VERİ MADENCİLİĞİ
İlişkilendirme Kuralları

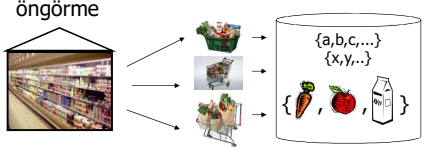
Prof. Dr. Şule Gündüz Öğüdücü
http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

İlişkilendirme Kuralları Madenciliği

- İlişkilendirme kuralı madenciliği
  - Veri kümesi içindeki yaygın örüntülerin, nesneleri oluşturan nitelikler arasındaki ilişkilerin bulunması
- İlişkilendirme kurallarını kullanma: veri içindeki kuralları belirleme
  - Hangi ürünler çoğunlukla birlikte satılıyor?
  - Kişisel bilgisayar satın alan bir kişinin bir sonraki satın alacağı ürün ne olabilir?
  - Yeni bir ilaca duyarlı olan DNA tipleri hangileridir?
  - Web dökümaları otomatik olarak sınıflandırılabilir mi?

İlişkilendirme Kuralları Bulma

- Bir niteliğin (veya nitelikler kümesinin) varlığını harekette bulunan başka niteliklerin varlıklarına dayanarak öngörme



hareketlerde sıkça birlikte yer alan ürünleri bulur

- Kural şekli: **"Gövde ⟶ Baş [destek, güven] "**

satın alma(x, "ekmek") ⟶ satın alma (x, "süt") [%0.6, %65]
öğrenci(x, "BLG"), kayıt(x, "VTYS") ⟶ not(x, "A") [%1, %75]

İlişkilendirme Kuralları Bulma

- Bütün niteliklerden oluşan küme $I=\{i_1, i_2,\ldots,i_d\}$
  - I={ekmek, süt, bira, kola, yumurta, bez}
- Hareket $T_j \subseteq I$, $Tj=\{i_{j1}, i_{j2},\ldots,i_{jk}\}$
  - T1={ekmek, süt}
- Hareketlerden oluşan veri kümesi $D=\{T_1, T_2,\ldots, T_N\}$

Market Alışveriş verisi

Hareket	Nitelikler
T1	Ekmek, Süt
T2	Ekmek, Bez, Bira, Yumurta
T3	Süt, Bez, Bira, Kola
T4	Ekmek, Süt, Bez, Bira
T5	Ekmek, Süt, Bez, Kola

Yaygın nitelikler:
Bez, bira
Süt, ekmek, yumurta, kola
Bira, ekmek, süt

Bulunan İlişkilendirme Kuralları
{Bez} → {Bira},
{Süt, Ekmek} → {Yumurta, Kola},
{Bira, Ekmek} → {Süt}

Yaygın Nitelikler

- nitelikler kümesi (Itemset)
  - Bir veya daha çok nitelikten oluşan küme
  - k-nitelikler kümesi (k-itemset): k nitelikten oluşan küme
    - 3-nitelikler kümesi: {Bez, Bira, Ekmek}
- Destek sayısı σ (Support count)
  - Bir nitelikler kümesinin veri kümesinde görülme sıklığı
    - σ({Süt, Ekmek, Bez}) = 2
- Destek s (Support)
  - Bir nitelikler kümesinin içinde bulunduğu hareketlerin toplam hareketlere oranı
    - s({Süt, Ekmek, Bez}) = 2 /5
- Yaygın nitelikler (Frequent itemset)
  - Destek değeri *minsup* eşik değerinden daha büyük ya da eşit olan nitelikler kümesi

İlişkilendirme Kuralları

- Veri kümesi $D=\{T_1, T_2,\ldots, T_N\}$
- en az, en küçük destek ve güven değerine sahip $X \rightarrow Y$ şeklinde kuralların bulunması
  - $X \subset I$, $Y \subset I$, $X \cap Y = \emptyset$
- Kuralları değerlendirme ölçütleri
  - destek (support) s: $X \cup Y$ nitelikler kümesinin bulunduğu hareket sayısının toplam hareket sayısına oranı

$$support(X \rightarrow Y) = \frac{\#X \cup Y}{N}$$

  - güven (confidence) c: $X \cup Y$ nitelikler kümesinin bulunduğu hareket sayısının X nitelikler kümesi bulunan hareket sayısına oranı

$$confidence(X \rightarrow Y) = \frac{\#(X \cup Y)}{\#X}$$

TID	Öğeler
T1	Ekmek, Süt
T2	Ekmek, Bez, Bira, Yumurta
T3	Süt, Bez, Bira, Kola
T4	Ekmek, Süt, Bez, Bira
T5	Ekmek, Süt, Bez, Kola

Örnek:
{Süt, Bez} ⇒ {Bira}

$$support = \frac{\sigma(sut, bira, bez)}{|T|} = \frac{2}{5} = 0,4$$

$$confidence = \frac{\sigma(sut, bira, bez)}{\sigma(sut, bez)} = \frac{2}{3} = 0,67$$

İlişkilendirme Kuralları Oluşturma

- İlişkilendirme kuralları madenciliğinde temel amaç: *D* hareket kümesinden kurallar oluşturmak
  - kuralların destek değeri, belirlenen en küçük destek (*minsup*) değerinden büyük ya da eşit olmalı
  - kuralların güven değeri, belirlenen en küçük güven (*minconf*) değerinden büyük ya da eşit olmalı
- Brute-force yaklaşım
  - Olası bütün kuralları listele
  - Her kural için destek ve güven değeri hesapla
  - *minsup* ve *minconf* eşik değerlerinden küçük destek ve güven değerlerine sahip kuralları sil
  - hesaplama maliyeti yüksek

$$R = 3^d - 2^{d+1} + 1$$

İlişkilendirme Kuralları Özellikleri

TID	Öğeler
1	Ekmek, Süt
2	Ekmek, Bez, Bira, Yumurta
3	Süt, Bez, Bira, Kola
4	Ekmek, Süt, Bez, Bira
5	Ekmek, Süt, Bez, Kola

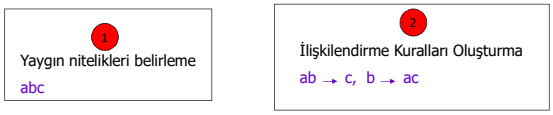
Örnek Kurallar:

{Süt, Bez} → {Bira} (s=0.4, c=0.67)
{Süt, Bira} → {Bez} (s=0.4, c=1.0)
{Bez, Bira} → {Süt} (s=0.4, c=0.67)
{Bira} → {Süt, Bez} (s=0.4, c=0.67)
{Bez} → {Süt, Bira} (s=0.4, c=0.5)
{Süt} → {Bez, Bira} (s=0.4, c=0.5)

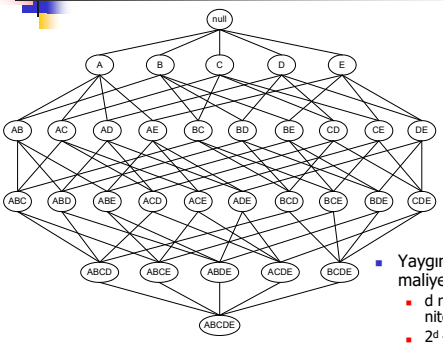
- Aynı nitelikler kümesinin ikili bölünmesine (binary partition) ait kurallar
  - {Süt, Bez, Bira}
- Aynı nitelikler kümesinden oluşan kuralların destek değerleri aynı, güven değerleri farklı
- Kurallar için destek ve güven şartları ayrı değerlendirilebilir

İlişkilendirme Kuralları Oluşturma

- İki adımda gerçeklenir
  1. **Yaygın nitelikleri belirleme:**
     - destek değeri *minsup* değerinden büyük ya da eşit olan nitelikler kümelerini bulma
  2. **Kural Oluşturma:**
     - Güven değeri *minconf* değerinden büyük ya da eşit olan ve yaygın niteliklerin ikili bölünmeleri olan kurallar oluşturma
     - Güçlü kurallar



Yaygın nitelik Adayları Oluşturma



- Yaygın nitelikleri bulmak maliyetli
  - d nitelik için 2^d -1 yaygın nitelik oluşabilir
  - 2^d -1 yaygın nitelik adayı

Yaygın Nitelik Oluşturma

- Brute-Force Yaklaşım
  - Her yaygın nitelik adayı için veri kümesini taranarak hareketlerde yaygın nitelik adayı bulunup bulunmadığı kontrol edilir

TID	Öğeler
1	Ekmek, Süt
2	Ekmek, Bez, Bira, Yumurta
3	Süt, Bez, Bira, Kola
4	Ekmek, Süt, Bez, Bira
5	Ekmek, Süt, Bez, Kola

N

w

Yaygın nitelik adayları

M

- yaygın nitelik adayları için destek değeri bulunur
- Destek değeri *minsup* değerine eşit yada büyük olanlar yaygın nitelikler
- karmaşıklığı: O(NMw), $M=2^d - 1 \Rightarrow$ hesaplaması maliyetli

Yaygın nitelik Oluşturma Yöntemleri

- Yaygın nitelik aday sayısını (M) azaltma
  - Örnek: Apriori algoritması
- Hareket sayısını (N) azaltma
- Veri kümesi tarama sayısını (MN) azaltma

Apriori Algoritması

- Apriori yöntemi
  - Rakesh Agrawal, Ramakrishnan Srikant, *Fast Algorithms for Mining Association Rules,* Proc. 20th Int. Conf. Very Large Data Bases, VLDB'94
  - Heikki Mannila, Hannu Toivonen, Inkeri Verkamo, *Efficient Algorithms for Discovering Association Rules* . AAAI Workshop on Knowledge Discovery in Databases (KDD-94).
- Temel yaklaşım:

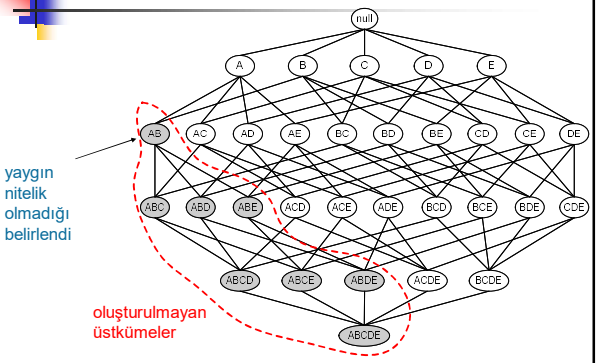
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Bir nitelikler kümesinin destek değeri altkümesinin destek değerinden büyük olamaz
  - anti-monotone özellik

Apriori Algoritması

- Bir yaygın nitelikler kümesinin herhangi bir altkümesi de yaygın nitelikler kümesidir
  - {Süt, Bez, Bira} kümesi yaygın nitelikler kümesi ise {Süt, Bez} kümesi de yaygın nitelikler kümesidir
  - {Süt, Bez, Bira} nitelikler kümesi olan her harekette {Süt, Bez} kümesi de vardır
- Yaygın nitelik aday sayısını azaltma yöntemi: Yaygın nitelikler kümesi olmayan bir kümenin üst kümeleri yaygın nitelik adayı olarak oluşturulmaz (destek değeri hesaplanmaz)
- Yöntem:
  - k-yaygın nitelikler kümesinden (k+1) yaygın nitelik adayları oluştur
  - yaygın nitelik adayları için destek değeri hesapla

Apriori Yaygın Nitelik Azaltma Yöntemi



Apriori Algoritması: Örnek

Nitelik	Sayı
Ekmek	4
Kola	2
Süt	4
Bira	3
Bez	4
Yumurta	1

1-nitelikler kümesi

Nitelikler kümesi	Sayı
{Ekmek,Süt}	3
{Ekmek,Bira}	2
{Ekmek,Bez}	3
{Süt,Bira}	2
{Süt,Bez}	3
{Bira,Bez}	3

2-nitelikler kümesi

(içinde kola ya da yumurta olan nitelikler kümesini yaygın nitelik adayı olarak oluşturmaya gerek yok)

minsup = 3

3-nitelikler kümesi

Nitelikler Kümesi	Sayı
{Ekmek, Süt, Bez}	3

Her nitelikler kümesi için destek hesaplansaydı,
$C_1 + C_2 + C_3 = 6 + 15 + 20 = 41$
Yaygın nitelik sayısı azaltılınca,
6 + 6 + 1 = 13

Apriori Algoritması

- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  C_{k+1} = candidates generated from L_k;
  for each transaction t in database do
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

Yaygın Nitelikler Adayı Oluşturma

- L_{k-1} kümesinde nitelikler alfabetik sıralanmış olsun
- 1. adım: L_{k-1} kendisiyle birleştirme
  insert into C_k
  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
  from L_{k-1} p, L_{k-1} q
  where $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- 2. adım: Eleme
  forall *itemsets c in* C_k do
    forall *(k-1)-subsets s of c* do
      if (*s is not in* L_{k-1}) **then delete** *c* **from** C_k

Örnek: Yaygın nitelikler Adayı Oluşturma

- Yaygın nitelik adayları oluşturma:
  - L_k kendisiyle birleştirilir (self join)
  - eleme
- Örnek:
  - L_3={*abc, abd, acd, ace, bcd*}
  - Kendisiyle birleştirme: L_3*L_3
    - *abc* ve *abd* nitelikler kümesinden *abcd*
    - *acd* ve *ace* nitelikler kümesinden *acde*
  - Eleme
    - *ade* L_3 kümesinin bir elemanı olmadığından *acde* yaygın niteliklere dahil edilmez
  - C_4={*abcd*}

19

Yaygın niteliklerden İlişkilendirme Kuralları Oluşturma

- Sadece güçlü ilişkilendirme kuralları oluşuyor
- Yaygın nitelikler *minsup* değerini sağlıyor
- Güçlü ilişkilendirme kuralları *minconf* değerini sağlıyor.
- Güven $(A \rightarrow B) = \text{Prob}(B|A) = \frac{\text{Destek}(A \cup B)}{\text{Destek}(A)}$
- Yöntem:
  - Her yaygın nitelikler kümesi *f*'in altkümelerini oluştur
  - Her altküme *s* için, $s \rightarrow (f-s)$ ilişkilendirme kuralı oluştur eğer:

  destek(f) / destek(s) ≥ minconf ise

20

İlişkilendirme Kuralları Oluşturma

- L yaygın niteliklerden f⊂L altkümelerinin bulunması
  - f→L-f kurallarının en küçük güven değeri koşulunu sağlaması gerekir
- Eğer {A,B,C,D} yaygın nitelikler ise olası ilişkilendirme kuralları

  ABC →D, ABD →C, ACD →B, BCD →A,
  A →BCD, B →ACD, C →ABD, D →ABC
  AB →CD, AC → BD, AD → BC, BC →AD,
  BD →AC, CD →AB
- |L| = k için $2^k - 2$ ilişkilendirme kuralı adayı vardır
  - L → ∅ ve ∅ → L kuralları geçerli kurallar değildir

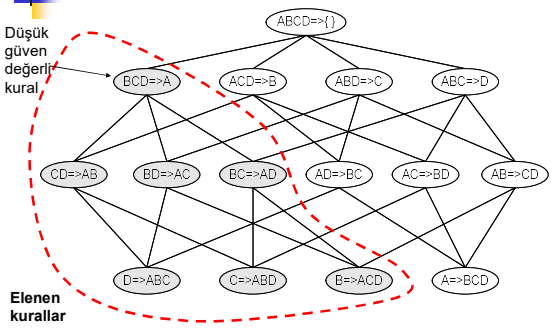
21

İlişkilendirme Kuralları Oluşturma

- İlişkilendirme kurallarının güven değerlerinin anti-monotone özelliği yok
  - c(ABC →D) değeri c(AB →D) değerinden küçük ya da büyük olabilir
- Aynı yaygın nitelikler kümesinden L = {A,B,C,D} oluşan ilişkilendirme kurallarının güven değerleri için anti-monotone özelliği var
  - c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)
  - İlişkilendirme kuralının solunda bulunan nitelik sayısı büyük olan kuralların güven değerleri de büyüktür.

22

Apriori Algoritması için Kural Oluşturma

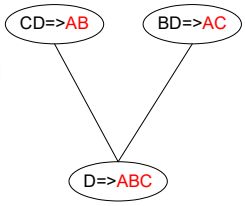


23

Apriori Algoritması için Kural Oluşturma

- Sağ taraflarındaki niteliklerin önekleri aynı olan iki kural birleştirilir
- (CD=>AB,BD=>AC) kuralları birleştirilerek D => ABC kural adayı oluşturulur.
- AD=>BC kuralının güven değeri büyük değilse

  D => ABC kuralı elenir



24

Destek Değerinin Etkisi

- minsup değeri büyük belirlenirse veri kümesinden bazı örüntüler elde edilmeyebilir:
  - veri kümesinde daha az bulunan
  - önemli bilgi taşıyan
- minsup değeri küçük belirlenirse
  - yöntem karmaşıklaşır
  - çok fazla sayıda yaygın nitelikler kümesi elde edilir
- Tek bir destek değeri her zaman yeterli olmayabilir.

25

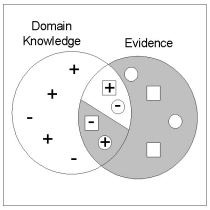
Bulunan Kuralların Önemi

- Tarafsız Ölçüt:
  - Örüntüler veri kümesinden elde edilen istatistiklere göre sıralanır
    - güven, destek, Jaccard, Gini, ...
- Taraflı Ölçüt
  - Örüntüler kullanıcının değerlendirmesine göre sıralanır
    - Bulunan örüntü kullanıcının beklentisi dışındaysa ilginçtir (Silberschatz & Tuzhilin)

26

Bulunan Kuralların Önemi

- Taraflı değerlendirme için kullanıcının beklentisinin modellenmesi gerekir



+ Yaygın olduğu düşünülen örüntü
- Yaygın olmadığı düşünülen örüntü
Yaygın olduğu bulunan örüntü
Yaygın olmadığı bulunan örüntü

Beklenen örüntüler
Beklenmeyen örüntüler

27

Apriori Algoritmasını Geliştirme

- Veritabanı tarama sayısını azaltma
  - k-yaygın nitelikler kümesi için veritabanı k kez taranıyor
- Hash yöntemi ile veritabanı tarama sayısını azaltma
- Veritabanındaki hareket sayısını ve hareketlerdeki nitelik sayısını azaltma
  - yaygın olmayan niteliklerin veritabanında yer almasına gerek yok
- Arama uzayını parçalama

28

Eniyilime: Hash Ağacı

- Her yaygın nitelik adayının destek değerinin belirlenmesi için veri kümesi taranır
  - Yaygın nitelik adayı sayısı çok büyük olabilir
- Karşılaştırma sayısını azaltmak için yaygın nitelik adayları hash ağacında saklanır
  - Her hareket bütün yaygın niteliklerle karşılaştırılmak yerine hash ağacının ilgili bölümündeki yaygın nitelikler adayları ile karşılaştırır

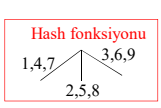
29

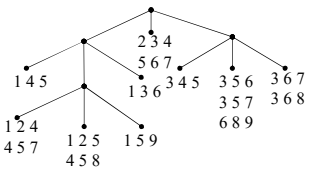
Hash Ağacı Oluşturma

- 15 adet 3-nitelikler kümesi

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

- Hash fonksiyonu kullanılarak bir hash ağacında saklanıyor
  - h(p) = p mod 3
- max yaprak boyutu: bir yaprakta bulunabilecek nitelik kümesi sayısı

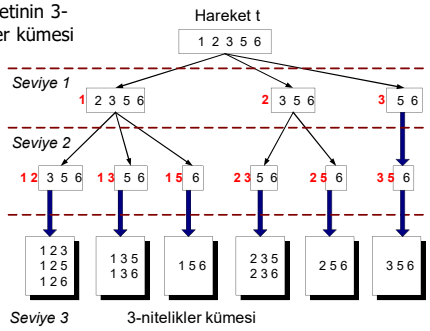




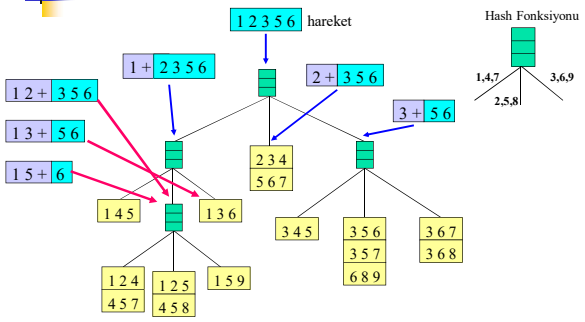
30

Bir Harekette Bulunabilecek Yaygın nitelikleri Belirleme

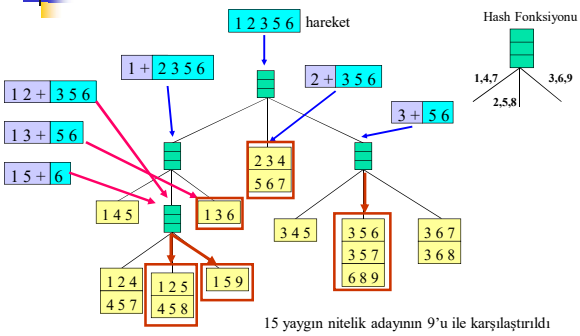
- t hareketinin 3-nitelikler kümesi



Bir Harekette Bulunabilecek Yaygın nitelikleri Hash Ağacı ile Belirleme



Bir Harekette Bulunabilecek Yaygın nitelikleri Hash Ağacı ile Belirleme



15 yaygın nitelik adayının 9'u ile karşılaştırıldı

Karmaşıklığı Etkileyen Etmenler

- Veri kümesindeki boyut sayısı (nitelik sayısı)
  - her niteliğin destek değerini saklamak için daha fazla saklama alanına ihtiyaç var
- Veri kümesinin büyüklüğü
  - Veri kümesi tarandığı için veri kümesindeki hareket sayısının fazla olması algoritmanın çalışma süresini uzatır
- Hareketlerin ortalama büyüklüğü (nitelik sayısı)
  - Yoğun veri kümelerinde hareketlerdeki nitelik sayısı fazla olur
  - Yaygın nitelik kümelerindeki nitelik sayısı daha fazladır
- En küçük destek değerini belirleme
  - En küçük destek değerini küçültme daha fazla sayıda yaygın nitelik oluşmasına neden olur
  - Yaygın nitelik adayı sayısının ve yaygın nitelik kümesindeki niteliklerin sayısının artmasına neden olur

Yaygın Nitelikleri Belirlemede Sorunlar

- Sorunlar
  - Veri kümesinin birçok kez taranması
  - Yaygın nitelikler aday sayısının fazlalığı
    - Yaygın nitelikleri bulmak için $i_1 i_2 \ldots i_{100}$
    - veri kümesini tarama sayısı: 100
    - Aday sayısı: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{1\,0\,0}{1\,0\,0} = 2^{100}-1 = 1.27*10^{30}$!
  - Yaygın nitelikler adayları için destek değerinin hesaplanması
- Çözümler
  - Veri kümesi tarama sayısını azaltma (S. Brin R. Motwani, J. Ullman, and S. Tsur. *Dynamic itemset counting and implication rules for market basket data*. In SIGMOD'97)
  - Yaygın nitelik aday sayısını azaltma (J. Park, M. Chen, and P. Yu. *An effective hash-based algorithm for mining association rules*. In SIGMOD'95)
  - Destek değerinin hesaplanmasını kolaylaştırma (M. Zaki et al. *New algorithms for fast discovery of association rules*. In KDD'97)
  - Yaygın nitelikler adayı oluşturmadan yaygın nitelikler bulunabilir. (J. Han, J. Pei, and Y. Yin, *Mining Frequent Patterns without Candidate Generation*. In SIGMOD'00.)

FP-Tree Algoritması

- Hareketlerden oluşan bir veri kümesinden destek değeri (ξ), kullanıcının belirlediği destek değerine eşit ya da daha büyük olan tüm yaygın örüntülerin bulunması

Giriş: DB:

TID	*Items bought*
100	{*f, a, c, d, g, i, m, p*}
200	{*a, b, c, f, l, m, o*}
300	{*b, f, h, j, o*}
400	{*b, c, k, s, p*}
500	{*a, f, c, e, l, p, m, n*}

Minimum support: ξ =3

Çıkış: bütün yagın örüntüler.,*f, a, …, fa, fac, fam, …*

Apriori Özelliği

- Apriori algoritmasının temeli:
  - (k-1) uzunluğundaki yaygın örüntülerden (L_{k-1}) k uzunluğunda yaygın örüntü adayı oluşturma (C_k)
  - veritabanı taranarak C_k içinde destek koşuluna sağlayan k uzunluklu yaygın örüntülerin bulunması L_k

TID	nitelikler
100	{f, a, c, d, g, i, m, p}
200	{a, b, c, f, l, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, f, c, e, l, p, m, n}

Apriori iteration	
C1	f,a,c,d,g,i,m,p,l,o,h,j,k,s,b,e,n
L1	f, a, c, m, b, p
C2	fa, fc, fm, fp, ac, am, ...bp
L2	fa, fc, fm, ...
...	

FP-Tree Algoritması

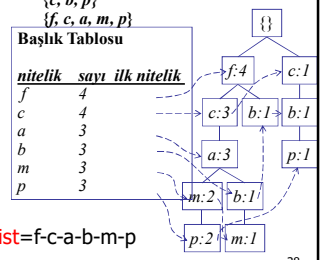
TID	nitelikler	(sıralı) yaygın
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

minsup = 3

1. DB bir kez taranarak 1-yaygın nitelik bulunuyor
2. Yaygın nitelikler destek sayısına göre büyükten küçüğe sıralanıor, f-list
3. DB bir kez daha taranarak FP-ağacı oluşturuluyor.

Başlık Tablosu

nitelik	sayı	ilk nitelik
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



F-list=f-c-a-b-m-p

FP-Tree Oluşturma

TID	nitelikler	(sıralı) yaygın
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, w}	{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}

minsup = 3

- 2 Adım:
  1. DB bir kez taranarak 1-yaygın nitelik bulunuyor
     L={f:4, c:4, a:3, b:3, m:3, p:3}
  2. Yaygın nitelikler destek sayısına göre büyükten küçüğe sıralanıyor: f-list. DB bir kez daha taranarak FP-ağacı oluşturuluyor.
     F-list=f-c-a-b-m-p

FP-Tree Oluşturma: Adım 1

- Veritabanını bir kez tarayarak L_1 oluşturma

TID	nitelik
100	{f, a, c, d, g, i, m, p}
200	{a, b, c, f, l, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, f, c, e, l, p, m, n}

Item	destek
f	4
c	4
a	3
b	3
m	3
p	3

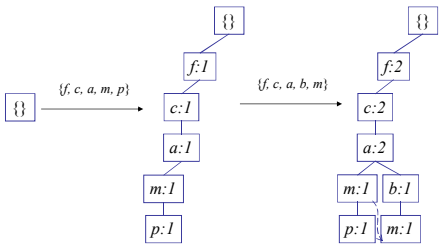
FP-Tree Oluşturma: Adım 2

- Veritabanını bir kez daha tarayarak her hareketteki niteliği f-list'e göre sırala

TID	nitelikler	(sıralı) yaygın
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, w}	{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}

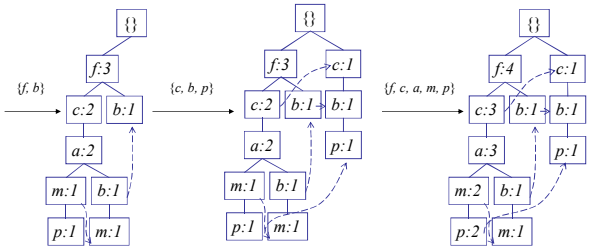
FP-Tree Oluşturma: Adım 2

- FP-Tree oluştur

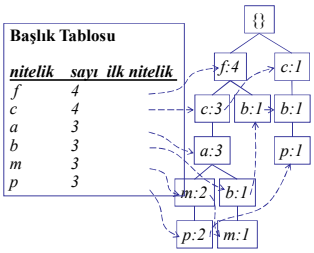


FP-Tree Oluşturma: Adım 2

- FP-Tree oluştur



Örnek: FP-Tree Oluşturma



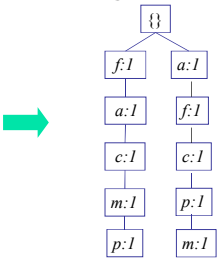
FP-Ağacının Özelliği

- En büyük avantajı veritabanını sadece 2 kez taraması
- Bütünlük
  - Yaygın nitelikleri bulmak için gerekli tüm bilgiyi barındırır
- Sıkıştırılmış
  - Yaygın olmayan nitelikler FP-ağacında bulunmaz
  - Destek sayısı daha büyük olan nitelikler köke daha yakın
  - Asıl veri kümesinden daha büyük değil

Özellikleri

- Hareketlerdeki niteliklerin f-list'e göre sıralanmadığı durum

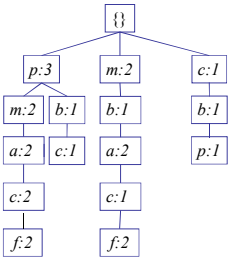
TID	sıralanmamış
100	{f, a, c, m, p}
500	{a, f, c, p, m}



Özellikleri

- f-list'teki niteliklerin destek değerinin artan sırada sıralandığı durum

TID	(ascended) frequent items
100	{p, m, a, c, f}
200	{m, b, a, c, f}
300	{b, f}
400	{p, b, c}
500	{p, m, a, c, f}



FP-Tree ile Yaygın Örüntülerin Bulunması

- Kısa yaygın niteliklere yeni nitelikler eklenerek daha uzun yaygın nitelikler elde etme
- 3 Adım
  1. Başlık tablosundaki her nitelik için nitelik koşullu örüntüleri oluştur
  2. Koşullu FP-Tree oluştur
  3. Yinelemeli olarak işlemi devam ettir. Koşullu FP-Tree yapısında tek dal kaldığında olası bütün örüntüleri listele
- Örnek:
  - "abc" bir yaygın nitelik
  - Veri kümesinde içinde "abc" nitelikleri bulunan hareketler (DB|abc)
  - DB|abc içinde d yaygın nitelik olarak bulunursa: "abcd" yaygın nitelik olarak belirlenir

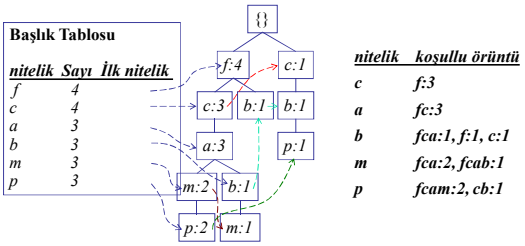
Örüntüleri ve Veri Kümesini Bölme

- Yaygın nitelikler f-listesine göre altkümelere bölünür
  - F-list=f-c-a-b-m-p
  - p niteliği bulunan örüntüler
  - m niteliği bulunan ancak p niteliği bulunmayan örüntüler
  - ...
  - c niteliği bulunan ancak a, b, m, p niteliği bulunmayan örüntüler
  - f niteliği bulunan örüntüler

49

Adım 1: Nitelik Koşullu Örüntü Oluşturma

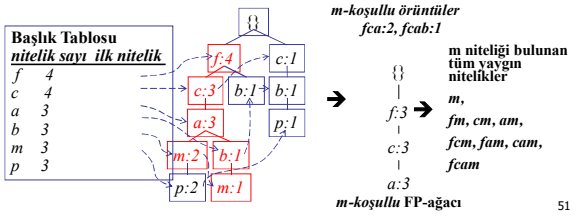
- Başlık tablosundan her niteliğin bulunduğu ilk düğüm bulunur.
- Bu düğümden başlayarak ağaçta niteliğin bulunduğu tüm düğümlere ulaşılır
- Kökten niteliğe kadar olan yollar bulunur (*transformed prefix paths)*



50

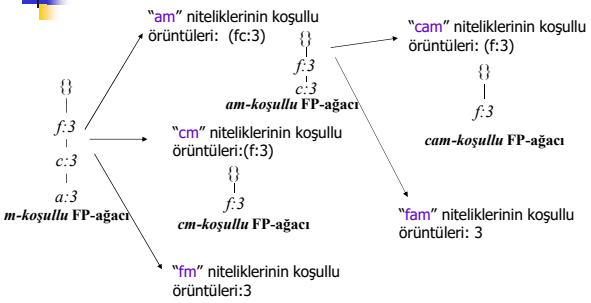
Adım 2: Nitelik Koşullu Örüntülerden Koşullu FP-Ağaçları Oluşturma

- Her koşullu örüntü için niteliklerin destek sayısı bulunur
- Yaygın nitelikler için koşullu FP-ağacı oluşturulur



51

Koşullu FP-Ağaçları



m:3, am:3, cm:3, fm:3, cam:3, fam:3, fcam:3, fcm:3

52

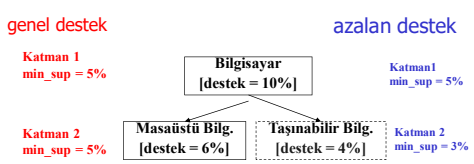
FP-Ağaçları ile Yaygın nitelikleri Bulma

- Yaygın niteliklere yinelemeli olarak yeni nitelikler ekleme
- Yöntem:
  - Her yaygın niteliğin koşullu örüntülerini ve koşullu FP-ağacını oluştur
  - İşlemi, yeni oluşturulan her koşullu FP-ağacı için tekrarla
  - Oluşturulan FP-ağaçlarında hiç nitelik bulunmayana kadar veya ağaçta tek bir dal kalana kadar işleme devam et
    - Ağaçta tek bir dal kaldığında yaygın nitelikler dalı oluşturan niteliklerin kombinasyonu

53

Çok Katmanlı İlişkilendirme Kurallarını Çıkarma

- Ürünler hiyerarşik yapıda olabilir
- Değişebilen destek değerleri
  - daha alt seviyelerdeki ürünlerin destek değeri daha düzşük olabilir



54

Gereksiz Kuralları Eleme

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

55