VERİ MADENCİLİĞİ
İlişkilendirme Kuralları

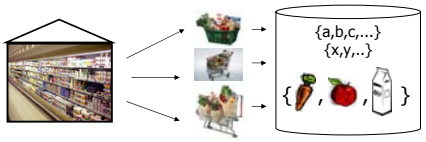
Yrd. Doç. Dr. Şule Gündüz Öğüdücü
http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

İlişkilendirme Kuralları Madenciliği

- İlişkilendirme kuralı madenciliği
  - Veri kümesi içindeki yaygın örüntülerin, nesneleri oluşturan öğeler 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 dokümaları otomatik olarak sınıflandırılabilir mi?

İlişkilendirme Kuralları Bulma

- Bir öğenin (veya öğeler kümesinin) varlığını harekette bulunan başka öğelerin varlıklarına dayanarak öngörme



hareketlerde sıkça birlikte yer alan öğeleri 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 öğelereden oluşan küme I={i_1, i_2,...,i_d}
  - I={ekmek, süt, bira, kola, yumurta, bez}
- Hareket $T_j \subseteq I$, $Tj=\{i_{j1}, i_{j2},...,i_{jk}\}$
  - T1={ekmek, süt}
- Hareketlerden oluşan veri kümesi $D=\{T_1, T_2,..., T_N\}$

Market Alışveriş verisi

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

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

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

Yaygın Öğeler

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

İlişkilendirme Kuralları

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

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

  - güven (confidence) c: $X \cup Y$ öğeler kümesinin bulunduğu hareket sayısının X öğeler 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

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

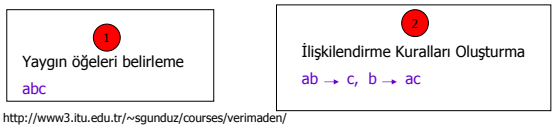
Örnek Kurallar:

{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)

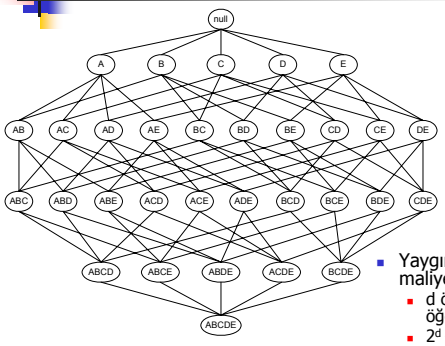
- Aynı öğeler kümesinin ikili bölünmesine (binary partition) ait kurallar
  - {Süt, Bez, Bira}
- Aynı öğeler kümesinden oluşan kuralların destek değerleri aynı, güven değerleri farklı
- Kurallar için destek ve güven şartları ayrı değerlendirilebilir

İlişkilendirme Kuralları Oluşturma

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



Yaygın Öğe Adayları Oluşturma



- Yaygın öğeleri bulmak maliyetli
  - d öğe için 2^d -1yaygın öğe oluşabilir
  - 2^d -1yaygın öğe adayı

Yaygın Öğe Oluşturma

- Brute-Force Yaklaşım
  - Her yaygın öğe adayı için veri kümesini taranarak hareketlerde yaygın öğe 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 öğe adayları

M

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

Yaygın Öğe Oluşturma Yöntemleri

- Yaygın öğe 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:

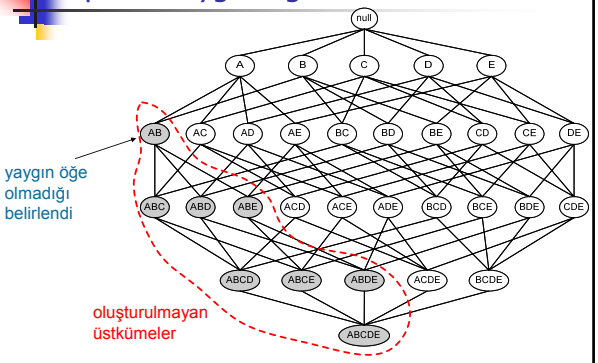
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

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

Apriori Algoritması

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

Apriori Yaygın Öğe Azaltma Yöntemi



Apriori Algoritması: Örnek

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

1-öğeler kümesi

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

2-öğeler kümesi

(kola ya da yumurta olan öğeler kümesini yaygın öğe adayı olarak oluşturmaya gerek yok)

minsup = 3

3-öğeler kümesi

Öğeler Kümesi	Sayı
{Ekmek, Süt, Bez}	3

Her öğeler kümesi için destek hesaplansaydı,
$C_1 + C_2 + C_3 = 6 + 15 + 20 = 41$
Yaygın öğe 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 Öğeler Adayı Oluşturma

- L_{k-1} kümesinde öğeler 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 Öğeler Adayı Oluşturma

- Yaygın öğe 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* öğeler kümesinden *abcd*
    - *acd* ve *ace* öğeler kümesinden *acde*
  - Eleme
    - *ade* L_3 kümesinin bir elemanı olmadığından *acde* yaygın öğelere dahil edilmez
  - C_4={*abcd*}

Yaygın Öğelerden İlişkilendirme Kuralları Oluşturma

- Sadece güçlü ilişkilendirme kuralları oluşuyor
- Yaygın öğeler *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 öğeler kümesi *f*'in altkümelerini oluştur
  - Her altküme *s* için, $s \rightarrow (f-s)$ ilişkilendirme kuralı oluştur eğer:

  $destek(f) / destek(s) \geq minconf$ ise

Apriori Algoritmasını Geliştirme

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

Eniyilime: Hash Ağacı

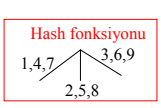
- Her yaygın öğe adayının destek değerinin belirlenmesi için veri kümesi taranır
  - Yaygın öğe adayı sayısı çok büyük olabilir
- Karşılaştırma sayısını azaltmak için yaygın öğe adayları hash ağacında saklanır
  - Her hareket bütün yaygın öğelerle karşılaştırılmak yerine hash ağacının ilgili bölümündeki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

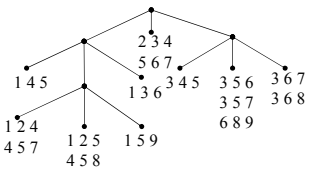
Hash Ağacı Oluşturma

- 15 adet 3-öğeler 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}

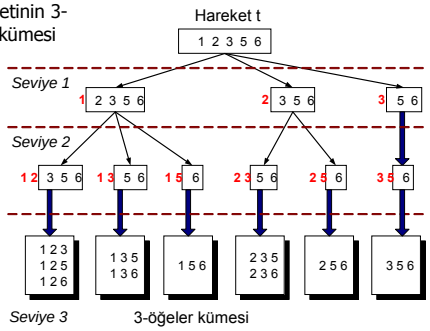
- Hash fonksiyonu kullanılarak bir hash ağacında saklanıyor
  - h(p) = p mod 3
- max yaprak boyutu: bir yaprakta bulunabilecek öğe kümesi sayısı



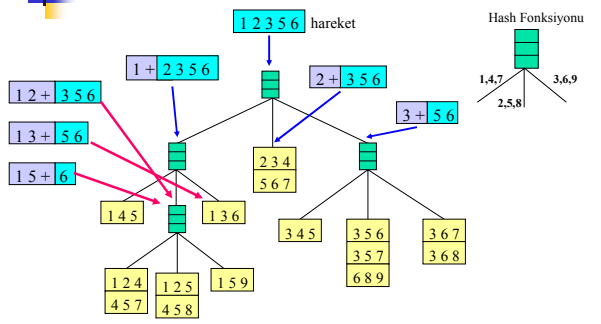


Bir Harekette Bulunabilecek Yaygın Öğeleri Belirleme

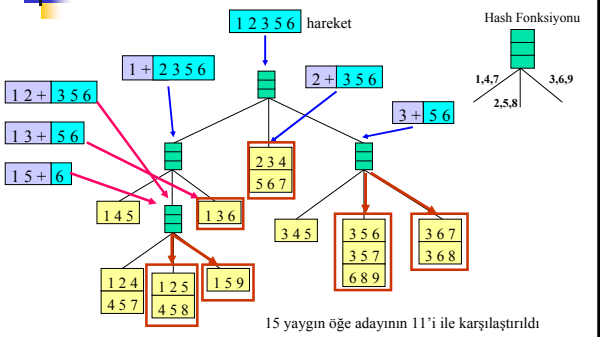
- t hareketinin 3-öğeler kümesi



Bir Harekette Bulunabilecek Yaygın Öğeleri Hash Ağacı ile Belirleme



Bir Harekette Bulunabilecek Yaygın Öğeleri Hash Ağacı ile Belirleme



15 yaygın öğe adayının 11'i ile karşılaştırıldı

Karmaşıklığı Etkileyen Etmenler

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

Yaygın Öğeleri Belirlemede Sorunlar

- Sorunlar
  - Veri kümesinin birçok kez taranması
  - Yaygın öğeler aday sayısının fazlalığı
    - Yaygın öğeleri bulmak için $i_1 i_2 \ldots i_{100}$
    - veri kümesini tarama sayısı: 100
    - Aday sayısı: $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100}-1 = 1.27*10^{30}$!
  - Yaygın öğeler 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 öğe 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 öğeler adayı oluşturmadan yaygın öğeler bulunabilir. (J. Han, J. Pei, and Y. Yin, *Mining Frequent Patterns without Candidate Generation*. In SIGMOD'00.)

Aday Oluşturmadan Yaygın Öğeleri Belirleme

- Kısa yaygın öğelere yeni öğeler eklenerek daha uzun yaygın öğeler elde etme
- Örnek:
  - "abc" bir yaygın öğe
  - Veri kümesinde içinde "abc" öğeleri bulunan hareketler (DB|abc)
  - DB|abc içinde d yaygın öğe olarak bulunursa: "abcd" yaygın öğe olarak belirlenir

FP-Tree Algoritması

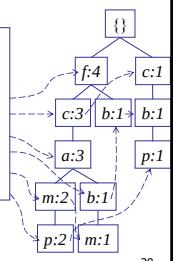
TID	*Öğeler*	*(sıralı) yaygın öğeler*
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

minsup = 3

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

Başlık Tablosu

Öğe	*sayı*	*ilk öğe*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



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

FP-Ağacının Özelliği

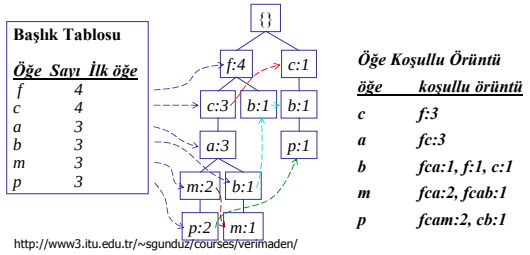
- Bütünlük
  - Yaygın öğeleri bulmak için gerekli tüm bilgiyi barındırır
- Sıkıştırılmış
  - Yaygın olmayan öğeler FP-ağacında bulunmaz
  - Destek sayısı daha büyük olan öğeler köke daha yakın
  - Asıl veri kümesinden daha büyük değil

Örüntüleri ve Veri Kümesini Bölme

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

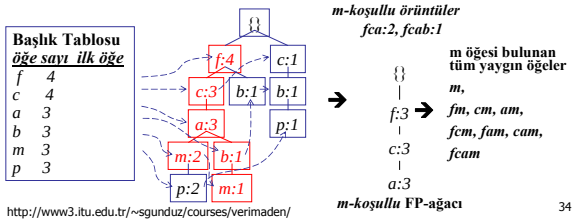
Öğe Koşullu Örüntü Oluşturma

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

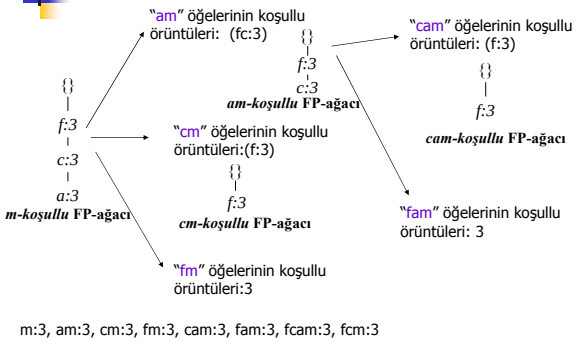


Öğe Koşullu Örüntülerden Koşullu FP-Ağaçları Oluşturma

- Her koşullu örüntü için öğelerin destek sayısı bulunur
- Yaygın öğeler için koşullu FP-ağacı oluşturulur



Koşullu FP-Ağaçları



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

FP-Ağaçları ile Yaygın Öğeleri Bulma

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

İlişkilendirme Kuralları Oluşturma

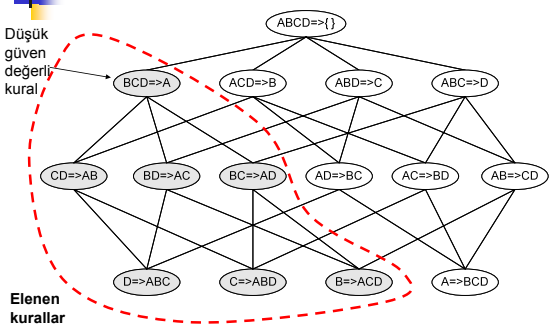
- L yaygın öğelerden f⊂L altkümelerinin bulunması
  - f→L-f kurallarının en küçük güven değeri koşulunu sağlaması gerekir
- Eğer {A,B,C,D} yaygın öğeler ise olası ilişkilendirme kuralları

  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

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 öğeler 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 öğe sayısı büyük olan kuralların güven değerleri de büyüktür.

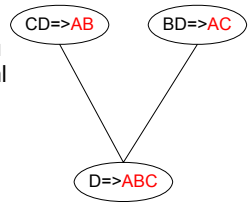
Apriori Algoritması için Kural Oluşturma



Apriori Algoritması için Kural Oluşturma

- Sağ taraflarındaki öğelerin 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



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 öğeler kümesi elde edilir
- Tek bir destek değeri herzaman yeterli olmayabilir.

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)

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

http://www3.itu.edu.tr/~sgunduz/courses/verimaden/ 43