New Skills for New Jobs

Apriori Algoritması

Doç. Dr. Serkan SAVAŞ

Apriori Algoritması: Temel Bilgiler

Apriori Algoritması, birliktelik kuralları ve sık öğe kümeleri madenciliği için etkili bir algoritmadır.

Anahtar Kavramlar:

- Sık Öğeler Kümesi: En az desteğe sahip öğe kümeleri (denoted by L_i for i_{th}-Itemset).
- Apriori Mülkiyet: Yeterli öğe kümesinin herhangi bir alt kümesi sık olmalıdır.
- Birleştirme İşlemi: L_k'yı bulmak için, L_{k-1}'i kendisiyle birleştirerek bir aday k-öğe kümesi oluşturulur.

Itemset Nedir?

Öğelerin bir araya gelmesiyle oluşan kümeye öğe kümesi denir.

Herhangi bir öğe kümesi k- öğeye sahipse buna k-öğe kümesi denir.

Bir öğe kümesi iki veya daha fazla öğeden oluşur.

Sıklıkla ortaya çıkan bir öğe kümesine sık öğe kümesi denir.

Bu nedenle *sık öğe kümesi madenciliği, sıklıkla bir arada bulunan öğeleri belirlemeye yönelik bir veri madenciliği tekniğidir*.

Örneğin, Ekmek ve tereyağı, Dizüstü bilgisayar ve Antivirüs yazılımı vb.

Sık Kullanılan Öğeler Kümesi Nedir?

Bir öğe kümesi, destek ve güven için minimum bir eşik değerini karşılıyorsa sık olarak adlandırılır.

Destek, öğelerin tek bir işlemde birlikte satın alındığı işlemleri gösterir.

Güven, öğelerin birbiri ardına satın alındığı işlemleri gösterir.

Sık öğe kümesi madenciliği yöntemi için, yalnızca minimum eşik desteği ve güven gereksinimlerini karşılayan işlemleri dikkate alıyoruz. Bu madencilik algoritmalarından elde edilen içgörüler birçok fayda, maliyet azaltma ve gelişmiş rekabet avantajı sunar.

Sık Kullanılan Öğeler Kümesi Nedir?

Veri madenciliği için harcanan zaman ve sık madencilik için veri hacmi arasında bir değiş tokuş vardır.

Sık madencilik algoritması, öğe kümelerinin gizli kalıplarını kısa sürede ve daha az bellek tüketimiyle çıkarmak için verimli bir algoritmadır.

Sık Örüntü Madenciliği (FPM)

Sık örüntü madenciliği algoritması, bir veri kümesindeki farklı öğeler arasındaki ilişkileri keşfetmek için veri madenciliğinin en önemli tekniklerinden biridir.

Bu ilişkiler birliktelik kuralları şeklinde temsil edilir. Verilerdeki düzensizlikleri bulmaya yardımcı olur.

FPM, veri analizi, yazılım hataları, çapraz pazarlama, satış kampanyası analizi, pazar sepeti analizi vb. alanlarda birçok uygulamaya sahiptir.

Sık Örüntü Madenciliği (FPM)

Apriori aracılığıyla keşfedilen sık öğe kümelerinin veri madenciliği görevlerinde birçok uygulaması vardır.

Veritabanındaki ilginç örüntüleri bulma, sıralamayı bulma ve birliktelik kurallarının madenciliği gibi görevler bunlardan en önemlileridir.

Birliktelik Kuralları

Birliktelik Kuralı Madenciliği şu şekilde tanımlanır:

- I= { ...} öğeler olarak adlandırılan 'n' ikili öznitelik kümesi olsun.
- D= { ....} veritabanı olarak adlandırılan işlem kümesi olsun.

D'deki her işlem benzersiz bir işlem kimliğine sahiptir ve I'daki öğelerin bir alt kümesini içerir.

Bir kural, X->Y biçiminde bir çıkarım olarak tanımlanır; burada X, Y? I ve X?Y=? X ve Y öğeleri kümesi sırasıyla kuralın öncülü ve sonucu olarak adlandırılır.

Birliktelik Kuralları

Birliktelik kurallarının öğrenilmesi, büyük veri tabanlarında öznitelikler arasındaki ilişkileri bulmak için kullanılır.

Bir birliktelik kuralı, A=> B, "bir dizi işlem için, A öğe kümesinin bazı değerleri, minimum destek ve güvenin karşılandığı koşullar altında B öğe kümesinin değerlerini belirler" şeklinde olacaktır.

Birliktelik Kuralları

Destek ve Güven şu örnekle gösterilebilir:

Ekmek=> tereyağı [destek=%2, güven=%60]

Yukarıdaki ifade bir birliktelik kuralı örneğidir. Bu, ekmek ve tereyağını birlikte satın alan %2'lik bir işlem olduğu ve tereyağının yanı sıra ekmek de satın alan %60'lık bir müşteri kitlesi olduğu anlamına gelir.

Birliktelik Kuralları

A ve B öğe kümeleri için Destek ve Güven formülleri:

$$\text{Support}\ (A) = \frac{\text{Number of transaction in which A appears}}{\text{Total number of transactions}}$$

$$\text{Confidence}\ (A \longrightarrow B) = \frac{\text{Support}(AUB)}{\text{Support}(A)}$$

Birliktelik Kuralları

- Lift değeri:

Lift (kaldırma) değeri, Apriori algoritmasında sıklıkla kullanılan bir metrik olarak karşımıza çıkar. Lift değeri, iki öğenin birlikte görülme olasılığının bağımsız olarak görülme olasılığına oranıdır. Yani, bir öğe setinin lift değeri, bu öğelerin birlikte görülme durumunun rastgele olasılığına göre ne kadar daha yüksek olduğunu gösterir.

$$\text{Lift}(X \to Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X) \times \text{support}(Y)}$$

Burada:

- X ve Y, iki öğe setidir.
- $\text{support}(X \cup Y)$, X ve Y öğe setlerinin birlikte görülme destek (support) değerini temsil eder.
- $\text{support}(X)$ ve $\text{support}(Y)$, sırasıyla X ve Y öğe setlerinin destek değerlerini temsil eder.

Birliktelik Kuralları

- Lift değeri 1'den büyükse, iki öğenin birlikte görülme olasılığı, rastgele bir şekilde görülme olasılığından daha yüksektir.
- Yani, lift değeri büyüdükçe, iki öğenin birlikte görülme eğilimi artar. Eğer lift değeri 1'e eşit veya daha küçükse, bu, iki öğenin birlikte görülme olasılığının rastgele seçilme olasılığından daha düşük olduğunu gösterir.
- Bu durumda, iki öğe arasında bir ilişki olabilir, ancak bu ilişki güçlü değildir.

Birliktelik Kuralları

Birliktelik kuralı madenciliği 2 adımdan oluşur:

1. Tüm sık öğe kümelerini bulun.

2. Yukarıdaki sık öğe kümelerinden birliktelik kuralları oluşturun.

Neden Sık Öğeler Kümesi Madenciliği?

Sık öğe kümesi veya örüntü madenciliği, sık örüntülere, sıralı örüntülere ve diğer birçok veri madenciliği görevine dayanan birliktelik kuralları, korelasyonlar ve grafik örüntü kısıtlamalarının madenciliğindeki geniş uygulamaları nedeniyle yaygın olarak kullanılmaktadır.

Apriori Algoritması - Sık Örüntü Algoritmaları

Apriori algoritması, sık öğe kümesi madenciliği için önerilen ilk algoritmadır.

Daha sonra R Agarwal ve R Srikant tarafından geliştirilmiş ve Apriori olarak bilinmeye başlanmıştır.

Bu algoritma, arama uzayını azaltmak için "birleştirme" ve "budama" olmak üzere iki adım kullanır. En sık rastlanan öğe kümelerini keşfetmek için yinelemeli bir yaklaşımdır.

Apriori der ki...

I öğesinin sık olmama olasılığı şöyledir:

- P(I) < minimum destek eşiği, o zaman I sık değildir.
- P (I+A) < minimum destek eşiği, o zaman I+A sık değildir, burada A da öğe kümesine aittir.
- Bir öğe kümesi minimum destekten daha düşük bir değere sahipse, tüm üst kümeleri de minimum desteğin altına düşecek ve böylece göz ardı edilebilecektir. Bu özellik Antimonotone özelliği olarak adlandırılır.

Apriori Algoritmasında izlenen adımlar...

- Birleştirme Adımı: Bu adım, her bir öğeyi kendisiyle birleştirerek K-öğe kümelerinden (K+1) öğe kümesi oluşturur.
- Budama Adımı: Bu adım, veritabanındaki her bir öğenin sayısını tarar. Aday öğe minimum desteği karşılamıyorsa, seyrek olarak kabul edilir ve bu nedenle kaldırılır. Bu adım, aday öğe kümelerinin boyutunu azaltmak için gerçekleştirilir.

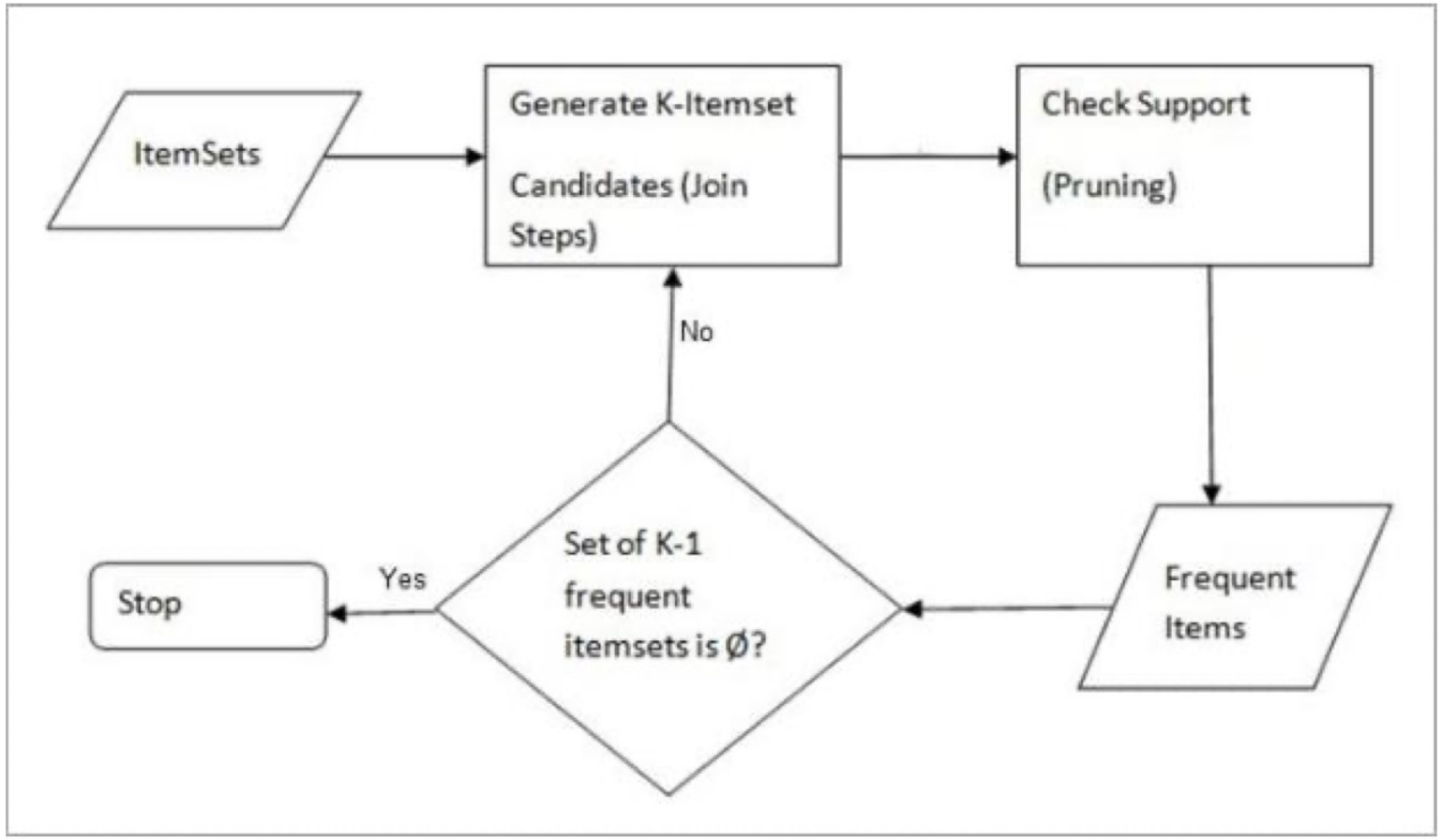
Apriori'de Adımlar

- #1) Algoritmanın ilk iterasyonunda, her bir öğe 1-itemset adayı olarak alınır. Algoritma her bir öğenin oluşumlarını sayacaktır.
- #2) Bazı minimum destek, min_sup (örneğin 2) olsun. Oluşumu min sup'u karşılayan 1 - öğe kümeleri kümesi belirlenir. Sadece min_sup'a eşit veya daha fazla olan adaylar bir sonraki iterasyon için öne alınır ve diğerleri budanır.
- #3) Daha sonra, min_sup ile 2'li sık öğeler kümesi keşfedilir. Bunun için birleştirme adımında öğeler kendisiyle birleştirilerek 2'li bir grup oluşturularak 2'li öğe kümesi oluşturulur.

Apriori'de Adımlar

- #4) 2 öğeli adaylar min-sup eşik değeri kullanılarak budanır. Artık tabloda yalnızca min-sup değerine sahip 2'li öğe kümesi olacaktır.
- #5) Bir sonraki iterasyon, birleştirme ve budama adımını kullanarak 3 öğeli kümeler oluşturacaktır. Bu yineleme, 3 öğeli kümelerin alt kümelerinin, yani her grubun 2 öğeli alt kümelerinin min_sup'a düştüğü antimonotone özelliğini izleyecektir. Tüm 2 öğeli alt kümeler sıksa, üst küme sık olacaktır, aksi takdirde budanacaktır.
- #6) Sonraki adım, 3 öğe kümesini kendisiyle birleştirerek ve alt kümesi min_sup kriterlerini karşılamıyorsa budayarak 4 öğe kümesi oluşturmayı takip edecektir. En sık öğe kümesine ulaşıldığında algoritma durdurulur.

Support threshold=50%, Confidence= 60%



Support threshold=50% => 0.5*6= 3 => min_sup=3

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

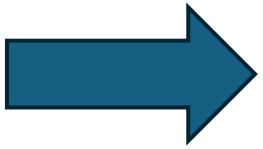
Budama Aşaması:
I5 öğesi min_sup=3'ü karşılamıyor, bu nedenle budanacak. Yalnızca I1, I2, I3, I4'ün min_sup sayısını karşıladığını görebiliriz.

Item	Count
I1	4
I2	5
I3	4
I4	4
I5	2

Item	Count
I1	4
I2	5
I3	4
I4	4

Birleştirme Adımı: 2 öğe kümesi oluşturma. İşlem tablosundan 2-itemset'in oluşumları...

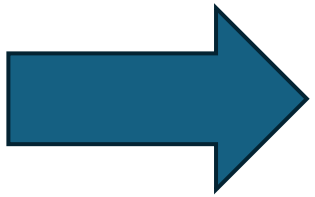
Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4



Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2

Budama Aşaması:
{I1, I4} and {I3, I4} öğeleri min_sup=3'ü karşılamıyor, bu nedenle budanacak.

Item	Count
I1,I2	4
I1,I3	3
I1,I4	2
I2,I3	4
I2,I4	3
I3,I4	2



Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

Birleştirme ve Budama Adımı: 3 öğe kümesi oluşturma.

İşlem tablosundan 3 öğeli kümenin oluşumlarını bulun.

Son tablodan da min_sup'ı destekleyen 2 öğeli alt kümeleri bulun.

Öğe kümesi {I1, I2, I3} alt kümeleri için son tabloda {I1, I2}, {I1, I3}, {I2, I3}'ün oluştuğunu görebiliriz, bu nedenle {I1, I2, I3} sıktır. {I1, I2, I4} öğe kümesi için {I1, I2}, {I1, I4}, {I2, I4}, kümelerine bakılmalıdır. {I1, I4} alt kümesinin sık olmadığını görebiliriz, çünkü tabloda yer almamaktadır, bu nedenle {I1, I2, I4} sık değildir, dolayısıyla silinmiştir.

Yalnızca {I1, I2, I3} sıktır.

Item
I1,I2,I3
I1,I2,I4
I1,I3,I4
I2,I3,I4

Transaction	List of items
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Item	Count
I1,I2	4
I1,I3	3
I2,I3	4
I2,I4	3

Bu tablolara göre kurallar:

{I1, I2} => {I3} :

- Confidence = support {I1, I2, I3} / support {I1, I2} = (3/ 4)* 100 = 75%

{I1, I3} => {I2}:

- Confidence = support {I1, I2, I3} / support {I1, I3} = (3/ 3)* 100 = 100%

{I2, I3} => {I1}:

- Confidence = support {I1, I2, I3} / support {I2, I3} = (3/ 4)* 100 = 75%

{I1} => {I2, I3}:

- Confidence = support {I1, I2, I3} / support {I1} = (3/ 4)* 100 = 75%

{I2} => {I1, I3}:

- Confidence = support {I1, I2, I3} / support {I2 = (3/ 5)* 100 = 60%

{I3} => {I1, I2}:

- Confidence = support {I1, I2, I3} / support {I3} = (3/ 4)* 100 = 75%

Bu, minimum güven eşiğinin %60 olması durumunda yukarıdaki tüm birliktelik kurallarının güçlü olduğunu göstermektedir.

Avantaj ve Dezavantajları

+++

Anlaşılması kolay algoritma

Join ve Prune adımlarının büyük veritabanlarındaki büyük öğe kümelerine uygulanması kolaydır

---

Öğe kümeleri çok büyükse ve minimum destek çok düşük tutulursa yüksek hesaplama gerektirir.

Tüm veritabanının taranması gerekir.

Apriori Verimliliğini Artırma Yöntemleri

Hash Tabanlı Teknik: Bu yöntem, k-öğe kümelerini ve bunlara karşılık gelen sayıyı oluşturmak için hash tablosu adı verilen hash tabanlı bir yapı kullanır. Tabloyu oluşturmak için bir hash fonksiyonu kullanır.

İşlem Azaltma: Bu yöntem iterasyonlarda taranan işlem sayısını azaltır. Sık öğe içermeyen işlemler işaretlenir veya kaldırılır.

Apriori Verimliliğini Artırma Yöntemleri

Bölümleme: Bu yöntem, sık öğe kümelerini çıkarmak için yalnızca iki veritabanı taraması gerektirir. Herhangi bir öğe kümesinin veritabanında potansiyel olarak sık olması için, veritabanının bölümlerinden en az birinde sık olması gerektiğini söyler.

Örnekleme: Bu yöntem D veritabanından rastgele bir S örneği seçer ve ardından S'de sık öğe kümesi arar. Bu durum min_sup değeri düşürülerek azaltılabilir.

Dinamik Öğe Kümesi Sayımı: Bu teknik, herhangi bir anda yeni aday öğe kümeleri ekleyebilir.

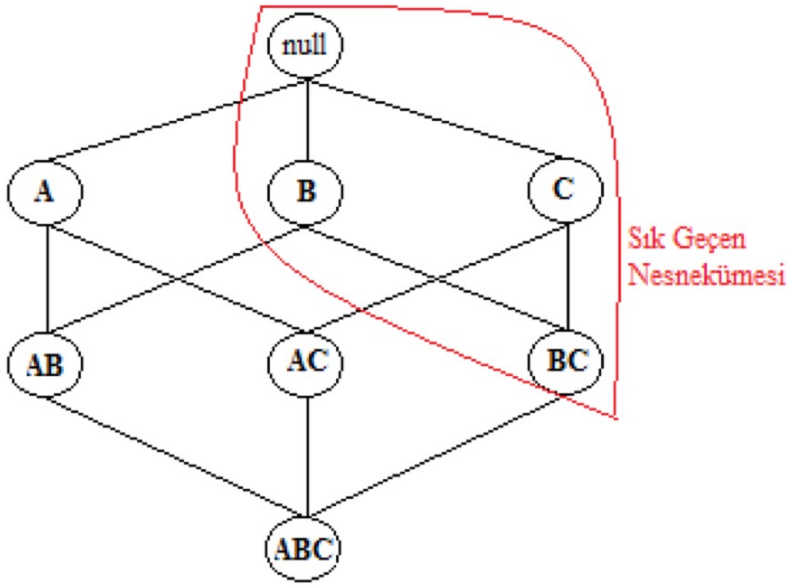
Apriori

Birliktelik kural analizinin temelini oluşturan, algoritmalar içerisinde en fazla bilinen ve kullanılan algoritmadır ve adı da önceki anlamındaki ‘prior’ kelimesinden gelmektedir.

Apriori algoritmasının temel özelliği; eğer bir nesne küme sık geçen ise bu kümenin bütün alt kümeleri de sık geçen nesne küme olmalıdır. Aynı mantıkla nesne küme sık kullanılmayan ise bu kümenin bütün alt kümeleri de sık kullanılmayan nesne küme olarak değerlendirilir.

Apriori

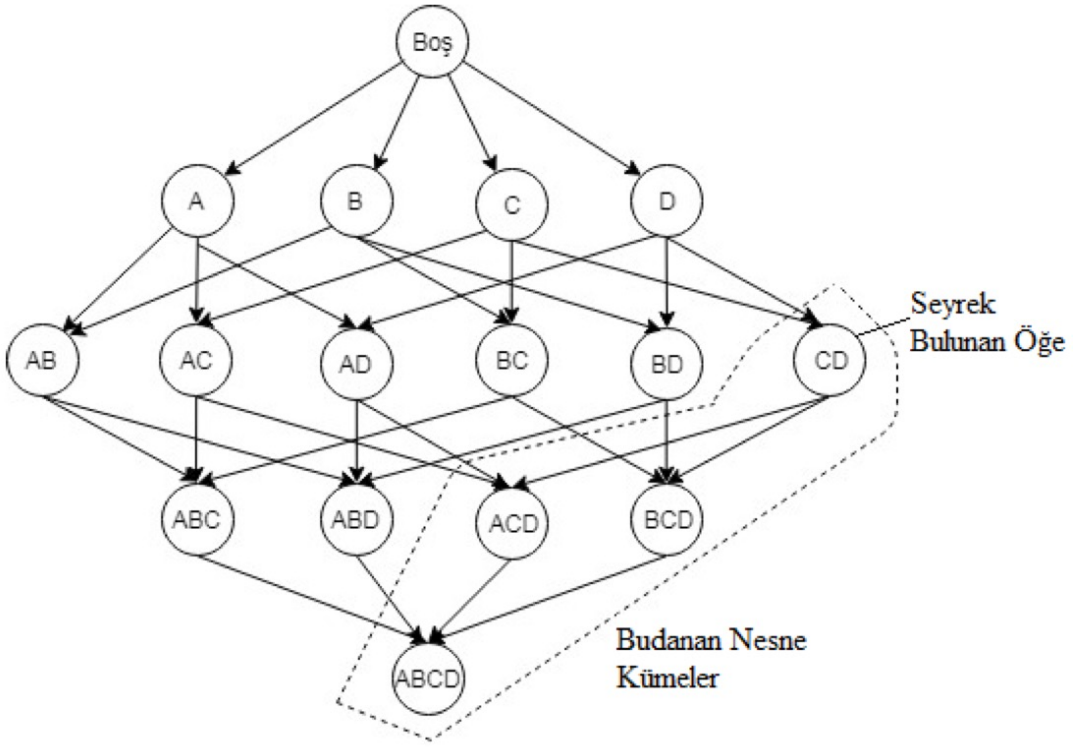
Şekilde gösterildiği gibi kırmızı yuvarlak içerisine alınmış alanın sık kullanılan nesne kümelerin olduğu düşünülürse {BC} kümesinin bir sık geçen nesneküme olduğu varsayılır. {BC} kümelerini içeren her işlemin bu kümenin alt kümeleri olan {B}, {C} kümelerini de içerir sonucuna varılır. {BC} sık geçen nesneküme ise bütün alt kümeleri de sık geçen nesne küme olmalıdır.



Apriori

Sık olmayan bir nesne kümenin bütün alt kümeleri de sık olmayan nesnekümelerdir.

Şekilde gösterilen {CD} kümesinin sık olmayan küme olarak bulunduğu tespit edilmiş, {CD} kümesini kapsayan {ACD}, {BCD}, {ABCD} üst kümeleri de sık olmayan küme kabul edileceği için bu kümeler budanabilir.



Apriori

Apriori algoritması başlangıçta veri tabanını tarar ve öğelerinin veri tabanında kaç kez geçtiklerine dair bilgiyi içeren destek değerlerini bulur. Başlangıçta girilen minimum destek değerinden daha küçük destek değerine sahip öğeler nesne kümelere dâhil edilmez, bu öğeler ayıklanarak veri tabanı sadeleştirilir.

Nesne kümelerin tekli destek değerlerinden yararlanarak ikili nesne kümeler oluşturur, ikili nesne kümelerin destek değerini hesaplayarak düşük destek değere sahip öğeleri ayıklayarak sadeleştirir.

Aynı işlemi ikili nesne kümelerden yararlanarak üçlü nesne kümeler oluşturarak devam eder. Bu tarama ve sadeleştirme işlemi, girilen minimum destek değerden küçük nesne kümesi kalmayana kadar devam eder.

Apriori

```
Apriori(T, MST)   // T veri tabanı, MST minimum destek değeri
{
        Ck: k büyüklükte aday nesne küme;
        Lk: k büyüklükte sık kullanılan nesne küme;
        L1: sık kullanılan ürünler;
        for (k=1; Lk != Boş küme; k++)
        {
                Ck+1= Lk tarafından oluşturulan adaylar;
                for each(veri tabanında işlem olduğu sürece)
                        Ck+1 inci ürünün destek değerini bir artır;
                Lk+1 = Ck+1 deki destek değeri büyük olanları;
        }
} end
Return Uk Lk;
```

Örnek

TID	Ürünler
1	Ekmek, Meyve Suyu, Soda
2	Su, Meyve Suyu, Çikolata
3	Ekmek, Su, Meyve Suyu, Çikolata
4	Su, Çikolata

İlk olarak kullanıcı tarafından destek değeri girilir. Bu örneklem için destek eşik değeri 2 olarak kabul edilmiştir. Tabloda bir D veri tabanına ait işlemler verilmiştir. Bu veri tabanında 4 işlem bulunmaktadır,
|D|=4 olarak belirlenir.
İlk olarak her bir nesne küme C1 aday kümesini oluşturmak için taranır ve elemanlarının destek değerleri hesaplanır.

Öğe Listesi	Destek Değeri
Ekmek	2
Su	3
Meyve Suyu	3
Soda	1
Çikolata	3

Tablodaki değerlerine bakıldığında 4 ürünün destek değeri minimum destek değerinin üzerinde olduğu için sık kullanılan nesne kümeler olarak kabul edilir. Soda sık geçen nesne kümesinden çıkarılır. Bir öğeli nesne kümeler ile sık geçen L1 kümesi oluşturulur.

Örnek

- L1 aday kümesi:

Öğe Listesi	Destek Değeri
Ekmek	2
Su	3
Meyve Suyu	3
Çikolata	3

İki öğeli sık geçen nesnekümeler (C2) oluşturulur. C2 aday kümesi öğelerin ikili kombinasyonu ile oluşur.

Öğe Listesi	Destek Değeri
Ekmek, Su	1
Ekmek, Meyve Suyu	2
Ekmek, Çikolata	1
Su, Meyve Suyu	2
Su, Çikolata	3
Meyve Suyu, Çikolata	2

D veri tabanı bir kez daha taranır ve C2 kümesindeki öğelerin destek değerleri hesaplanır.
C2 kümesindeki minimum destek değerine sahip olan 2 öğeli nesne kümeleri, L2 aday kümesini oluşturur.

Örnek

- L2 aday kümesi:

Öğe Listesi	Destek Değeri
Ekmek, Meyve Suyu	2
Su, Meyve Suyu	2
Su, Çikolata	3
Meyve Suyu, Çikolata	2

Üç öğeli sık geçen nesne kümeleri, kalan öğelerin üçlü kombinasyonu ile oluşturulur.

Su, Meyve Suyu, Çikolata → 2

Destek değeri 2'nin altında ürün olmadığı için L3 ve C3 aynı çıkacaktır.
3. adıma geçerken Apriori algoritmasının budama özelliği kullanılmamıştır.
Son olarak, oluşturulan 3 elemanlı {Su, Meyve Suyu, Çikolata} sık geçen nesne kümesinden %40 minimum destek kriterini sağlayan kural oluşturulmuştur.

Genelleştirilmiş Kural Türetme (Generalized Rule Induction - GRI)

GRI algoritması, verilerden kurallar türeterek çalışan kural tabanlı bir sınıflandırma algoritmasıdır.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) ve CN2 (Classification based on 2-of-n) gibi geleneksel kural tümevarım algoritmalarının bir genellemesidir ve daha esnek kural gösterimlerine izin verir.

GRI, bir veri kümesindeki örnekleri doğru bir şekilde sınıflandıran bir dizi kural oluşturmayı amaçlamaktadır.

Apriori'den farklı olarak GRI, birliktelik kuralı madenciliğinden ziyade sınıflandırmaya odaklanır.

Kural Üretimi

- Kural üretimi aşamasında, algoritma farklı özniteliklerin ve değerlerinin kombinasyonlarını düşünerek olası kural uzayını keşfeder.
- Bu keşif çeşitli stratejilerle yönlendirilebilir:
  - **En İyi-İlk Arama**: İlk olarak en umut verici öznitelik-değer kombinasyonlarını seçer ve aramayı ilerletir.
  - **Işın Araması**: Belirli bir boyutta aday kural kümesini korur ve devam etmek için en umut verici olanları seçer.
  - **Genetik Algoritmalar**: Evrimsel teknikleri kullanarak kural oluşturur ve ardışık nesiller boyunca kural oluşturur ve geliştirir.
- Strateji seçimi, veri kümesinin boyutu, kural uzayının karmaşıklığı ve hesaplama kaynakları gibi faktörlere bağlıdır.

Kural Değerlendirme

- Aday kurallar, kalitelerini ve kullanışlılıklarını değerlendirmek için birden fazla kriter temel alınarak değerlendirilir:
  - **Kapsam**: Kuralın veri kümesinde kapsadığı örneklerin oranı. Yüksek kapsama, kuralın veri kümesinin büyük bir bölümüne uygulandığını gösterir.
  - **Doğruluk**: Kural tarafından kapsanan örneklerin doğru bir şekilde sınıflandırılan oranı. Yüksek doğruluk, kuralın kapsadığı örneklerin çoğu için doğru tahminler yapılmasını sağlar.
  - **Basitlik**: Kuralın karmaşıklığı, genellikle içeren koşulların veya özniteliklerin sayısıyla ölçülür. Basit kurallar daha kolay yorumlanabilir ve genellenebilir.
  - **Genellik**: Kuralın eğitim veri kümesinin ötesindeki örnekler üzerinde ne kadar geçerli olduğu. Yüksek genellik oranı, kuralların yeni verilere de genelleme yapma olasılığının daha yüksek olduğunu gösterir.
- Değerlendirme metrikleri, alan bilgisi veya kullanıcı tercihlerine göre farklı şekillerde ağırlıklandırılabilir.

Kural Budama ve Kural Seçimi

- Değerlendirmeden sonra, aday kurallar tekrarlanarak gereksiz veya düşük kaliteli kurallar ortadan kaldırılır:
  - **Tekrarlılık Kaldırma**: Kural setinde örtüşen veya diğer kurallar tarafından emilen kuralları tanımlar ve kaldırır.
  - **Doğruluk Eşiği**: Düşük doğruluğa sahip veya belirli bir eşiğin altında kapsama alanına sahip kurallar, kural setinin kalitesini korumak için budanabilir.
  - **Basitlik Kısıtlamaları**: Karmaşık kurallar, genellikle daha basit ve yorumlanabilir kurallara öncelik vermek için tercih edilir.
- Budama, nihai kural setinin özlü, yorumlanabilir ve sınıflandırma için etkili olduğundan emin olmak için yapılır.

Kural Uygulaması

- Son kural seti elde edildikten sonra, yeni örneklerin sınıflandırılması veya tahmin edilmesi için uygulanabilir:
  - Sınıflandırma sırasında, her örnek, kurallara sırayla uygulanır ve örneğin koşullarını karşılayan ilk kurala atanır.
  - Kurallar, sırayla, güven seviyelerine veya kullanıcı tarafından belirlenen diğer kriterlere göre uygulanabilir.
- Sınıflandırma süreci şeffaf ve yorumlanabilir olduğundan, her bir örneğin nasıl karar verildiğine dair içgörüler sunar.

İteratif İyileştirme

- Bazı uygulamalarda, GRI algoritması, kural setini daha da optimize etmek için tekrarlayan rafine adımlar içerebilir:
  - İlk kural setini uyguladıktan sonra, sınıflandırma sonuçlarından gelen geri bildirimler, kuralları güncellemek veya iyileştirmek için kullanılabilir.
  - Rafine stratejiler, kuralların performansına dayalı olarak, doğrulama verileri veya diğer kriterlerle uyumlu olarak koşulları ayarlamak, yeni kurallar eklemek veya gereksiz kuralları kaldırmak gibi adımları içerebilir.
- İteratif iyileştirme, kuralların doğruluğunu ve genelleme yeteneğini artırarak, birkaç iterasyon boyunca kural setinin kalitesini artırır.

Örnek Uygulamalar

Kaynaklar

- University of Delphi: https://www.cvs.edu.in/upload/Apriori%20Algorithm.pdf
- Tokyürek, E:

BİRLİKTELİK KURAL ÇIKARIM ALGORİTMALARI KULLANILARAK MARKET SEPET ANALİZİ – Yüksek Lisans Tezi.