VERİ MADENCİLİĞİ İlişkilendirme Kuralları: Farklı Yöntemler

Prof. Dr. Şule Gündüz Öğüdücü

http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

Çok Katmanlı İlişkilendirme Kuralları

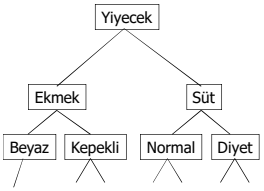
- Nitelikler bir hiyerarşi oluşturabilir.
  - kavram hiyerarşisi
- Daha alt katmanlarda yer alan ürünlerin destek değeri daha düşük
- Alt seviyelerde oluşan kurallar özel durumları tanımlıyor

  Beyaz ekmek => normal süt

  Kepekli ekmek => normal süt

  Kepekli ekmek => diyet süt

  kuralları ekmek => süt kuralının bir göstergesi



Çok Katmanlı İlişkilendirme Kuralları

- Destek ve güven değerlerinin katmanlar arasında değişimi
  - X1 ve X2 niteliklerinin üst düğümünde yer alan nitelik X

    $\sigma(X) \geq \sigma(X1) + \sigma(X2)$
  - $\sigma(X1 \cup Y1) \geq$ minsup, X1 niteliğinin üst düğümünde yer alan nitelik X, Y1 niteliğinin üst düğümünde yer alan nitelik Y

    $\sigma(X \cup Y1) \geq$ minsup, $\sigma(X1 \cup Y) \geq$ minsup

    $\sigma(X \cup Y) \geq$ minsup
  - conf(X1 $\Rightarrow$ Y1) $\geq$ minconf ise conf(X1 $\Rightarrow$ Y) $\geq$ minconf

Çok Katmanlı İlişkilendirme Kuralları

- 1. Yaklaşım:

  Her hareket t, üst katmanlardaki nitelikler cinsinden yazılarak genişletilmiş harekete t' dönüştürülür

  t = {beyaz ekmek, normal süt}

  t' = {beyaz ekmek, normal süt, ekmek, süt, yiyecek}

  t' hareketleri için ilişkilendirme kuralları bulunur
  - Problemler:
    - Üst düğümlerde yer alan niteliklerin destek değerleri daha büyük
      - Farklı seviyelerde farklı destek değerleri
    - Veri kümesinin boyutu artıyor

Çok Katmanlı İlişkilendirme Kuralları

- Alt katmanlarda destek değerini küçültme
  - Her katmanda bağımsız hesaplama (Level-by-level independent)
  - İlk olarak en üst katman için yaygın nitelikler bulunur. Her aşamada bir alt katmandaki nitelikler arasındaki ilişkiler bulunur
    - Problemler:
      - Kullanışlı değil: her katman için veri kümesi araştırılıyor
      - Farklı katmanlar arasındaki ilişkiler bulunamıyor

Çok Katmanlı İlişkilendirme Kuralları

- Alt katmanlarda destek değerini küçültme
  - Her katmanda bağımsız hesaplama (Level-by-level independent)
  - Üst düğümleri yaygın nitelik olan niteliklerin destek değerini hesaplama
    - Tek niteliğe göre budama (Level-cross filtering by single item)
    - nitelikler kümesine göre budama (Level-cross filtering by k-itemset)

Çok Katmanlı İlişkilendirme Kuralları

- 2. Yaklaşım

  İlk olarak en üst katman için yaygın nitelikler bulunur

  Her aşamada bir alt katmandaki nitelikler arasındaki ilişkiler bulunur

  - Problemler:
    - Kullanışlı değil: her katman için veri kümesi araştırılıyor
    - Farklı katmanlar arasındaki ilişkiler bulunamıyor

7

Gereksiz Kuralları Eleme

- Ürünler arasındaki ilişki nedeniyle bazı kurallar gereksiz olabilir
- Örnek
  - satın_alma (X, "Ekmek") => satın_alma(X,"Süt") [%8, %70]
  - satın_alma (X, "Beyaz Ekmek") => satın_alma(X,"Süt") [%2,%70]
- İlk kural ikinci kuralın üst kuralı
- Eğer bir kuralın destek değeri üst kuralının destek değerinden tahmin edilebiliyorsa gereksizdir

8

Nicel İlişkilendirme Kuralları

Kayıt No	Yaş	Medeni Durum	Araba Sayısı
1	23	Bekar	1
2	25	Evli	1
3	29	Bekar	0
4	34	Evli	2
5	38	Evli	2
…	…	…	…

- Örnek ilişkilendirme kuralları:

  yaş:30-39, medeni durum: evli => araba sayısı:2 (destek: %40, güven: %100)

  araba sayısı: 0-1 => medeni durum: bekar (destek: %40, güven: %66,70)

9

Nicel İlişkilendirme Kuralları

- İkili değişkenlerden oluşan ilişkilendirme kurallarına dönüştürülüyor:
  - Kategorik olamayan nitelik değerleri için bölmeleme
    - yaş: 20-29, 30-39
  - ikili değişken oluşturma
    - kategorik niteliklerin her değeri için bir ikili değişken
    - kategorik olmayan niteliklerin her bölmesi için bir ikili değişken

Kayıt No	Yaş	Medeni Durum	Araba Sayısı
1	23	Bekar	1
5	38	Evli	2

Kayıt No	Yaş 20-29	Yaş 30-39	M.Durum Evli	M.Durum Bekar	Araba 0	Araba 1	Araba 2
1	1	0	0	1	0	1	0
5	0	1	1	0	0	0	1

10

Nicel İlişkilendirme Kuralları

- Dönüştürmeden kaynaklanan problemler
  - bölme sayısı az: Bilgi kaybolması
  - Destek değeri küçük: fazla sayıda kural

11

Koşullu Veri Madenciliği

- Veri madenciliği algoritmasının olası bütün örüntüleri çıkarması
  - gerçekci değil
  - çok fazla sayıda amaca hizmet etemeyen örüntü çıkabilir
- Veri madenciliği yöntemleri kullanıcı ile etkileşimli çalışabilir
- Koşullu veri madenciliği
  - kullanıcıya esneklik sağlar
  - sistemin daha verimli çalışmasını sağlar

12

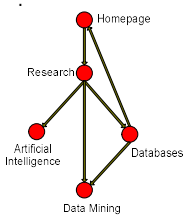
Veri Madenciliği için Koşullar

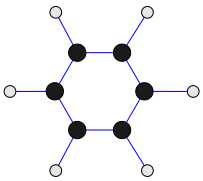
- Bilgi koşulu
  - ilişkilendirme, sınıflandırma..
- Veri koşulu
  - SQL benzer sorgulamalar
  - İstanbul'da aralık 2007'de en çok satılan ürün çiftlerini bul
- Boyut koşulu (katman koşulu)
  - veri madenciliğinde kullanılacak boyutları veya katmanları belirler
- İlginçlik koşulu
  - min_sup>%3 ve min_conf>%70 olan kuralların bulunması
- Kural (örüntü) koşulu
  - kuralların yapısını belirler

13

Yaygın Altçizge Madenciliği

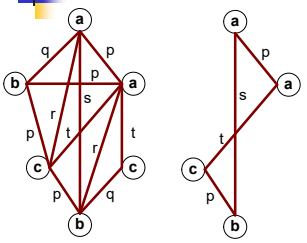
- Frequent Subgraph Mining
  - İlişkilendirme kuralları madenciliği yöntemleri uygulanarak yaygın altçizgeler bulunur
- Uygulama alanları
  - Web madenciliği
  - Biyoinformatik
  - :





14

Çizge Tanımları



(a) Etiketli Çizge (b) Altçizge

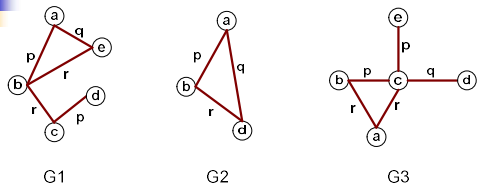
- $V=\{x_i\}$ düğümler kümesi
- $E=\{w_{ij}\}$ x_i ve x_j düğümleri arasındaki ağırlık
- Altçizge: $G_i(V_i,E_i)$

$$\bigcup_{i=1}^{k} V_i = V$$

$$E_i = \{\{u,v\} \in E \wedge u,v \in V_i\}$$

15

Çizge-Hareket Dönüşümü



G1 G2 G3

	(a,b,p)	(a,b,q)	(a,b,r)	(b,c,p)	(b,c,q)	(b,c,r)	…	(d,e,r)
G1	1	0	0	0	0	1	…	0
G2	1	0	0	0	0	0	…	0
G3	0	0	1	1	0	0	…	0
G3	…	…	…	…	…	…	…	…

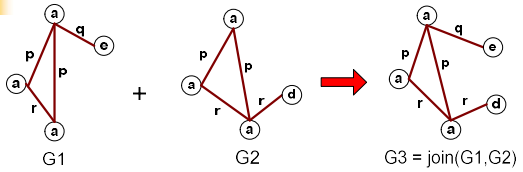
16

Apriori Temelli Yaklaşım

- Bir altçizgenin destek değeri:
  - Altçizgenin bulunduğu çizgelerin toplam çizgelere oranı
- Apriori temelli yaklaşım
  - yaygın (k+1)-altçizgeleri bulmak için k-altçizgelerin kullanılması
    - düğüm ekleme: k düğüm sayısı
    - ayrıt ekleme: k ayrıt sayısı

17

Düğüm Ekleme

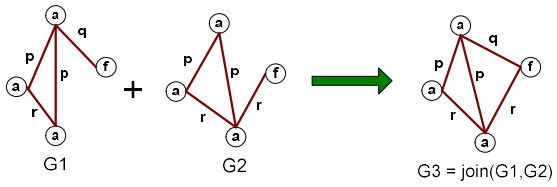


G1 G2 G3 = join(G1,G2)

$$M_{G1} = \begin{pmatrix} 0 & p & p & q \\ p & 0 & r & 0 \\ p & r & 0 & 0 \\ q & 0 & 0 & 0 \end{pmatrix} \quad M_{G2} = \begin{pmatrix} 0 & p & p & 0 \\ p & 0 & r & 0 \\ p & r & 0 & r \\ 0 & 0 & r & 0 \end{pmatrix} \quad M_{G3} = \begin{pmatrix} 0 & p & p & 0 & q \\ p & 0 & r & 0 & 0 \\ p & r & 0 & r & 0 \\ 0 & 0 & r & 0 & 0 \\ q & 0 & 0 & 0 & 0 \end{pmatrix}$$

18

Ayrıt Ekleme



19

Apriori Temelli Algoritma

- Yaygın 1-altçizgelerin belirlenmesi
- Yeni yaygın altçizge oluşmayana kadar tekrarla
  - Aday oluşturma: Yaygın (k-1) altçizgeleri kullanarak k-altçizgeler oluşturma
  - Aday azaltma: (k-1)-altçizgesi yaygın olmayan adayların silinmesi
  - Destek hesaplaması: Kalan adaylar için destek değerinin hesaplanması
  - Aday eleme: min_sup değerini sağlamayan adayların silinmesi

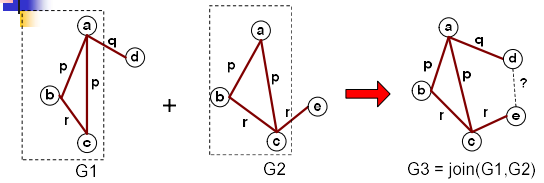
20

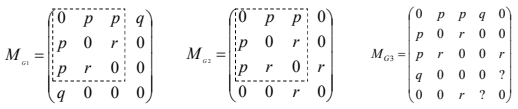
Aday Oluşturma

- Apriori yöntemi:
  - İki (k-1)-yaygın niteliklerin birleştirilmesi sonucu sadece bir k-yaygın nitelikler oluşur.
- Altçizge madenciliği
  - İki (k-1)-yaygın altçizgenin birleştirilmesi sonucu birden fazla k-yaygın altçizge oluşur
    - iki (k-1)-yaygın altçizgenin birleştirilmesi için (k-2)-altçizgelerinin aynı olması gerekir (çekirdek)

21

Düğüm Ekleme: Farklı Aday Oluşması

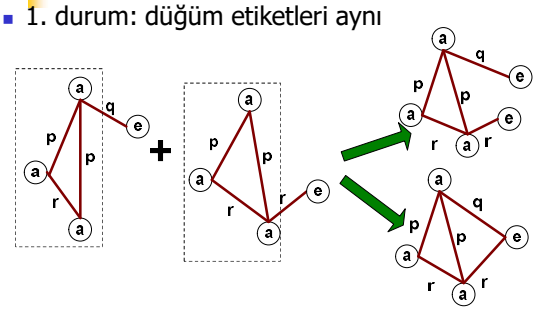




22

Ayrıt Ekleme: Farklı Aday Oluşması

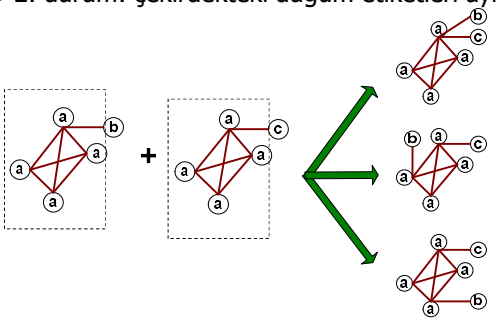
- 1. durum: düğüm etiketleri aynı



23

Ayrıt Ekleme: Farklı Aday Oluşması

- 2. durum: çekirdekteki düğüm etiketleri aynı



24

Çizge Madenciliği

- Jun Huan, Wei Wang, Jan Prins, and Jiong Yang, *SPIN: Mining maximal frequent subgraphs from graph databases*. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), 2004.
- X. Yan, X. J. Zhou, and J. Han, *Mining Closed Relational Graphs with Connectivity Constraints*, Proc. 2005 Int. Conf. on Knowledge Discovery and Data Mining (KDD'05), Chicago, IL, Aug. 2005.

25

Sıralı Diziler

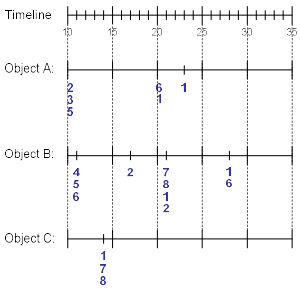
- İlişkilendirme kuralları hareketlerde yer alan nitelikler arasındaki ilişkiyi bulmayı hedefler. Zaman bilgisini kullanmaz.

- Sıralı dizi analizi zaman bilgisini kullanarak hareketlerin sırasına göre nitelikler arasındaki ilişkiyi bulur.

26

Sıralı Dizi Verisi

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



- Örnek sıralı dizi verisi:
  - müşteri hareketleri
  - tıbbi veriler
  - web verileri
  - telefon görüşme kayıtları
  - DNA dizileri

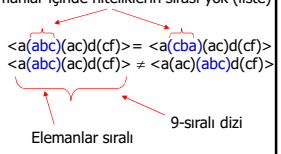
27

Sıralı Dizi Tanımı

Sıralı dizi veritabanı

SID	sıralı dizi
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Elemanlar içinde niteliklerin sırası yok (liste)



- Elemanlar nitelikler kümesi, nitelikler sıralı değildir, genelde alfabetik sırada sıralanırlar (abc)
- Elemanların sıralı listesi (<a(abc)(ac)d(cf)>)
- Elemanlar zaman veya yer ile nitelendirilir
- Sıralı dizinin uzunluğu |s|: Sıralı diziyi oluşturan nitelikler sayısı
- k-sıralı dizi: k niteliği olan sıralı dizi

28

Sıralı Dizi Örnekleri

- Web verileri:

< (Homepage) (Electronics) (Digital Cameras) (Canon Digital Camera) (Shopping Cart) (Order Confirmation) (Return to Shopping) >

- Kütüphaneden alınan kitaplar:

<(Fellowship of the Ring) (The Two Towers) (Return of the King)>

- DNA dizileri

ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCC

29

Alt Dizi Tanımı

- $\beta=<b_1 b_2 \ldots b_m>$ sıralı dizisi içinde $\alpha=<a_1 a_2 \ldots a_n>$ sıralı dizinin bulunabilmesi için ($m \geq n$):
  - $i1 < i2 < \ldots <$ in koşulunda $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$
    - <a(abc)(ac)d(cf)> sıralı dizinin bir alt dizisi: <a(bc)dc>
    - <ad>, <(ad)c(bc)(ae)> sıralı dizinin alt dizisi değil
- α dizisi için β "super sequence" olarak tanımlanır.
- s dizisi hiç bir dizinin alt dizisi değilse "maximal" olarak tanımlanır.
- Bir alt dizinin destek değeri: Alt dizinin bulunduğu sıralı dizilerin toplam sıralı dizi sayısına oranı
- Sıralı dizi örüntüsü: Destek değeri en küçük destek değerine (minsup) eşit ya da büyük olan alt diziler

30

Sıralı Dizi Analizi

- Sıralı dizi veritabanında en az *minsup* destek değerine sahip alt dizilerin bulunması (sıralı dizi örüntülerinin bulunması)

sıralı dizi veritabanı

SID	sıralı dizi
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

$\alpha 1$= <(ab)c> minsup($\alpha 1$)=2

31

Zorluklar

- Veri kümesi içinde çok fazla sayıda sıralı dizi örüntüsü bulunabilir. Sıralı dizi bulmak için kullanılan algoritmanın:
  - bütün sıralı dizi örüntülerini bulması gerekir
  - Veri kümesini az sayıda tarayarak sonuç üretmesi gerekir
    - ölçeklenebilir
  - Değişik veri tipleri üzerinde çalışabilmesi gerekir

32

Sıralı Dizi Örüntülerinin Temel Özelliği

- Apriori:
  - Sıralı dizi s, minsup değerini sağlamıyorsa, s'in hiçbir üst kümesi de minsup değerini sağlamaz.
  - Örnek: <hb> minsup değerini sağlamıyor → <hab> ve <(ah)b> sıralı dizileri de minsup değerini sağlamaz.

SID	Sıralı dizi
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

minsup =2

33

Apriori All

- Rakesh Agrawal and Ramakrishnan Srikant, *Mining Sequential Patterns,* In Proc. of 11h Int. Conf. on Data Engineering, Taipei, Taiwan, 1995.
- Apriori algoritmasının yaklaşımını kullanır
- Aday sıralı dizileri bulmak için Apriori algoritmasının Apriori-generate fonksiyonunu kullanır. Aday sıralı dizilerin minsup değerlerini bulmak için sıralı dizi verisini tarar.

34

Apriori All Algoritması

- 5 aşamadan oluşur
  1. Sıralama aşaması (sort phase)
     Hareketlerden oluşan veri kümesinden sıralı dizileri oluşturma
  2. Yaygın nitelikler kümesi aşaması (large itemset phase)
     Apriori ile yaygın nitelikler kümeleri bulunur
  3. Dönüştürme aşaması (Transformation phase)
     Her yaygın nitelikler kümesi bir tamsayı ile eşleştirilir
  4. Sıralı dizi aşaması (sequence phase)
     Apriori kullanılarak bütün sıralı dizi örüntüleri çıkarılır
  5. Maximal aşaması (maximal phase)
     Maximal olmayan sıralı diziler elenir

35

Apriori All Algoritması (1)

- C_k:k uzunluğunda yaygın sıralı dizi adayları
- L_k:k uzunluğunda yaygın sıralı diziler

L_1 = {1-frequent sequences}; //2. aşama sonucu
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each sequence c in database do
    increment the count of all candidates in C_{k+1} that are contained in c
  L_{k+1} = candidates in C_{k+1} with min_support
end
return $\cup_k L_k$;

- *Aday oluşturma:* L_{k-1} kendisiyle birleştirme
  insert into C_k
  select $p.item_1$, $p.item_2$, …, $p.item_{k-1}$, $q.item_{k-1}$
  from L_{k-1} p, L_{k-1} q
  where $p.item_1=q.item_1$, …,
  $p.item_{k-2}=q.item_{k-2}$

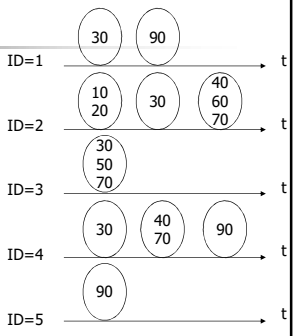
Örnek:
{1,2,3}x{1,2,4}= {1,2,3,4} ve {1,2,4,3}

Tanım : Dizinin uzunluğu, dizi içindeki nitelikler kümesi sayısıdır.

36

Örnek: Sıralı Dizi Veritabanı

ID	Zaman	Ürün
1	1	30
1	2	90
2	1	10,20
2	2	30
2	3	40,60,70
3	1	30,50,70
4	1	30
4	2	40,70
4	3	90
5	1	90



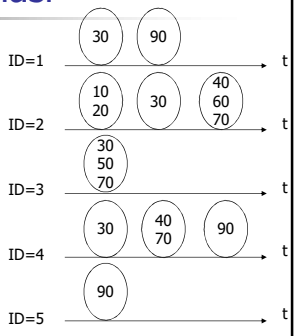
minsup=%40 =>en az iki farklı ID

Sonuç: <(30)(90)> (ID=1,4) <(30)(40 70)> (ID=2,4)

<30><40><70><90>, <(30)(40)> <(30)(70)> <(40 70)> sıralı dizi örüntüsü değil

Sıralama Aşaması

ID	Zaman	Ürün
1	1	30
1	2	90
2	1	10,20
2	2	30
2	3	40,60,70
3	1	30,50,70
4	1	30
4	2	40,70
4	3	90
5	1	90



- ID
  - Zaman

Yaygın Nitelikler Kümesi Aşaması

- Apriori algoritması kullanılarak yaygın nitelikler belirlenir.
  - Sıralı dizi örüntülerinde geçen nitelikler yaygın nitelikler olması gerekiyor
- Destek değerinin farklı hesaplanması gerekiyor
  - Bir niteliğin kaç farklı "ID" için bulunduğu

ID	Zaman	Ürün
1	1	30
1	2	90
2	1	10,20
2	2	30
2	3	40,60,70
3	1	30,50,70
4	1	30
4	2	40,70
4	3	90
5	1	90

minsup=%40 => en az iki farklı ID

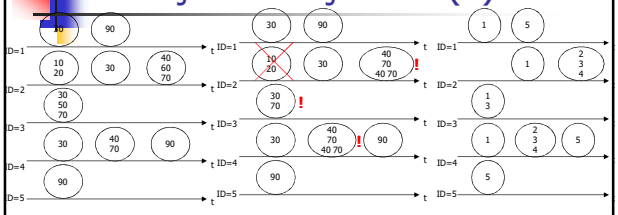
Yaygın nitelikler kümeleri:
{30}{40} {70}{40 70} {90}

Dönüştürme Aşaması (1)

- Her yaygın nitelikler kümesi bir tamsayı ile eşleştirilir
  - İki farklı nitelikler kümesinin veritabanı içinde eşit zamanda taranması için
- Hareketler tamsayılar ile temsil edilir

Nitelikler	Tamsayı
{30}	1
{40}	2
{70}	3
{40 70}	4
{90}	5

Dönüştürme Aşaması (2)



ID	Sıralı Diziler
1	<(30) (90)>
2	<(10 20)30(40 60 70)>
3	<(30 50 70)>
4	<30 (40 70) 90>
5	<(90)>

Nitelikler	Tamsayı
{30}	1
{40}	2
{70}	3
{40 70}	4
{90}	5

ID	Tamsayı Eşleneği
1	<{1} {5 }>
2	<{1}{2 3 4 }>
3	<{1 3 }>
4	<{1} { 2 3 4 } { 5 }>
5	<{5 }>

Sıralı Dizi Aşaması

- Nitelikler kümesi kullanılarak istenen destek değerini sağlayan sıralı diziler bulunuyor
- Apriori algoritması ile aynı yaklaşım

Nitelikler	Tamsayı
{30}	1
{40}	2
{70}	3
{40 70}	4
{90}	5

minsup=%40

ID	Tamsayı Eşleneği
1	<{1} {5}>
2	<{1}{2 3 4}>
3	<{1 3}>
4	<{1} { 2 3 4} { 5}>
5	<{5}>

Nitelikler	Destek
{1}	4
{2}	2
{3}	3
{4}	2
{5}	3
({1} {2})	2
({1} {3})	3
({1} {4})	2
({1} {5})	2

<30>
<40>
<70>
<40 70>
<90>
<30><40>
<30><70>
<30 (40 70)>
<30><90>

Maximal Aşaması

- Maximal olmayan sıralı dizileri sil
  - Başka bir sıralı dizinin altdizisi olan dizileri sil

for k=n; k=1; k- -
  for each k-sequence s_k do
    delete all subsequences of s_k

~~<30>~~
~~<40>~~
~~<70>~~
~~<40 70>~~
~~<90>~~
~~<30><40>~~
~~<30><70>~~
<30 (40 70)>
<30><90>

GSP (A Generalized Sequential Pattern Mining Algorithm)

- Ramakrishnan Srikant, Rakesh Agrawal, *Mining Sequential Patterns: Generalizations And Performance Improvements*, Proc. 5th Int. Conf. Extending Database Technology, EDBT'96
  - 1. Aşama:
    - Veritabanını bir kez tarayarak 1 nitelikten oluşan sıralı dizi örüntülerini bulunur
  - 2. Aşama: Yeni bir sıralı dizi örüntüsü bulunmayıncaya kadar tekrarla
    - Aday oluşturma: (k-1). adımda bulunan sıralı dizi örüntüleri birleştirilerek k uzunluklu sıralı dizi örüntü adayları bulunur
    - Aday azaltma: (k-1) alt dizileri minsup değerini sağlamayan adaylar elenir
    - Destek değeri hesaplama: Kalan sıralı dizi örüntü adayları için veritabanı taranarak destek değeri hesaplanır
    - Aday eleme: minsup değeri sağlamayan sıralı dizi örüntü adayları elenir.

GSP: Aday Oluşturma

- İlk durum (k=2): Bir uzunluklu sıralı dizilerden iki uzunluklu sıralı diziler oluşturma
  - $s_{11}=<i_1>$ ve $s_{12}=<i_2>$ sıralı dizlerin birleştirilmesi sonucunda $s_{21}= <i_1\ i_1>$, $s_{22}= <i_1\ i_2>$, $s_{23}= <i_2\ i_2>$, $s_{24}= <i_2\ i_1>$ ve $s_{25}= <(i_1\ i_2)>$ sıralı dizileri oluşur
- Genel durum (k>2):
  - k uzunluklu s_{ki} ve s_{kj} sıralı dizileri şu şartlar altında birleşir:
    - s_{ki} sıralı dizinin ilk niteliği ve s_{kj} sıralı dizinin son niteliği silindiğinde oluşan alt dizinin aynı olması
  - (k+1) uzunluklu sıralı dizi örüntü adayı, s_{ki} sıralı diziye s_{kj} sıralı dizinin son niteliğinin eklenmesi ile bulunur
    - s_{kj} sıralı dizinin son iki niteliği aynı elemana ait: s_{ki} sıralı dizinin son niteliği s_{ki} sıralı dizisinin son elemanının parçası olur
    - diğer durumda s_{kj} sıralı dizisinin son niteliği s_{ki} sıralı dizisinin ayrı bir elemanı olur.

Örnek

- s_1=<(a) (b c) (d)> ve s_2 =<(b c) (d e)> sıralı dizileri birleştirilirse
  - aday sıralı dizi <(a) (b c) (de)>
    - s_2 sıralı dizideki son iki nitelik (d ve e) aynı elemana ait
- s_1=<(a) (b c) (d)> ve s_2 =<(b c) (d)(e)> sıralı dizileri birleştirilirse
  - aday sıralı dizi < (a) (b c) (d)(e)>
    - s_2 sıralı dizideki son iki nitelik (d ve e) aynı elemana ait değil
- s_1 =<(a)(b f)(d)> ve s_2 =<(a) (b) (d e)> sıralı dizileri <(a) (b f) (de)> sıralı dizi örüntü adayı oluşturmak üzere birleştirilemez
  - s_2 =<(b f) (d x)> veya s_2 =<(b f) (d) (x)> şeklinde olmadığı için

Örnek

- Bir nitelikten oluşan dizilerin bulunması
- Adaylar
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Veritabanı taranarak destek sayıları hesaplanır

Aday	Des
<a>	**3**
<b>	**5**
<c>	**4**
<d>	**3**
<e>	**3**
<f>	**2**
~~<g>~~	1
~~<h>~~	1

min_sup =2

SID	Sıralı Dizi
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Örnek

- iki uzunluklu sıralı dizilerin bulunması
  - 51 aday

36

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

15

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Apriori özelliği kullanılmadan aday sayısı
8*8+8*7/2=92

Apriori aday sayısını %44.57 azaltıyor

Örnek

- 3 uzunluklu adayların bulunması
  - 2 uzunluklu sıralı dizilerden 3 uzunluklu adaylar oluşturma
    - Apriori özelliği kullanılıyor
    - <ab>, <aa> ve <ba> : 2 uzunluklu sıralı diziler → <aba> : 3 uzunluklu sıralı dizi
    - <(bd)>, <bb> ve <db>: 2 uzunluklu sıralı diziler → <(bd)b>: 3 uzunluklu sıralı dizi
  - 46 aday oluşuyor
- 3 uzunluklu sıralı dizilerin bulunması
  - Veritabanı taranarak adayların destek sayıları hesaplanır
  - 46 adaydan 19 adet 3 uzunluklu sıralı dizi örüntüsü bulunur

GSP Örnek

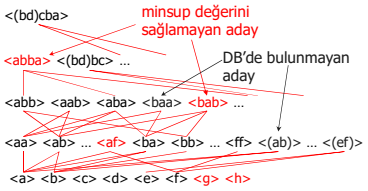
5. adım: 1 aday, 1 adet 5-sıralı dizi örüntüsü

4. adım: 8 aday, 6 adet 4-sıralı dizi örüntüsü

3. adım: 46 aday, 19 adet 3-sıralı dizi örüntüsü. 20 aday DB'de değil

2. adım: 51 aday, 19 adet 2-sıralı dizi örüntüsü. 10 aday DB'de değil

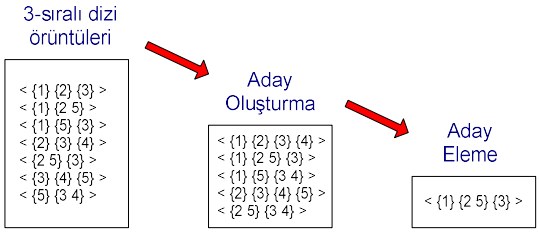
1. adım: 8 aday, 6 adet 1-sıralı dizi örüntüsü.



minsup =2

ID	Sıralı dizi
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

GSP Örnek



GSP Algoritmasında Problemler

- Çok sayıda sıralı dizi örüntü adayı oluşturulabilir
  - 1000 adet 1-sıralı dizi örüntülerinden $1000 \times 1000 + \frac{1000 \times 999}{2} = 1,499,500$ adet sıralı dizi örüntü adayı oluşur.
- Veritabanı birçok kez taranır
- Uzun sıralı dizileri belirlemek çok zor
  - sıralı diziler art arda eklenerek daha uzun sıralı diziler oluşuyor
  - 100-sıralı dizi oluşturmak için 10^{30} sıralı dizi gerekiyor.

$$\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}$$

Sıralı Dizi Örüntü Adayı Oluşturma

- Parçala ve çöz yaklaşımı
  - veritabanı yinelemeli küçültülür
  - oluşan her veritabanında sıralı dizi örüntüleri bulunur
- Algoritmalar:
  - J. Han J. Pei, B. Mortazavi-Asi, Q. Chen, U. Dayal and M.C. Hsu. *FreeSpan: Frequent pattern-projected sequential pattern mining*. KDD'00
  - J. Pei, J. Han, B. Mortazavi-Asi, H. Pinto, Q. Chen, U. Dayal, and MC. Hsu. *Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth*. In Proceedings of the 17th International Conference on Data Engineering, 2001.

FreeSpan Algoritması

- Örnek: Sıralı dizi veritabanı ve min_sup=2

Sıralı dizi Veritabanı
< (bd) c b (ac) >
< (bf) (ce) b (fg) >
< (ah) (bf) a b f >
< (be) (ce) d >
< a (bd) b c b (ade) >

- 1. Adım: 1-sıralı dizi örüntülerinin bulunması ve destek değeri küçülen sırada sıralanmaları (f-list)

f_list: b:5, c:4, a:3, d:3, e:3, f:2

FreeSpan Algoritması

- 2. Adım: Arama uzayının bölünmesi. Sıralı dizi veritabanı altkümelere bölünür
  - *f* niteliği bulunan alt diziler
  - *e* niteliği bulunan *f* niteliği bulunmayan
  - *d* niteliği bulunan *e* ve *f* nitelikleri bulunmayan
  - *a* niteliği bulunan *d*, *e* ve *f* nitelikleri bulunmayan
  - *c* niteliği bulunan *a*, *d*, *e* ve *f* nitelikleri bulunmayan
  - *b* niteliği bulunan
- Her alt dizi (α) için bu alt dizilerin bulunduğu sıralı dizilerden oluşan veritabanları, α-yansımalı veritabanları (α-projected database), oluşturulur.

55

FreeSpan Algoritması

- c niteliği bulunan *a*, *d*, *e* ve *f* nitelikleri bulunmayan sıralı diziler
  - <c>-yansımalı veritabanı: <b c b c>, <b c b>, <b c>, <b b c b>
  - c niteliği bulunan *a*, *d*, *e* ve *f* nitelikleri bulunmayan bütün alt dizilerin bulunması: <bc>: 4, <cb>: 3
  - Veritabanı yansıtılarak sıralı diziler bulunmaya devam edilir.

Sıralı dizi Veritabanı
< (bd) c b (ac) >
< (bf) (ce) b (fg) >
< (ah) (bf) a b f >
< (be) (ce) d >
< a (bd) b c b (ade) >

56