} دوتا دوتا همه را بررسی می‌کنیم
{Nut, egg} {egg, milk}

3-Itemset = {Nut , egg , milk}

حالا 4-Itemset چیزی پیدا نمیکنیم

پس تا اون بعد تکراری نمیشه. چون تراکنشی هر ۴ تا با هم خریده باشند پیدا نمیشود

اما اگر min support = 1 باشد این ۴ تایی هم پیدا میشود

پس min support را جوری در نظر میگیرند که این مجموعه ای که بدست میاید متناسب باشد.

support confidence

Coke ⇒ diaper $(\frac{۳}{۵} = ۶۰\% \;,\; \frac{۳}{۳} = ۱۰۰\%)$

diaper ⇒ coke $(\frac{۳}{۵} = ۶۰\% \;,\; \frac{۳}{۴} = ۷۵\%)$ یعنی diaper خریده بعد coke هم میخرد

تعیین ضرایب confidence و support.

اما محیط داده کاوی خیلی بزرگ است و این روش ها کم میارند :(

اگر یک مجموعه پرتکرار باشد، آیا زیر مجموعه های آن هم پرتکرار است؟ بله

فرض کنید یک مجموعه ۱۰۰ تایی پیدا کردیم که پرتکرار است.

تعداد زیر مجموعه های این مجموعه: (زیرمجموعه تک عضوی، زیرمجموعه ۲ عضوی)

$$\binom{۱۰۰}{۱} + \binom{۱۰۰}{۲} + \binom{۱۰۰}{۳} + \cdots + \binom{۱۰۰}{۱۰۰} = ۱۰^{۳۰}$$

پس $۱۰^{۳۰}$ مجموعه پرتکرار پیدا کردیم

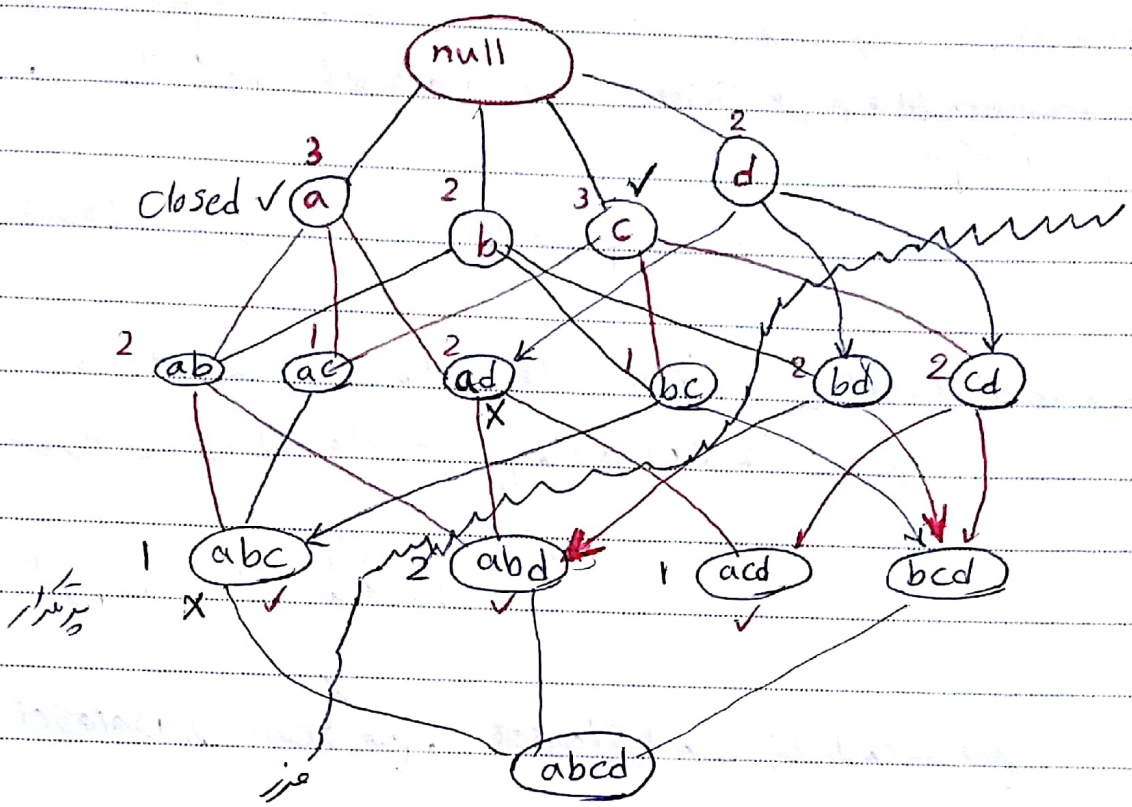
اما نیازی نیست همه این ها را ذخیره کنیم فقط همان مجموعه ۱۰۰ تایی را ذخیره میکنیم.

Closed Frequent Itemset (بسته): یک Itemset مانند x را Closed میگویند اگر هیچ Super Itemset ای مانند y وجود نداشته باشد که support آن مشابه x باشد.

(Super Itemset: آیتم ست بالاتر از خودش - فراتر از خودش)

مثلا در مثال قبل از مجموعه 3-Itemset ای چیزی بیشتر پیدا نکردیم پس این مجموعه closed شد.

Year. Month. Date. ( )



در این گراف یک مرز بین پرتکرارها و کم تکرارها می‌کشد.
کشیدن این مرز ممکن است دقیق نباشد.
آیتم‌هایی که روی مرز هستند support هایشان را بررسی می‌کنیم.
اگر آیتم همه زیرمجموعه‌هایش در سمت پایین مرز باشد، آن ماکسیمال فری کوئنت
دیتاست می‌گوییم مثل ad و abc

۳ تا الگوریتم برای پیدا کردن الگوهای پرتکرار:

Apriori - FP Growth - ECLAT

Apriori براساس دانش قبلی از یک الگوی پرتکرار استخراج می‌کند.
اگر مجموعه‌ای پرتکرار باشد ← زیرمجموعه‌های آن پرتکرار است.
این الگوریتم از این قانون استفاده می‌کند.
اگر یک K-Itemset پرتکرار داشته باشیم از روی آن می‌توانیم یک (K+1) Itemset
پرتکرار پیدا کنیم.

Subject:
Year. Month. Date. ( )

مراحل الگوریتم:

۱- در ابتدا مجموعه Dataset را Scan می‌کنیم: اول باید دیتاست را نگاه کنیم ببینیم کجا این اتفاق افتاده

۲- 1-Itemset های پرتکرار را استخراج می‌کنیم: l_1

۳- 2-Itemset ها را با استفاده از join داخلی 1-Itemset ها بدست می‌آوریم
بعد پرتکرارهای آن را استخراج می‌کنیم و اسم آن را C2 می‌ذاریم.

حال مثال فروشگاه را در نظر بگیرید:

۱- Dataset را Scan کنیم و 1-Itemset های پرتکرار را بدست آوریم:

l_1 = { {cok} {nut} {diaper} {coffee} {egg} {milk} }

۲- l_2 = { ~~{cok, nut}~~ {cok, diaper} ~~{cok, coffee}~~ ~~{cok, egg}~~ ~~{cok, milk}~~
{nut, diaper} ~~{nut, coffee}~~ {nut, egg} {nut, milk} }

minsupport = 2 حالا هر کدام که تعداد تکرار آن از minsupport کمتر باشد حذف می‌کنیم.

برای پیدا کردن 3-Itemset ها l_2 را با خودش join می‌کنیم (همشه از مرحله قبلی)
پرتکرارهای l_2 را با خودش join می‌کنیم ← اگر ۴ تایی شد نمی‌نویسیم فقط ۳ تایی‌ها را می‌نویسیم

l_3 = { cok, nut, diaper }

این مثال کامل نیست

این روند را تا جایی ادامه می‌دهیم که دیگر مجموعه پرتکرار پیدا نکنیم و می‌گوییم Closed شده.

مثال: مجموعه های پرتکرار را بدست آورید با استفاده از الگوریتم.

minsupport = 2

TID	
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

$L_1 = \{ \{A\}_2 \ \{B\}_3 \ \{C\}_3 \ \cancel{\{D\}_1} \ \{E\}_3 \}$

حرس می کنیم

Pruning

چون minsupport 2 است

$L_2 = \{ \cancel{\{A,B\}_1} \ \{A,C\}_2 \ \cancel{\{A,E\}_1} \ \{B,C\}_2 \ \{B,E\}_3 \ \{C,E\}_2 \}$

$L_3 = \{ \cancel{\{A,B,C\}_1} \ \cancel{\{A,C,E\}_1} \ \{B,C,E\}_2 \}$

پس مجموعه $\{B,C,E\}$ پرتکرار است ← زیر مجموعه های آن هم پرتکرار است.

مجموعه D حذف شده در مراحل بعدی D را نداریم.

پس اگر یک مجموعه پرتکرار نبود در مراحل بعدی نیز پرتکرار نیست.

مثال: * در صفحه بعد کاربرد دارد. minsupport = 2

TID	
T100	I_1, I_2, I_5
T200	I_2, I_4
T300	I_2, I_3
T400	I_1, I_2, I_4
T500	I_1, I_3
T600	I_2, I_3
T700	I_1, I_3
T800	I_1, I_2, I_3, I_5
T900	I_1, I_2, I_3

Year. Month. Date. ( )

عیب این روش این است که هربار باید کل Dataset را Scan نمیم.
یا مثلاً یک سری مجموعه های اضافه داریم که باید آن ها را هرس کنیم پس بیاد چیزایی تولید نکنه که بدرد نخور باشند و همه را تولید نکند
در دیتا بیس های بزرگ خیلی بد می شود ← یعنی زمان بر - زیاد اینجا مثلاً دیتا بیس کوچک بود.

اصلی ترین چالش های محاسباتی:
۱- مرور چندباره پایگاه داده تراکنش ۲- تعداد زیاد کاندیداهای تولید شده
۳- محاسبه Support برای کاندیداها زمان گیر است.
به دنبال راه حلی هستیم که این چالش ها را برطرف کنیم.
راه حل که: ← ۱- کاهش مرور (اسکن) پایگاه داده
۲ کاندیداهای مفید تولید شوند
۳ محاسبه سریع Support ها

۵ روش مجبود Apriori: ۱ درهم سازی ۲ کاهش تراکنش ها ۳ پارتیشن بندی
۴ نمونه گیری ۵ شمارش پویا

درهم سازی: روی یک مثال پیاده می کنیم.
ایده اصلی: اهزمان با اینکه 1-Itemset ها تولید می شوند 2-Itemset ها هم تولید می شوند
۱ تابع درهم سازی تعریف می کنیم و عناصر 2-Itemset را به مقدار تابع hash نسبت می دهد
۳- هر مجموعه یک bucket گفته می شود.
۴ اگر شمارش اعضای bucket از min-Support بیشتر باشد
عنوان الگوی پرتکرار frequent شناخته می شود.

مثال: این مثال در صفحه قبل نوشته شده.

TID	Items
	I_1, I_2, I_3

$$hash = ((order\ of\ x) \times 10 + (order\ of\ y)) \% 7$$

باقی مانده تقسیم بر ۷ می‌تواند اعداد ۰ تا ۶ باشد. پس این حالت ها را می‌توانیم داشته باشیم

مثلاً $(I_1, I_4) \Rightarrow 1 \times 10 + 4 = 14 \% 7 = 0$

bucket Address	0	1	2 ✓	3	4	5 ✓	6 ✓ (frequent)
bucket count (تعداد باکت ها)	2	2	4	2	1	4	4
bucket content (محتوای باکت)	$\{I_1,I_4\}$ $\{I_3,I_5\}$	$\{I_1,I_5\}$ $\{I_1,I_5\}$	$\{I_2,I_3\}$ $\{I_2,I_3\}$ $\{I_2,I_3\}$ $\{I_2,I_3\}$	$\{I_2,I_4\}$ $\{I_2,I_4\}$	$\{I_2,I_5\}$	$\{I_1,I_2\}$ $\{I_1,I_2\}$ $\{I_1,I_2\}$ $\{I_1,I_2\}$	$\{I_1,I_3\}$ $\{I_1,I_3\}$ $\{I_1,I_3\}$ $\{I_1,I_3\}$

اگر minSupport=3 ← مجموعه هایی که bucket count آن ها بیشتر مساوی ۳ باشد frequent است.

تابع hash یک سری تداخل ها دارد. دوتا چیز ممکن است به یک frequent بخورند.

این روش خطا دارد. می‌توانیم تابع hash را جوری انتخاب کنیم که تا حد ممکن این تداخل برطرف شود.

تداخل ← مثلاً ۴ تا مجموعه داریم اما این ۴ تا مجموعه همجنس نیستند مثلاً دوتا $\{I_1,I_2\}$ و دوتا $\{I_2,I_5\}$ با فرض اینجا نمی‌دونیم که frequent را چی بنویسیم.

کاهش تراکنش: اگر تراکنش در مجموعه k_Itemset الگوی پرتکرار ندارد پس در مجموعه k+1 Itemset هم حتماً الگوی پرتکرار ندارد بنابراین آن تراکنش را حذف می‌کنیم.

مثلاً $\{I_2,I_4\}$ پرتکرار نیست ← مربوط به کدام تراکنش است؟ تراکنش 200

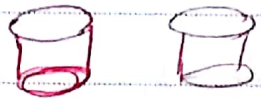
پس تراکنش 200 را حذف می‌کنیم و دفعات بعدی آن را بررسی نمی‌کنیم

اما تراکنش 400 را حذف نمی‌کنیم! چون I_1 را هم دارد.

تراکنش هایی را حذف می‌کنیم که فقط I_2, I_4 را دارند.

Subject:
Year. Month. Date. ( )

پارتیشن بندی: فرض کنید یک دیتابیس داریم که 10G است اما فضایی که داریم 1G است. اگر قرار باشد هر دفعه 1G از دیتابیس را بیاریم تو حافظه و 1-Itemset پیدا کنیم، به این شکل باید 10 بار از دیسک اطلاعات را بیاریم توی حافظه اصلی.
مثلاً فرض کنید minSupport 20 باشد



دیتابیس را 10 قسمت کردیم اگر هر مجموعه یا آیتم در هر قسمت 2 بار تکرار شده باشد در نهایت در 10 قسمت 20 بار تکرار می‌شود و پرتکرار است.
اما ممکن است توزیع داده‌ها یکسان نباشد مثلاً یک جا 10 بار تکرار شده و یک جا صفر بار.
پس نمی‌توان نتیجه گرفت اگر در یک قسمت پرتکرار باشد حتماً در نهایت هم پرتکرار است.

local ← در یک پارتیشن ، global ← در کل

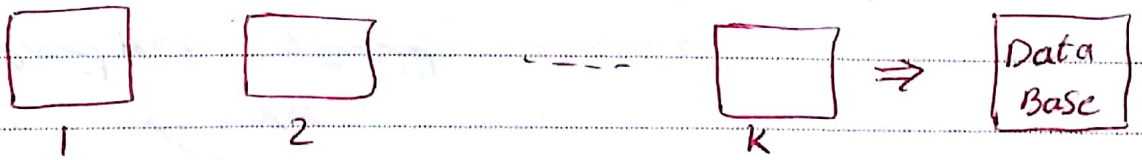
اگر یک Item در کل پرتکرار باشد در یک قسمت یا local هم پرتکرار می‌شود.
این قسمت مثل قسمت‌های قبلی نیست که Data Base کار داشته باشد بلکه با استفاده از دیسک و حافظه کار داریم.

local global

frequent در یک local ⟸ frequent

frequent ⇏

پارتیشن بندی در دو فاز انجام می‌شود:
در فاز اول: داده‌ها به چند بخش تقسیم می‌شوند (به طوری که هر بخش بتواند در حافظه اصلی قرار گیرد)



Sup > min support نسبی

Sup > min support

تعریف min support نسبی: یا درصد پشتیبان نسبی = min تکرار برای هر min support ⟸ کل تعداد تراکنش‌های یک بخش
پارتیشن برابر است با

مثلاً min support = 20 است و دیتابیس به 10 قسمت تقسیم می‌شود به طور متوسط در هر قسمت 2 بار تکرار باید

min support نسبی

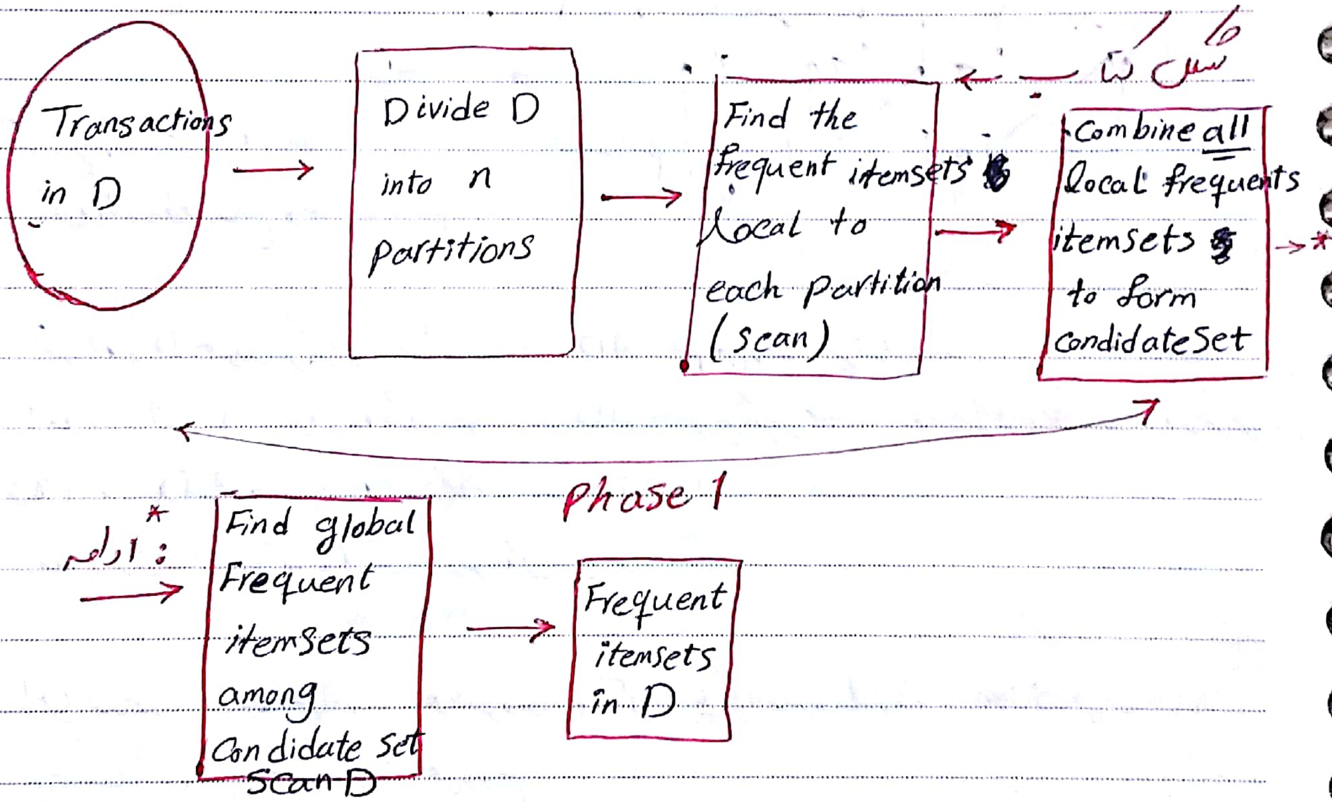
چند نتیجه می‌گیریم ← ۱. الگوهای پرتکرار هر بخش (الگوهای محلی) ممکن است نسبت به کل پایگاه داده پرتکرار باشند یا نباشند

۲. هر Itemset که در پایگاه پرتکرار باشد باید حداقل در یکی از پارتیشن های پایگاه داده پرتکرار باشد.

۳. نتیجه: هر itemset پرتکرار در یک بخش می‌تواند در مجموعه کاندید کل پایگاه داده باشد.
(یعنی هنوز کاندید است و اینکه پرتکرار باشد یا نه در فاز دوم مشخص می‌شود)

فاز دوم ← مرور مجدد پایگاه داده
برای هر itemset کاندید باید support واقعی را محاسبه می‌کنیم

در واقع این روش هم دو بار پایگاه داده را مرور می‌کنه نه چند بار.



support کل هر itemset را
به min support کل مقایسه کنیم.

Subject:
Year. Month. Date. ( )

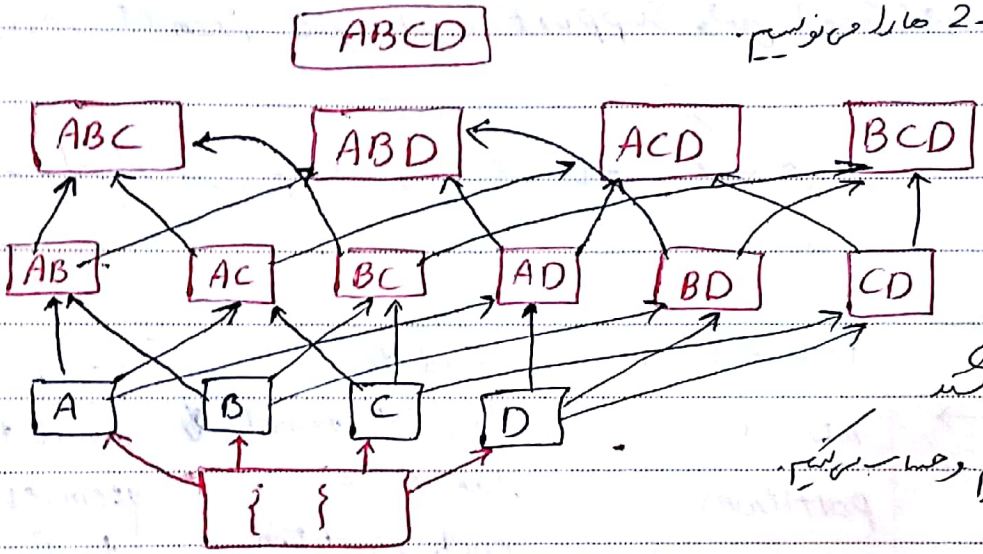
نمونه برداری :

مشکل : در حالتی نمونه برداری می‌کنیم که نمونه‌های حاضری به هم نزدیک باشند اما در نمونه‌ها از هم فاصله داشته باشند این کار قابل قبول نیست.

شمارش پویا : هدف این کار برای کاهش تعداد اسکن‌هاست.

مثال : اول کار null است، هیچی نداریم. بعد 1-Itemset ها را می‌نویسیم.

مرحله بعد 2-Itemset ها را می‌نویسیم.



اگر زیری‌ها پرتکرار باشند آن را در نظر می‌گیریم و حساب می‌کنیم.

اگر A و D هر دو پرتکرار باشند ← AD هم می‌تواند پرتکرار باشد.

اما اگر A و D پرتکرار نباشند ← AD هم پرتکرار نیست و در مراحل بعد آن را در نظر نمی‌گیریم.

BCD ← AC و BC و BD و CD

زمانی پرتکرار است که تمام مجموعه زیر آن پرتکرار باشد.

این روش Apriori و همه‌ی آن پایگاه داده را زیاد Scan می‌کند.

پس سراغ روش‌های دوم می‌رویم.