VERİ MADENCİLİĞİ

Farklı Sınıflandırma Yöntemleri

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

Konular

- Sınıflandırma yöntemleri
  - Örnek tabanlı yöntemler
    - k-En Yakın Komşu Yöntemi
  - Genetik Algoritmalar
  - Karar Destek Makinaları
  - Bulanık Küme Sınıflandırıcılar
  - Öngörü
    - Eğri Uydurma
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Örnek Tabanlı Yöntemler

- Örnek tabanlı sınıflandırma:
  - Öğrenme kümesi saklanır
  - Sınıflandırılacak yeni bir örnek geldiğinde öğrenme kümesi sınıf etiketini öngörmek için kullanılır (tembel (lazy) yöntemler)
- Yöntemler
  - k-en yakın komşu yöntemi

Örnek Tabanlı Yöntemler

Öğrenme Kümesi

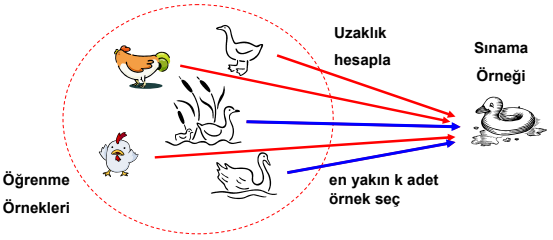
Nit1	.....	NitN	Sınıf
			A
			B
			B
			C
			A
			C
			B

Yeni örnek

Nit1	.....	NitN

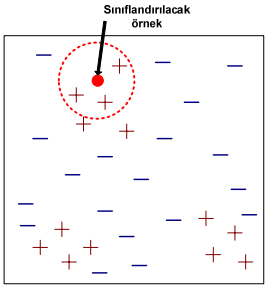
En Yakın Komşu Yöntemi

- Temel yaklaşım: Sınıflandırılmak istenen örneğe en yakın örnekleri bul.
  Örnek: ördek gibi yürüyor, ördek gibi bağırıyor
  => büyük olasılıkla ördek

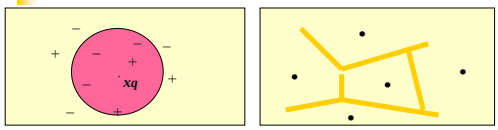


En Yakın Komşu Sınıflandırıcı

- Bütün örnekler n-boyutlu uzayda bir noktaya karşı düşürülür
- Nesneler arasındaki uzaklık (Öklid uzaklığı)
- Öğrenilen fonksiyon ayrık değerli veya gerçel değerli olabilir
- Ayrık değerli fonksiyonlarda k-komşu algoritması Xq örneğine en yakın k öğrenme örneğinde en çok görülen sınıf değerini verir
- Sürekli değerli fonksiyonlarda en yakın k öğrenme örneğinin ortalaması alınır



K-En Yakın Komşu Yöntemi



- *Xq* örneği 1-en yakın komşuya göre pozitif olarak, 5-en yakın komşuya göre negatif olarak sınıflandırılır
- Voronoi diyagramları: Her öğrenme örneğini çevreleyen dışbükey çokgenlerden oluşan karar yüzeyi

K-En Yakın Komşu Yöntemi

- Uzaklık-ağırlıklı k-en yakın komşu algoritması
  - Öğrenme kümesindeki örneklere (x_i), sınıflandırılmak istenen örneğe (x_q) olan uzaklıklarına göre ağırlıklar verilmesi
  - yakın örneklerin ağırlığı daha fazla $w \equiv \frac{1}{d(x_q, x_i)^2}$
- k-en yakın komşunun ortalaması alındığı için gürültülü veriden az etkileniyor
- İlgisiz nitelikler uzaklığı etkileyebilir
  - bu nitelikler uzaklık hesaplarken kullanılmayabilir

Konular

- Sınıflandırma yöntemleri
  - Örnek tabanlı yöntemler
    - k-En Yakın Komşu Yöntemi
  - Genetik Algoritmalar
  - Karar Destek Makinaları
  - Bulanık Küme Sınıflandırıcılar
  - Öngörü
    - Eğri Uydurma
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Genetik Algoritmalar

- Optimizasyon amaçlı
- Bir başlangıç çözümü öneriyor, tekrarlanan her ara adımda daha iyi çözüm üretmeye çalışıyor.
- Doğal evrime ve en iyi olanın yaşamını sürdürmesine dayanıyor
- Çözümü birey olarak sunuyor.
- Birey: $I=I_1,I_2,\ldots,I_n$ – I_j kullanılan alfabenin bir karakteri
- gen: I_j
- Toplum: Bireylerden oluşan küme

Genetik Algoritmalar

- Genetik Algoritmalar (GA) 5 parçadan oluşuyor:
  - Bireylerden oluşan bir başlangıç kümesi, P
  - Çaprazlama (Crossover): Bir anne babadan yeni bireyler üretmek için yapılan işlem
  - Mutasyon: Bir bireyi rastgele değiştirme
  - Uygunluk (fitness): En iyi bireyleri belirleme
  - Çaprazlama ve mutasyon tekniklerini uygulayan ve uygunluk fonksiyonuna göre toplum içindeki en iyi bireyleri seçen algoritma

Çaprazlama Örnekleri

Parents	Children
000 | 000 111 | 111	000 | 111 111 | 000

a) Single Crossover

Parents	Children
000 | 000 | 00 111 | 111 | 11	000 | 111 | 00 111 | 000 | 11

a) Multiple Crossover

Genetik Algoritma

```
Input:
   P      //Initial Population
Output:
   P'     //Improved Population
Genetic Algorithm:
                //Illustrates Genetic Algorithm
   repeat
      N = | P |;
      P' = ∅;
      repeat
         i1, i2 = select(P);
         o1, o2 = cross(i1, i2);
         o1 = mutate(o1);
         o2 = mutate(o2);
         P' = P' ∪ {o1, o2};
      until | P' |= N;
      P = P';
   until termination criteria satisfied;
```

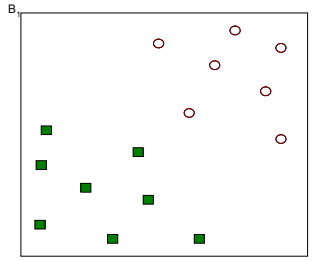

GA – Avantajlar, Dezavantajlar

- Avantaj
  - Paralel çalışabilir
  - NP karmaşık problem çözümlerine uygun
- Dezavantaj
  - Son kullanıcının modeli anlaması güç
  - Problemi GA ile çözmeye uygun hale getirmek zor
  - Uygunluk fonksiyonunu belirlemek zor
  - Çaprazlama ve mutasyon tekniklerini belirlemek zor

Konular

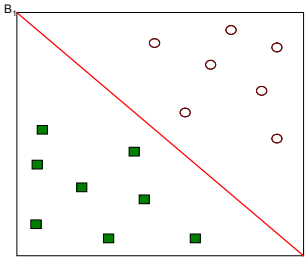
- Sınıflandırma yöntemleri
  - Örnek tabanlı yöntemler
    - k-En Yakın Komşu Yöntemi
  - Genetik Algoritmalar
  - Karar Destek Makinaları
  - Bulanık Küme Sınıflandırıcılar
  - Öngörü
    - Eğri Uydurma
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Karar Destek Makineleri



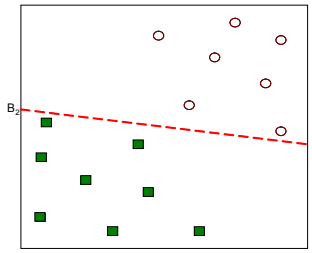
- Karar Destek Makineleri (Support Vector Machines SVM): Veriyi ayıracak doğrusal bir sınır

Karar Destek Makineleri



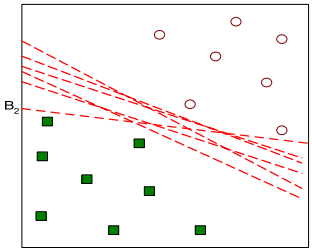
- Bir çözüm

Karar Destek Makineleri



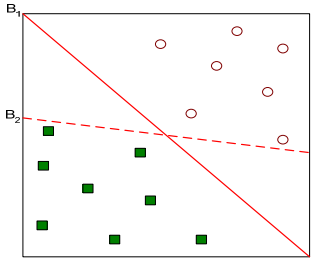
- Başka bir çözüm

Karar Destek Makineleri



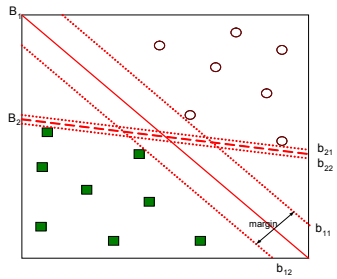
- Diğer çözümler

Karar Destek Makineleri



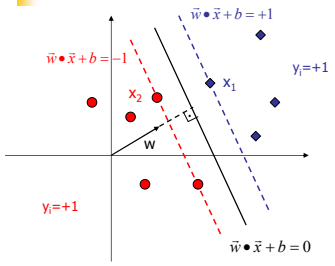
- Hangisi daha iyi? B1 mi, B2 mi?
- Daha iyi nasıl tanımlanır?

Karar Destek Makineleri



- Farklı sınıftan örnekler arasındaki uzaklığı enbüyüten düzlemi bul => B1, B2'den daha iyi

Karar Destek Makineleri



$(w\cdot x) + b \geq +1,\ y_i=+1$
$(w\cdot x) + b \leq -1,\ y_i=-1$
$\Rightarrow y_i(w\cdot x + b) \geq +1$

$(w\cdot x_1)+b=y_i=+1$
$(w\cdot x_2)+b=y_i=-1$

$\Rightarrow w\cdot(x_1-x_2)=2$
$\Rightarrow \frac{w}{||w||}\cdot(x_1-x_2)=\frac{2}{||w||}$

Karar Destek Makineleri

- $\frac{2}{||w||^2}$ enbüyük olması isteniyor
  - $y_i(w\cdot x + b) \geq +1$ olacak şekilde $\frac{||w||^2}{2}$ enküçük olmalı
  - kısıtlı eniyileme (constraint optimization) problem

$$Lp = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N}\alpha_i\left[y_i(w\cdot x_i + b) - 1\right]$$

  - Problem

$\alpha_1 \ldots \alpha_N$ bulunması
$\Sigma\alpha_i$ - ½$\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ en büyük olacak
kısıtlar:
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0,\ \forall\alpha_i$

Eniyileme Problemi Çözümü

- Çözüm

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w}^T\mathbf{x_k}\ \forall\ \mathbf{x_k},\ \alpha_k \neq 0$$

- Sınıflandırma fonksiyonu

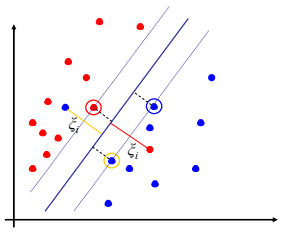
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$$

  - f(x) = 1 ise x pozitif olarak, diğer durumlarda negatif olarak sınıflandırılıyor.

Karar Destek Makineleri

- Öğrenme kümesi doğrusal olarak ayrılamıyor
  - ξ_i değişkenleri ekleniyor

$(w \cdot x) + b \geq +1 - \xi_i$, $y_i = +1$
$(w \cdot x) + b \leq -1 + \xi_i$, $y_i = -1$
$\Rightarrow y_i(w \cdot x + b) \geq +1 - \xi_i$
$\xi_i \geq 0, \forall i$



$$Lp = \frac{1}{2}\|w\|^2 - C\left(\sum_{i=1}^{N} \xi_i\right)^k$$

- Problem:
  $\alpha_1 \ldots \alpha_N$ bulunması
  $\Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ en büyük olacak
  kısıtlar:
  (1) $\Sigma\alpha_i y_i = 0$
  (2) $0 \leq \alpha_i \leq C, \forall\alpha_i$

Eniyileme Problemi Çözümü

- Çözüm

$$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$$
$$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x_k}, \text{ k} = \text{argmax } \alpha_k$$

- Sınıflandırma fonksiyonu

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$$

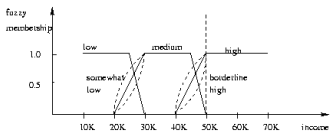
SVM Uygulamaları

- Boser, Guyon ve Vapnik tarafından 1992 yılında önerildi. 1990'ların sonlarına doğru yaygın olarak kullanılmaya başlandı
- SVM için en yaygın eniyileme algoritmaları SMO [Platt '99] ve SVM^{light} [Joachims' 99]

Konular

- Sınıflandırma yöntemleri
  - Örnek tabanlı yöntemler
    - k-En Yakın Komşu Yöntemi
  - Genetik Algoritmalar
  - Karar Destek Makinaları
  - Bulanık Küme Sınıflandırıcılar
  - Öngörü
    - Eğri Uydurma
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Bulanık Küme Sınıflandırıcılar



- Bulanık mantık 0.0 ve 1.0 arasında gerçel değerler kullanarak üyelik dereceleri hesaplar
- Nitelik değerleri bulanık değerlere dönüştürülür
- Kurallar kümesi oluşturulur
- Yeni bir örneği sınıflandırmak için birden fazla kural kullanılır
- Her kuraldan gelen sonuç toplanır

Konular

- Sınıflandırma yöntemleri
  - Örnek tabanlı yöntemler
    - k-En Yakın Komşu Yöntemi
  - Genetik Algoritmalar
  - Karar Destek Makinaları
  - Bulanık Küme Sınıflandırıcılar
  - Öngörü
    - Eğri Uydurma
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Öngörü

- Sınıflandırma problemleriyle aynı yaklaşım
  - model oluştur
  - bilinmeyen değeri hesaplamak için modeli kullan
    - eğri uydurma
      - doğrusal
      - doğrusal olmayan
- Sınıflandırma ayrık değerli
- Öngörü sürekli değerli

Eğri Uydurma

- Doğrusal eğri uydurma:
  - en basit eğri uydurma yöntemi
  - veri doğrusal bir eğri ile modellenir.
    - veri kümesindeki niteliklerin doğrusal fonksiyonu

$$y = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

  - öğrenme kümesindeki y_i sınıfından bir x_i örneği için çıkış

$$y = w_0 x_{i0} + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_k x_{ik} = \sum_{j=0}^{k} w_j x_{ij}$$

  - karesel hatayı enküçültecek ağırlıkları bulma

$$\sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{k} w_j x_{ij} \right)^2$$

Konular

- Sınıflandırma yöntemleri
- Model Değerlendirme
  - Hata oranı
  - Anma
  - Duyarlılık
  - F-ölçütü
  - ROC eğrileri
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme

Sınıflandırma Modelini Değerlendirme

- Model başarımını değerlendirme ölçütleri nelerdir?
  - Hata oranı
  - Anma
  - Duyarlılık
  - F-ölçütü
- Farklı modellerin başarımı nasıl karşılaştırılır?
  - ROC

Sınıflandırma Hatası

- Sınıflandırma yöntemlerinin hatalarını ölçme
  - başarı: örnek doğru sınıfa atandı
  - hata: örnek yanlış sınıfa atandı
  - hata oranı: hata sayısının toplam örnek sayısına bölünmesi
- Hata oranı sınama kümesi kullanılarak hesaplanır

Model Başarımını Değerlendirme

- Model başarımını değerlendirme ölçütleri
  - modelin ne kadar doğru sınıflandırma yaptığını ölçer
    - hız, ölçeklenebilirlik gibi özellikleri değerlendirmez
- Karışıklık matrisi:

		ÖNGÖRÜLEN SINIF	
		Sınıf=1	Sınıf=-1
DOĞRU SINIF	Sınıf =1	a	b
	Sınıf =-1	c	d

a: TP (true positive)
b: FN (false negative)
c: FP (false positive)
d: TN (true negative)

Model Başarımını Değerlendirme: Doğruluk

	ÖNGÖRÜLEN SINIF	
DOĞRU SINIF	+1	-1
+1	a (TP)	b (FN)
-1	c (FP)	d (TN)

- Modelin başarımı:

$$Dogruluk = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Hata\ \ Orani = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{TP+TN+FP+FN}$$

Örnek

Sınıflandırıcı A	
TP=25	FN=25
FP=25	TN=25

Doğruluk=%50

Sınıflandırıcı B	
TP=50	FN=0
FP=25	TN=25

Doğruluk=%75

Sınıflandırıcı C	
TP=25	FN=25
FP=0	TN=50

Doğruluk=%75

- Hangi sınıflandırıcı daha iyi?
  - B ve C, A'dan daha iyi bir sınıflandırıcı
  - B, C'den daha iyi bir sınıflandırıcı mı?

Model Başarımını Değerlendirme: Duyarlılık

	ÖNGÖRÜLEN SINIF	
DOĞRU SINIF	+1	-1
+1	a (TP)	b (FN)
-1	c (FP)	d (TN)

$$\text{Duyarlılık} = \frac{\text{Doğru sınıflandırılmış pozitif örnek sayısı}}{\text{Pozitif sınıflandırılmış örneklerin sayısı}} = \frac{TP}{TP+FP}$$

Model Başarımını Değerlendirme: Anma

	ÖNGÖRÜLEN SINIF	
DOĞRU SINIF	+1	-1
+1	a (TP)	b (FN)
-1	c (FP)	d (TN)

$$\underset{\text{Doğru pozitif oranı}}{\text{Anma}} = \frac{\text{Doğru sınıflandırılmış pozitif örnek sayısı}}{\text{Pozitif örneklerin sayısı}} = \frac{TP}{TP+FN}$$

Anma / Duyarlılık

- A modeli B modelinden daha iyi anma ve duyarlılık değerine sahipse A modeli daha iyi bir sınıflandırıcıdır.
- Duyarlılık ve anma arasında ters orantı var.

Sınıflandırıcıları Karşılaştırma

- Doğruluk en basit ölçüt
- Duyarlılık ve anma daha iyi ölçme sağlıyor
  - Model A'nın duyarlılığı model B'den daha iyi ancak model B'nin anma değeri model A'dan daha iyi olabilir.

Model Başarımını Değerlendirme: F-ölçütü

- F-ölçütü: Anma ve duyarlılığın harmonik ortalamasını alır.

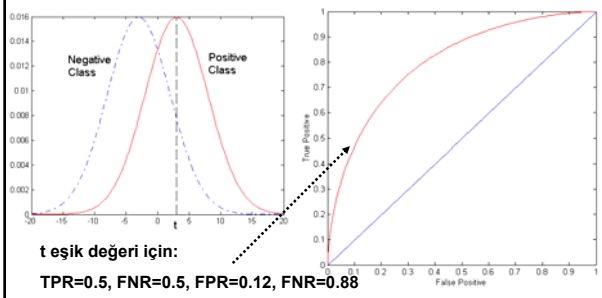
$$\text{F-ölçütü} = \frac{2 * \text{duyarlılık} * \text{anma}}{\text{duyarlılık} + \text{anma}}$$

ROC (Receiver Operating Characteristic)

- İşaret işlemede bir sezicinin, gürültülü bir kanalda doğru algılama oranının yanlış alarm oranına karşı çizdirilen grafiği (algılayıcı işletim eğrisi)
- Farklı sınıflandırıcıları karşılaştırmak için ROC eğrileri
- Doğru pozitif (TPR - y ekseni) oranının yanlış pozitif (FPR – x ekseni) oranına karşı çizdirilen grafiği
  - TPR = TP / (TP + FN)
  - FPR = FP / (TN + FP)
- ROC üzerindeki her nokta bir sınıflandırıcının oluşturduğu bir modele karşı düşer

ROC Eğrisi

- İki sınıftan oluşan tek boyutlu bir veri kümesi (positive – negative)
- x > t için her örnek pozitif olarak sınıflandırılıyor



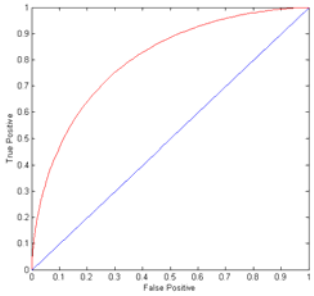
t eşik değeri için:

TPR=0.5, FNR=0.5, FPR=0.12, FNR=0.88

ROC Eğrisi

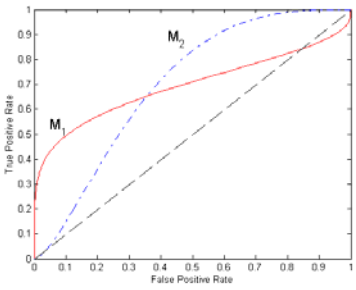
(FPR,TPR)

- (0,0): Bütün örneklerin negatif sınıflandırılması
- (1,1): Bütün örneklerin pozitif sınıflandırılması
- (0,1): ideal durum
- Çapraz çizgi:
  - Rastlantısal tahmin



ROC Eğrilerinin Kullanılması

- Farklı modelleri karşılaştırmak için
- M_1 veya M_2 birbirlerine üstünlük sağlamıyor
  - küçük FPR değerleri için M_1 daha iyi
  - büyük FPR değerleri için M_2 daha iyi
- ROC eğrisi altında kalan alan
  - ideal = 1
  - Rastlantısal tahmin=0.5



ROC Eğrisinin Çizilmesi

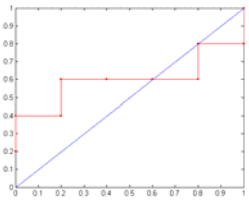
- Her örnek için P(+|A) olasılığı hesaplanır
- P(+|A) değeri azalarak sıralanır
- Her farklı P(+|A) değeri için bir eşik değeri uygulanır
- Her eşik değeri için TP, FP, TN, FN hesaplanır

Örnek	P(+|A)	Sınıf
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

ROC Eğrisinin Çizilmesi

Class	+	-	+	-	-	-	+	-	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

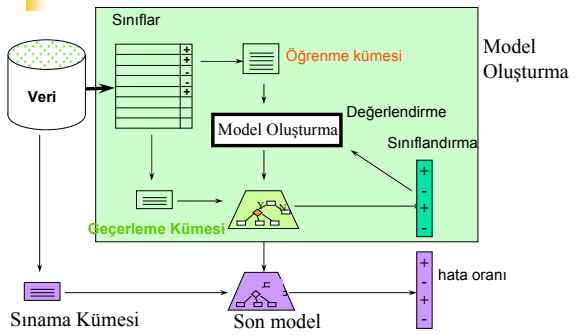
ROC Eğrisi:



Model Parametrelerini Belirleme

- Sınama kümesi sınıflandırıcı oluşturmak için kullanılmaz
- Bazı sınıflandırıcılar modeli iki aşamada oluşturur
  - modeli oluştur
  - parametreleri ayarla
- Sınama kümesi parametreleri ayarlamak için kullanılmaz
- Uygun yöntem üç veri kümesi kullanma: öğrenme, geçerleme, sınama
  - geçerleme kümesi parametre ayarlamaları için kullanılır
  - model oluşturulduktan sonra öğrenme ve geçerleme kümesi son modeli oluşturmak için kullanılabilir

Sınıflandırma: Öğrenme, Geçerleme, Sınama



Model Başarımını Tahmin Etme

- Örnek: Doğruluğu %25 olan bir modelin gerçek başarımı ne kadardır?
  - Sınama kümesinin büyüklüğüne bağlı
- Sınıflandırma (hileli) yazı tura atmaya benziyor
  - tura doğru sınıflandırma (başarı), yazı yanlış sınıflandırma (başarısızlık)
- İstatistikte birbirinden bağımsız olayların başarı ya da başarısızlıkla sonuçlanmaları Bernoulli dağılımı ile modellenir.
- Gerçek başarı oranını belirlemek için istatistikte güven aralıkları tanımlanmıştır.

Güven Aralığı

- p belli bir güvenle belli bir aralıkta bulunmaktadır.
- Örnek: N=1000 olayda S=750 başarı sağlanmış.
  - Tahmin edilen başarı oranı: 75%
  - Gerçek başarıya ne kadar yakın
    - %80 güven ile $p \in [73{,}2 - 76{,}7]$
- Örnek: N=100 olayda S=75 başarı sağlanmış.
  - Tahmin edilen başarı oranı: 75%
  - Gerçek başarıya ne kadar yakın
    - %80 güven ile $p \in [69{,}1 - 80{,}1]$

Ortalama Değer ve Varyans

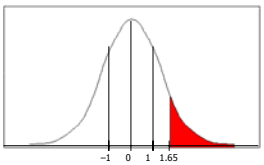
- Başarı oranı p olan tek bir Bernoulli olayının ortalama değeri ve varyansı: p, $p\,(1-p)$
- N kere tekrarlanan Bernoulli olayının beklenen başarı oranı $f=S/N$
- Büyük N değerleri için, f normal dağılım
- f için ortalama değer ve varyans: p, $p\,(1-p)/N$
- Ortalama değeri 0 ve varyansı 1 olan X rastlantı değişkeninin $\%c$ güven aralığı :

  $\Pr[-z \le X \le z]=c$
- Simetrik bir dağılım için:

  $\Pr[-z \le X \le z]=1 - 2*\Pr[X \ge z]$

Güven Sınırları

- Ortalama değeri 0 ve varyansı 1 olan bir normal dağılımın güven sınırları



Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- Pr[$-1{,}65 \le X \le 1{,}65$]=90%
- *f*'in ortalama değerinin 0, varyansının 1 olacak şekilde dönüştürülmesi gerekir.

Dönüşüm

- *f*'in ortalama değerinin 0, varyansının 1 olacak şekilde dönüştürülmesi için $\frac{f-p}{\sqrt{p(1-p)/N}}$
- Güven aralığı $\Pr\left[-z \le \frac{f-p}{\sqrt{p(1-p)/N}} \le z\right] = c$
- p'nin değeri $p = \left(f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}\right) \Big/ \left(1 + \frac{z^2}{N}\right)$

Örnek

- *f* = 75%, *N* = 1000, *c* = 80% (*z* = 1.28):
  p ∈ [0,732 – 0,767]
- *f* = 75%, *N* = 100, *c* = 80% (*z* = 1.28):
  p ∈ [0,691 – 0,801]

Konular

- Sınıflandırma yöntemleri
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
  - holdout
  - k-kat çapraz geçerleme
  - Bootstrap
- Sınıflandırıcıları birleştirme

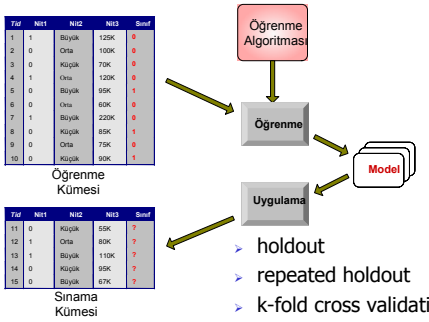
Verinin Dengesiz Dağılımı

- Küçük veya dengesiz veri kümeleri için örnekler tanımlayıcı olmayabilir
- Veri içinde bazı sınıflardan çok az örnek olabilir
  - tıbbi veriler: %90 sağlıklı, %10 hastalık
  - elektronik ticaret: %99 alışveriş yapmamış, %1 alışveriş yapmış
  - güvenlik: %99 sahtekarlık yapmamış, %1 sahtekarlık yapmış
- Örnek: Sınıf1: 9990 örnek, Sınıf2: 10 örnek
  - bütün örnekleri sınıf1'e atayan bir sınıflandırıcının hata oranı: 9990 / 10000= %99,9
    - hata oranı yanıltıcı bir ölçüt olabilir

Dengeli Dağılım Nasıl Sağlanır?

- Veri kümesinde iki sınıf varsa
  - iki sınıfın eşit dağıldığı bir veri kümesi oluştur
  - Az örneği olan sınıftan istenen sayıda rasgele örnekler seç
  - Çok örneği olan sınıftan aynı sayıda örnekleri ekle
- Veri kümesinde iki sınıftan fazla sınıf varsa
  - Öğrenme ve sınama kümesini farklı sınıflardan aynı sayıda örnek olacak şekilde oluştur

Örnek



- holdout
- repeated holdout
- k-fold cross validation
- bootstrapping

Büyük Veri Kümelerinde Değerlendirme

- Veri dağılımı dengeli ise: Veri kümesindeki örnek sayısı ve her sınıfa ait örnek sayısı fazla ise basit bir değerlendirme yeterli
  - *holdout* yöntemi: Belli sayıda örnek sınama için ayrılır, geriye kalan örnekler öğrenme için kullanılır
    - genelde veri kümesinin 2/3'ü öğrenme, 1/3'i sınama kümesi olarak ayrılır
  - öğrenme kümesi kullanılarak model oluşturulur ve sınama kümesi kullanılarak model değerlendirilir

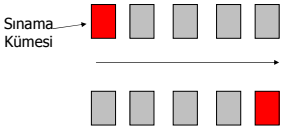
Tekrarlı Holdout Yöntemini

- Veri kümesini farklı altkümelere bölerek holdout yöntemini tekrarlama
  - Her eğitme işleminde veri kümesinin belli bir bölümü öğrenme kümesi olarak rasgele ayrılır
  - Modelin hata oranı, işlemler sonunda elde edilen modellerin hata oranlarının ortalaması
- Problem: Farklı eğitme işlemlerindeki sınama kümeleri örtüşebilir

k-Kat Çapraz Geçerleme

- Veri kümesi eşit boyutta *k* adet farklı gruba ayrılır.

- Bir grup sınama, diğerleri öğrenme için ayrılır.



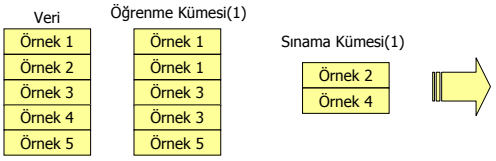
- Her grup bir kere sınama kümesi olacak şekilde deneyler *k* kere tekrarlanır.

Biri Hariç Çapraz Geçerleme

- k-kat çapraz geçerlemenin özel hali
  - k sayısı veri kümesindeki örnek sayısına (N) eşit
- Model *N-1* örnek üzerinde eğitilir, dışarıda bırakılan *1* örnek üzerinde sınanır
- Bu işlem her örnek *1* kez sınama için kullanılacak şekilde tekrarlanır
  - model *N* kez eğitilir
- Model başarımı denemelerin başarımının ortalaması
- Verinin en etkin şekilde kullanımı

Bootstrap Yöntemi

- Veri kümesinden yerine koyma yöntemi ile örnekler seçilerek öğrenme kümesi oluşturulur
  - *N* örnekten oluşan veri kümesinden yerine koyarak *N* örnek seçilir
  - Bu küme öğrenme kümesi olarak kullanılır
  - Öğrenme kümesinde yer almayan örnekler sınama kümesi olarak kullanılır



0.632 bootstrap

- N örnekten oluşan bir veri kümesinde bir örneğin seçilmeme olasılığı: $1-\frac{1}{N}$
- Sınama kümesinde yer alma olasılığı:

$$\left(1-\frac{1}{n}\right)^n \approx e^{-1} = 0.368$$

- Öğrenme kümesi veri kümesindeki örneklerin %63,2'sinden oluşuyor

Bootstrap Yönteminde Model Hatasını Belirleme

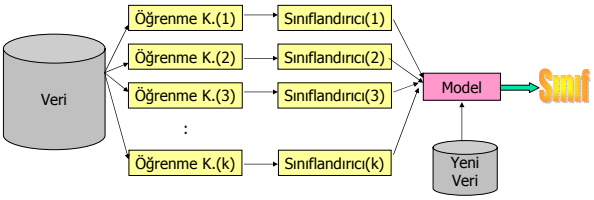
- Model başarımını sadece sınama kümesi kullanarak belirleme kötümser bir yaklaşım
  - model örneklerin sadece ~%63'lük bölümüyle eğitiliyor
- Model başarımı hem öğrenme kümesindeki hem de sınama kümesindeki başarım ile değerlendirilir

  hata = 0,632 $\text{hata}_{(\text{sınama})}$ + 0,368 $\text{hata}_{(\text{öğrenme})}$
- İşlem birkaç kez tekrarlanarak hatanın ortalaması alınır.

Konular

- Sınıflandırma yöntemleri
- Model Değerlendirme
- Öğrenme, sınama, geçerleme kümelerini oluşturma
- Sınıflandırıcıları birleştirme
  - Bagging
  - Boosting

Model Başarımını Artırma



- Bir grup sınıflandırıcı kullanma
  - Bagging
  - Boosting

Bagging

- *N* örnekten oluşan bir veri kümesinde bootstrap yöntemi ile *T* örnek seç
- Bu işlemi *k* öğrenme kümesi oluşturmak üzere tekrarla
- Aynı sınıflandırma algoritmasını *k* öğrenme kümesi üzerinde kullanarak *k* adet sınıflandırıcı oluştur
- Yeni bir örneği sınıflandırmak için her sınıflandırıcının sonucunu öğren
- Yeni örnek en çok hangi sınıfa atanmışsa o sınıfın etiketiyle etiketlendir.

Boosting

- Öğrenme kümesindeki her örneğin bir ağırlığı var
- Her öğrenme işleminden sonra, her sınıflandırıcı için yapılan sınıflandırma hatasına bağlı olarak örneklerin ağırlığı güncelleniyor
- Yeni bir örneği sınıflandırmak için her sınıflandırıcının doğruluğuna bağlı olarak ağırlıklı ortalaması alınıyor.

VERİ MADENCİLİĞİ
Demetleme Yöntemleri

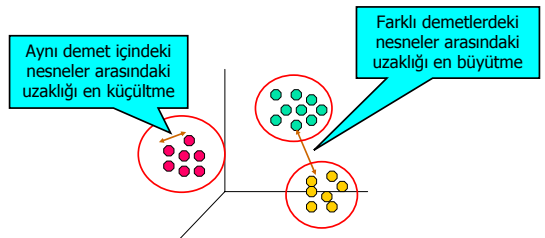
Yrd. Doç. Dr. Şule Gündüz Öğüdücü
http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

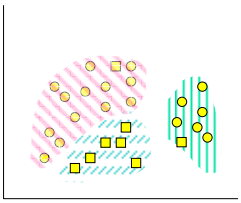
Demetleme

- Nesneleri demetlere (gruplara) ayırma
- Demet: birbirine benzeyen nesnelerden oluşan grup
  - Aynı demetteki nesneler birbirine daha çok benzer
  - Farklı demetlerdeki nesneler birbirine daha az benzer



Demetleme

- Gözetimsiz öğrenme: Hangi nesnenin hangi sınıfa ait olduğu ve sınıf sayısı belli değil
- Uygulamaları:
  - verinin dağılımını anlama
  - başka veri madenciliği uygulamaları için ön hazırlık



Demetleme Uygulamaları

- Örüntü tanıma
- Görüntü işleme
- Ekonomi
- Aykırılıkları belirleme
- WWW
  - Doküman demetleme
  - Kullanıcı davranışlarını demetleme
  - Kullanıcıları demetleme
- Diğer veri madenciliği algoritmaları için bir önişleme adımı
  - Veri azaltma – demet içindeki nesnelerin temsil edilmesi için demet merkezlerinin kullanılması

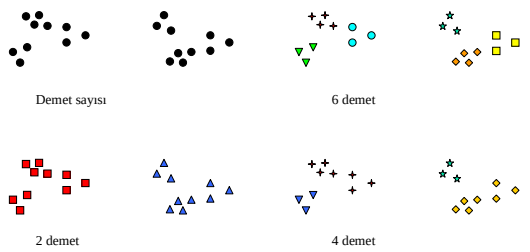
Veri Madenciliğinde Demetleme

- Ölçeklenebilirlik
- Farklı tipteki niteliklerden oluşan nesneleri demetleme
- Farklı şekillerdeki demetleri oluşturabilme
- En az sayıda giriş parametresi gereksinimi
- Hatalı veriler ve aykırılıklardan en az etkilenme
- Model oluşturma sırasında örneklerin sırasından etkilenmeme
- Çok boyutlu veriler üzerinde çalışma
- Kullanıcıların kısıtlarını göz önünde bulundurma
- Sonucun yorumlanabilir ve anlaşılabilir olması

İyi Demetleme

- İyi demetleme yöntemiyle elde edilen demetlerin özellikleri
  - aynı demet içindeki nesneler arası benzerlik fazla
  - farklı demetlerde bulunan nesneler arası benzerlik az
- Oluşan demetlerin kalitesi seçilen benzerlik ölçütüne ve bu ölçütün gerçeklenmesine bağlı
  - Uzaklık / Benzerlik nesnelerin nitelik tipine göre değişir
    - Nesneler arası benzerlik: s(i,j)
    - Nesneler arası uzaklık: d(i,j) = 1 – s(i,j)
- İyi bir demetleme yöntemi veri içinde gizlenmiş örüntüleri bulabilmeli
- Veriyi gruplama için uygun demetleme kriteri bulunmalı
  - demetleme = aynı demetteki nesneler arası benzerliği enbüyüten, farklı demetlerdeki nesneler arası benzerliği enküçülten fonksiyon
- Demetleme sonucunun kalitesi seçilen demetlerin şekline ve temsil edilme yöntemine bağlı

Farklı Demetler



Temel Demetleme Yaklaşımları

- Bölünmeli yöntemler: Veriyi bölerek, her grubu belirlenmiş bir kritere göre değerlendirir
- Hiyerarşik yöntemler: Veri kümelerini (ya da nesneleri) önceden belirlenmiş bir kritere göre hiyerarşik olarak ayırır
- Yoğunluk tabanlı yöntemler: Nesnelerin yoğunluğuna göre demetleri oluşturur
- Model tabanlı yöntemler: Her demetin bir modele uyduğu varsayılır. Amaç bu modellere uyan verileri gruplamak

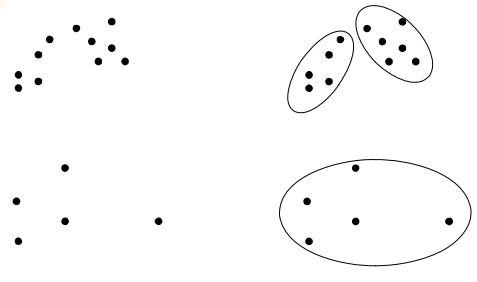
Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
    - K-means demetleme yöntemi
    - K-medoids demetleme yöntemi
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

Bölünmeli Yöntemler

- Amaç: n nesneden oluşan bir veri kümesini (D) k ($k \leq n$) demete ayırmak
  - her demette en az bir nesne bulunmalı
  - her nesne sadece bir demette bulunmalı
- Yöntem: Demetleme kriterini enbüyütücek şekilde D veri kümesi k gruba ayırma
  - Global çözüm: Mümkün olan tüm gruplamaları yaparak en iyisini seçme (NP karmaşık)
  - Sezgisel çözüm: k-means ve k-medoids
    - k-means (MacQueen'67): Her demet kendi merkezi ile temsil edilir
    - k-medoids veya PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Her demet, demette bulunan bir nesne ile temsil edilir

Bölünmeli Demetleme

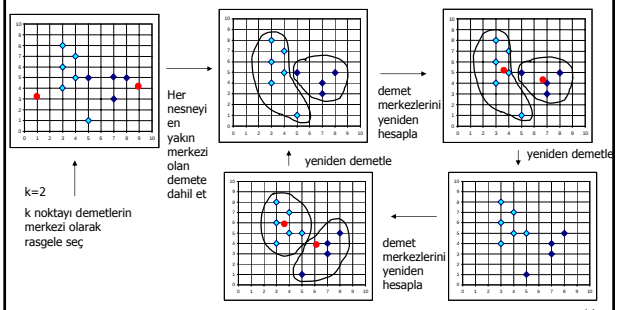


K-means Demetleme

- Bilinen bir *k* değeri için k-means demetleme algoritmasının 4 aşaması vardır:
  1. Veri kümesi *k* altkümeye ayrılır (her demet bir altküme)
  2. Her demetin ortalaması hesaplanır: merkez nokta (demetteki nesnelerin niteliklerinin ortalaması)
  3. Her nesne en yakın merkez noktanın olduğu demete dahil edilir
  4. Nesnelerin demetlenmesinde değişiklik olmayana kadar adım 2'ye geri dönülür.

K-means Demetleme Yöntemi

- Örnek

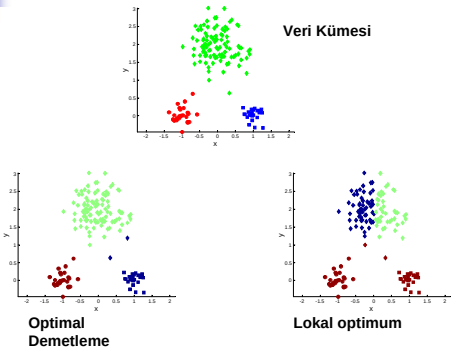


k-means Demetleme Yöntemi

- Demet sayısının belirlenmesi gerekir
- Başlangıçta demet merkezleri rasgele belirlenir
  - Her uygulamada farklı demetler oluşabilir
- Benzerlik Öklid uzaklığı, kosinüs benzerliği gibi yöntemlerle ölçülebilir
- Az sayıda tekrarda demetler oluşur
  - Yakınsama koşulu çoğunlukla az sayıda nesnenin demet değiştirmesi şekline dönüştürülür
- Karmaşıklığı:
  - Yer karmaşıklığı - *O((n+k) d)*
  - Zaman karmaşıklığı - *O(ktnd)*

  k: demet sayısı, *t*: tekrar sayısı, *n*: nesne sayısı, *d*: nitelik sayısı

K-Means: İki Farklı Demetleme



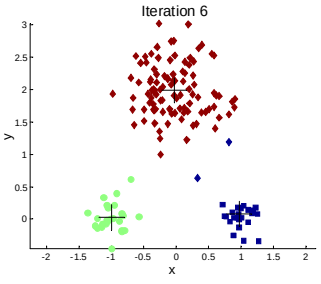
K-Means Demetleme Yöntemini Değerlendirme

- Yaygın olarak kullanılan yöntem hataların karelerinin toplamı (Sum of Squared Error SSE)
  - Nesnelerin bulundukları demetin merkez noktalarına olan uzaklıklarının karelerinin toplamı

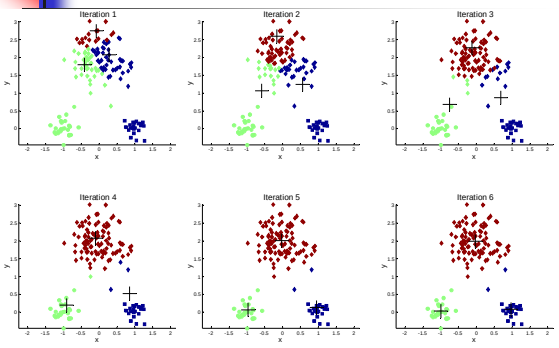
$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$$

  x: C_i demetinde bulunan bir nesne, m_i : C_i demetinin merkez noktası
- Hataların karelerinin toplamını azaltmak için *k* demet sayısı artırılabilir
  - Küçük k ile iyi bir demetleme, büyük k ile kötü bir demetlemeden daha az SSE değerine sahip olabilir.
- Başlangıç için farklı merkez noktaları seçerek farklı demetlemeler oluşturulur
- En az SSE değerini sahip olan demetleme seçilir

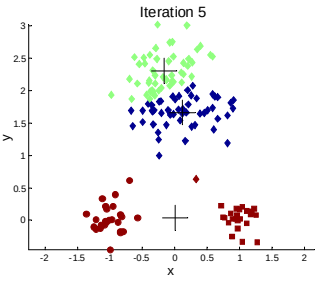
Merkez Noktaların Seçimi



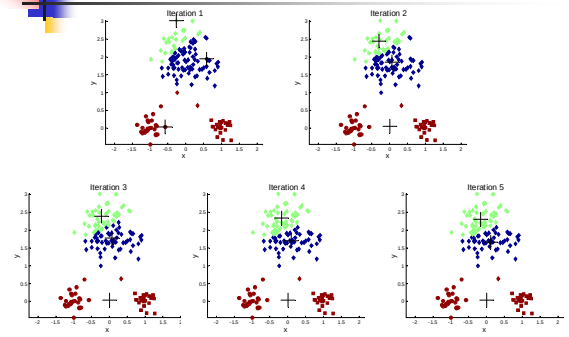
Merkez Noktaların Seçimi



Merkez Noktaların Seçimi



Merkez Noktaların Seçimi



K-Means Demetleme Çeşitleri

- K-Means demetlemeye başlamadan önce yapılanlar
  - Veri kümesini örnekleyerek hiyerarşik demetleme yap. Oluşan *k* demetin ortalamasını başlangıç için merkez nokta seç
  - Başlangıçta *K*'dan fazla merkez nokta seç. Daha sonra bunlar arasından *k* tane seç.
- K-Means demetleme işlemi sonrasında yapılanlar
  - Küçük demetleri en yakın başka demetlere dahil et
  - En büyük toplam karesel hataya sahip olan demeti böl
  - Merkez noktaları birbirine en yakın demetleri birleştir
  - Toplam karesel hatada en az artışa neden olacak iki demeti birleştir

K-Means Demetleme Algoritmasının Özellikleri

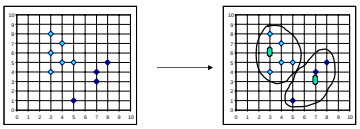
- Gerçeklemesi kolay
- Karmaşıklığı diğer demetleme yöntemlerine göre az
- K-Means algoritması bazı durumlarda iyi sonuç vermeyebilir
  - Veri grupları farklı boyutlarda ise
  - Veri gruplarının yoğunlukları farklı ise
  - Veri gruplarının şekli küresel değilse
  - Veri içinde aykırılıklar varsa

Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
    - K-means demetleme yöntemi
    - K-medoids demetleme yöntemi
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

K-Medoids Demetleme Yöntemi

- Her demeti temsil etmek için demet içinde orta nokta olan nesne seçilir.
  - 1, 3, 5, 7, 9 ortalama: 5
  - 1, 3, 5, 7, 1009 ortalama 205
  - 1, 3, 5, 7, 1009 orta nokta 5



K-Medoids Demetleme Yöntemi

- PAM (Partitioning Araound Medoids 1987)
  - Başlangıçta *k* adet nesne demetleri temsil etmek üzere rasgele seçilir x_{ik}
  - Kalan nesneler en yakın merkez nesnenin bulunduğu demete dahil edilir
  - Merkez nesne olmayan rasgele bir nesne seçilir x_{rk}
  - x_{rk} merkez nesne olursa toplam karesel hatanın ne kadar değiştiğini bulunur

$$TC_{ik} = \sum_{j=1}^{n_k}(x_{ik} - x_{jk})^2 - \sum_{j=1}^{n_k}(x_{rk} - x_{jk})^2$$

n_k: *k* demeti içindeki nesne sayısı
x_{jk}: *k* demeti içindeki *j.* nesne

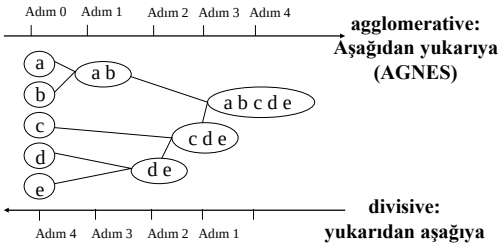
  - TC_{ik}<0 ise O_{rk} merkez nesne olarak atanır.
  - Demetlerde değişiklik oluşmayana kadar 3. adıma geri gidilir.
- Küçük veri kümeleri için iyi sonuç verebilir, ancak büyük veri kümeleri için uygun değil
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994)

Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

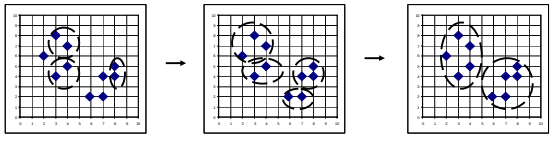
Hiyerarşik Demetleme

- Demet sayısının belirlenmesine gerek yok
  - Sonlanma kriteri belirlenmesi gerekiyor



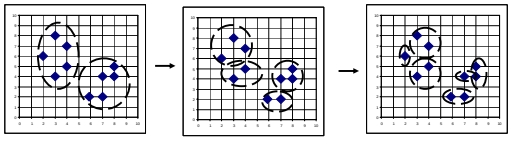
Hiyerarşik Yöntemler

- *AGNES* (*AG*glomerative *NES*ting):
  - Kaufmann ve Rousseeuw tarafından 1990 yılında önerilmiştir.
  - Birinci adımda her nesne bir demet oluşturur.
  - Aralarında en az uzaklık bulunan demetler her adımda birleştirilir.
  - Bütün nesneler tek bir demet içinde kalana kadar ya da istenen sayıda demet elde edene kadar birleştirme işlemi devam eder.



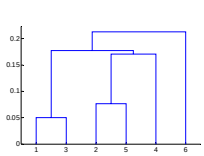
Hiyerarşik Yöntemler

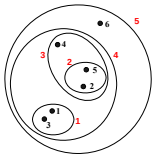
- *DIANA* (*DI*visive *ANA*lysis):
  - Kaufmann ve Rousseeuw tarafından 1990 yılında önerilmiştir.
  - AGNES'in yaptığı işlemlerin tersini yapar.
  - En sonunda her nesne bir demet oluşturur.
  - Her nesne ayrı bir demet oluşturana ya da istenilen demet sayısı elde edene kadar ayrılma işlemi devam eder.



Hiyerarşik Demetleme

- Dendogram: Demetler hiyerarşik olarak ağaç yapısı şeklinde görüntülenebilir
- Ara düğümler çocuk düğümlerdeki demetlerin birleşmesiyle elde edilir
  - Kök: bütün nesnelerden oluşan tek demet
  - Yapraklar: bir nesneden oluşan demetler
- Dendogram istenen seviyede kesilerek demetler elde edilir

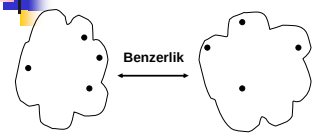


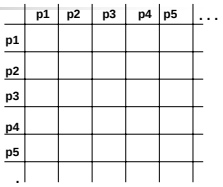


Aşağıdan Yukarıya Demetleme

- Algoritma
  1. Uzaklık matrisini hesapla
  2. Her nesne bir demet
  3. Tekrarla
  4. En yakın iki demeti birleştir
  5. Uzaklık matrisini yeniden hesapla
  6. Sonlanma: Tek bir demet kalana kadar
- Uzaklık matrisini hesaplarken farklı yöntemler farklı demetleme sonuçlarına neden olurlar

Demetler Arası Uzaklık

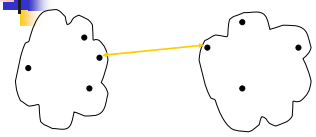


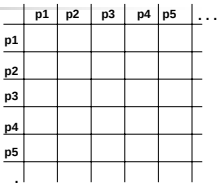


Uzaklık Matrisi

- MIN (Tek Bağ)
- MAX (Tam Bağ)
- Ortalama
- Merkezler arası uzaklık

Demetler Arası Uzaklık

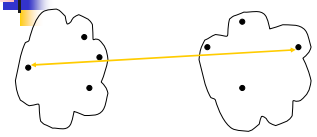


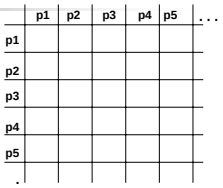


Uzaklık Matrisi

- MIN (Tek Bağ)
- MAX (Tam Bağ)
- Ortalama
- Merkezler arası uzaklık

Demetler Arası Uzaklık

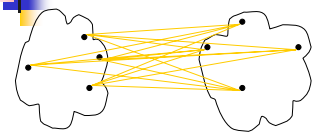


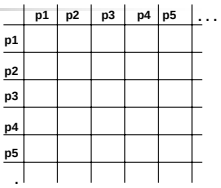


Uzaklık Matrisi

- MIN (Tek Bağ)
- MAX (Tam Bağ)
- Ortalama
- Merkezler arası uzaklık

Demetler Arası Uzaklık

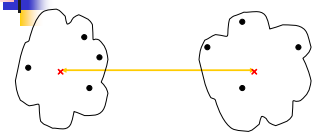


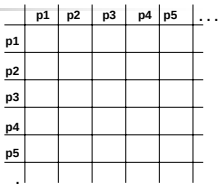


Uzaklık Matrisi

- MIN (Tek Bağ)
- MAX (Tam Bağ)
- Ortalama
- Merkezler arası uzaklık

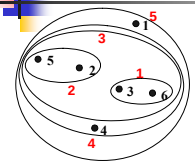
Demetler Arası Uzaklık

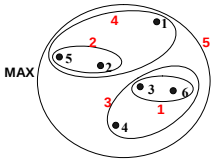


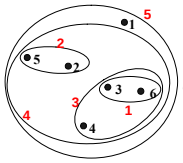


- MIN (Tek Bağ)
- MAX (Tam Bağ)
- Ortalama
- Merkezler arası uzaklık

Farklı Uzaklık Yöntemlerinin Etkisi







Hiyerarşik Demetleme Yöntemlerinin Özellikleri

- Demetleme kriteri yok
- Demet sayılarının belirlenmesine gerek yok
- Aykırılıklardan ve hatalı verilerden etkilenir
- Farklı boyuttaki demetleri oluşturmak problemli olabilir
- Yer karmaşıklığı – $O(n^2)$
- Zaman karmaşıklığı – $O(n^2 \log n)$

  n : nesne sayısı

Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

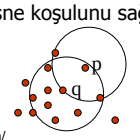
Yoğunluk Tabanlı Yöntemler

- Demetleme nesnelerin yoğunluğuna göre yapılır.
- Başlıca özellikleri:
  - Rasgele şekillerde demetler üretilebilir.
  - Aykırı nesnelerden etkilenmez.
  - Algoritmanın son bulması için yoğunluk parametresinin verilmesi gerekir.
- Başlıca yoğunluk tabanlı yöntemler:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

DBSCAN

- İki parametre:
  - *Eps*: En büyük komşuluk yarıçapı
  - *MinPts*: Eps yarıçaplı komşuluk bölgesinde bulunan en az nesne sayısı
- $N_{eps}(p)$: {q∈D | d(p,q)≤Eps}
- Doğrudan erişilebilir nesne: *Eps* ve *MinPts* koşulları altında bir *q* nesnesinin doğrudan erişilebilir bir *p* nesnesi şu şartları sağlar:
  - $p \in N_{eps}(q)$
  - *q* nesnesinin çekirdek nesne koşulunu sağlaması

$N_{eps}(q) \geq MinPts$

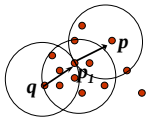


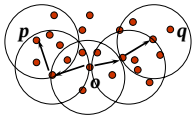
MinPts = 5

Eps = 1 cm

DBSCAN

- Erişilebilir nesne:
  - *Eps* ve *MinPts* koşulları altında *q* nesnesinin erişilebilir bir *p* nesnesi olması için:
    - $p_1,p_2,...,p_n$ nesne zinciri olması,
    - $p1=q$, $pn=p$,
    - p_i nesnesinin doğrudan erişilebilir nesnesi:p_{i+1}
- Yoğunluk bağlantılı Nesne:
  - *Eps* ve *MinPts* koşulları altında *q* nesnesinin yoğunluk bağlantılı nesnesi *p* şu koşulları sağlar:
    - *p* ve *q* nesneleri *Eps* ve *MinPts* koşulları altında bir *o* nesnesinin erişilebilir nesnesidir.





Yoğunluk Tabanlı Yöntemler: DBSCAN

- Veri tabanındaki her nesnenin *Eps* yarıçaplı komşuluk bölgesi araştırılır.
- Bu bölgede *MinPts*'den daha fazla nesne bulunan *p* nesnesi çekirdek nesne olacak şekilde demetler oluşturulur.
- Çekirdek nesnelerin doğrudan erişilebilir nesneleri bulunur.
- Yoğunluk bağlantılı demetler birleştirilir.
- Hiçbir yeni nesne bir demete eklenmezse işlem sona erer.
- Yer karmaşıklığı – *O(n)*
- Zaman karmaşıklığı – *O(nlogn)*
  n: nesne sayısı

Konular

- Demetleme işlemleri
  - Demetleme tanımı
  - Demetleme uygulamaları
- Demetleme Yöntemleri
  - Bölünmeli Yöntemler
  - Hiyerarşik Yöntemler
  - Yoğunluk Tabanlı Yöntemler
  - Model Tabanlı Yöntemler

Model Tabanlı Demetleme Yöntemleri

- Veri kümesi için öngörülen matematiksel model en uygun hale getiriliyor.
- Verinin genel olarak belli olasılık dağılımlarının karışımından geldiği kabul edilir.
- Model tabanlı demetleme yöntemi
  - Modelin yapısının belirlenmesi
  - Modelin parametrelerinin belirlenmesi
- Örnek *EM (Expectation Maximization)* Algoritması

Model Tabanlı Demetleme Yöntemleri

- İstatistiksel yaklaşım:
  - K nesneden oluşan bir veri kümesi $D=\{x_1,x_2,...,x_K\}$
  - her $\mathbf{x}_i$ $(i\in[1,...K])$ nesnesi $\boldsymbol{\Theta}$ parametre kümesiyle tanımlanan bir olasılık dağılımından oluşturulur.
  - Olasılık dağılımının, $c_j\in\mathbf{C}=\{c_1, c_2, ...,c_G\}$ şeklinde G adet bileşeni vardır.
  - Her $\boldsymbol{\Theta}_g$, $g\in[1, ...,G]$ parametre kümesi g bileşeninin olasılık dağılımını belirleyen, $\boldsymbol{\Theta}$ kümesinin ayrışık bir alt kümesidir.
  - Herhangi bir $\mathbf{x}_i$ nesnesi öncelikle, $p(c_g|\boldsymbol{\Theta}) = \tau_g$ ($\Sigma_G \tau_g=1$ olacak şekilde) bileşen katsayısına (ya da bileşenin seçilme olasılığına) göre bir bileşene atanır.
  - Bu bileşen $p(\mathbf{x}_i|c_g;\boldsymbol{\Theta}_g)$ olasılık dağılımına göre $\mathbf{x}_i$ değişkenini oluşturur.
  - Böylece bir $\mathbf{x}_i$ nesnesinin bu model için olasılığı bütün bileşenlerin olasılıklarının toplamıyla ifade edilebilir:

$$p(x_i \mid \boldsymbol{\Theta}) = \sum_{g=1}^{G} p(c_g \mid \boldsymbol{\Theta}) p(\mathbf{x}_i \mid c_g; \boldsymbol{\Theta}_g)$$

$$p(x_i \mid \boldsymbol{\Theta}) = \sum_{g=1}^{G} \tau_g \, p(\mathbf{x}_i \mid c_g; \boldsymbol{\Theta}_g)$$

Model Tabanlı Demetleme Problemi

- Model parametrelerinin belirlenmesi
  - Maximum Likelihood (ML) yaklaşımı

$$\ell_{ML}(\Theta_1,...,\Theta_G;\tau_1,...,\tau_G \mid D) = \prod_{i=1}^{K} \sum_{g=1}^{G} \tau_g \, p(x_i \mid c_g, \Theta_g)$$

  - Maximum Aposteriori (MAP) yaklaşımı

$$\ell_{MAP}(\Theta_1,...,\Theta_G;\tau_1,...,\tau_G \mid D) = \prod_{i=1}^{K} \sum_{g=1}^{G} \frac{\tau_g \, p(x_i \mid c_g, \Theta_g) p(\Theta)}{p(D)}$$

  - Uygulamada her ikisinin logaritması

$$L(\Theta_1,...,\Theta_G;\tau_1,...,\tau_G \mid D) = \sum_{i=1}^{K} \ln \sum_{g=1}^{G} \left(\tau_g \, p(x_i \mid c_g, \Theta_g)\right)$$

$$L(\Theta_1,...,\Theta_G;\tau_1,...,\tau_G \mid D) = \sum_{i=1}^{K} \ln \sum_{g=1}^{G} \left(\tau_g \, p(x_i \mid c_g, \Theta_g)\right) + \ln p(\Theta)$$

EM Algoritması

- Veri kümesi: $D=\{x_1,x_2,\ldots,x_K\}$
- Gizli değişkenler $H=\{z_1,z_2,\ldots,z_K\}$ (her nesnenin hangi demete dahil olduğu bilgisi)
- Verinin eksik olduğu durumda, tam verinin beklenen değeri hesaplanır:

$$Q(\Theta,\Theta') = E[L_c(D,H\mid\Theta)\mid D,\Theta')$$
$$= \sum_{i=1}^{K}\sum_{g=1}^{G} p(c_g\mid x_i)[\ln p(x_i\mid c_g)+\ln\tau_g]$$

- EM Algoritmasının adımları:
  - Θ' için başlangıç değerleri atama
  - (E) Expectation: Q(Θ| Θ') hesaplanması
  - (M) Maximization: argmax Q(Θ| Θ')

http://www3.itu.edu.tr/~sgunduz/courses/verimaden/ 49

VERİ MADENCİLİĞİ
İlişkilendirme Kuralları

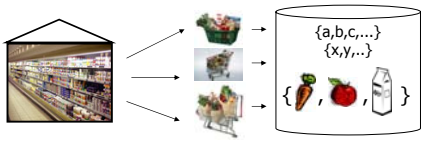
Yrd. Doç. Dr. Şule Gündüz Öğüdücü
http://www3.itu.edu.tr/~sgunduz/courses/verimaden/

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

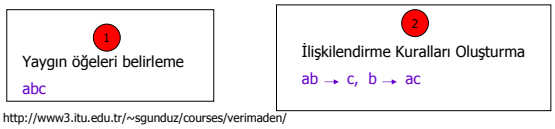
Örnek Kurallar:

{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)

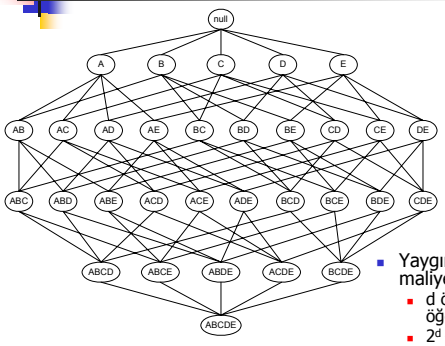
- Aynı öğeler kümesinin ikili bölünmesine (binary partition) ait kurallar
  - {Süt, Bez, Bira}
- Aynı öğeler kümesinden oluşan kuralların destek değerleri aynı, güven değerleri farklı
- Kurallar için destek ve güven şartları ayrı değerlendirilebilir

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:

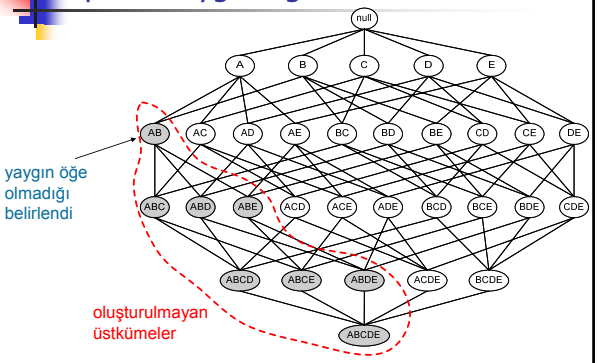
$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - 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

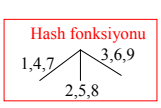
- Her yaygın öğe adayının destek değerinin belirlenmesi için veri kümesi taranır
  - Yaygın öğe adayı sayısı çok büyük olabilir
- Karşılaştırma sayısını azaltmak için yaygın öğe adayları hash ağacında saklanır
  - Her hareket bütün yaygın öğelerle karşılaştırılmak yerine hash ağacının ilgili bölümündeki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

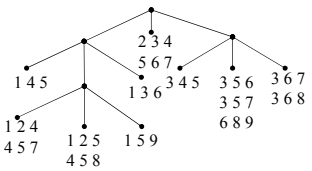
Hash Ağacı Oluşturma

- 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}

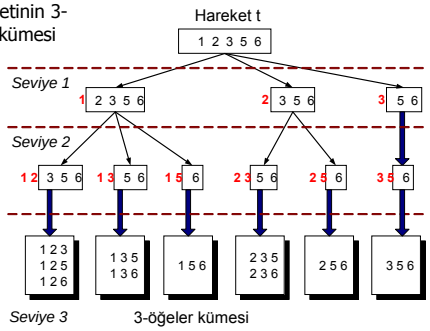
- Hash fonksiyonu kullanılarak bir hash ağacında saklanıyor
  - h(p) = p mod 3
- max yaprak boyutu: bir yaprakta bulunabilecek öğe kümesi sayısı



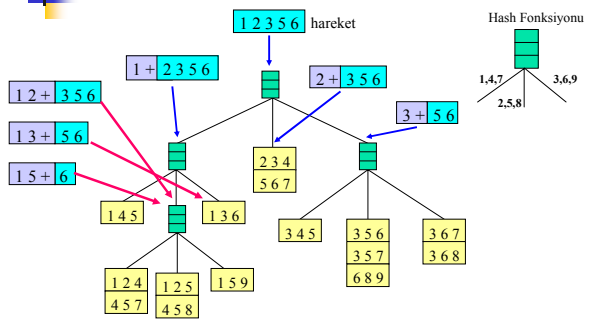


Bir Harekette Bulunabilecek Yaygın Öğeleri Belirleme

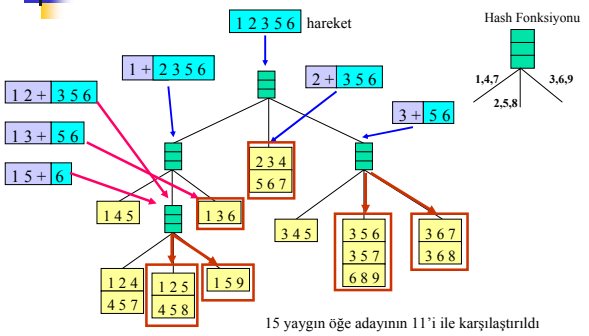
- t hareketinin 3-öğeler kümesi



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

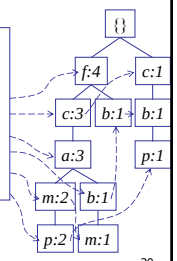
TID	*Öğeler*	*(sıralı) yaygın öğeler*
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

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

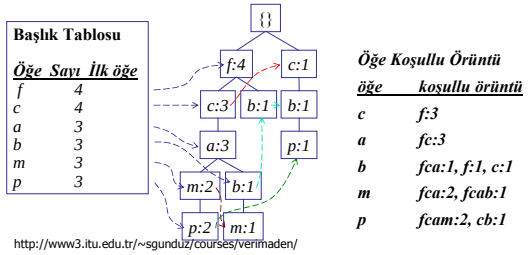
- Bütünlük
  - Yaygın öğeleri bulmak için gerekli tüm bilgiyi barındırır
- Sıkıştırılmış
  - Yaygın olmayan öğeler FP-ağacında bulunmaz
  - Destek sayısı daha büyük olan öğeler köke daha yakın
  - Asıl veri kümesinden daha büyük değil

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

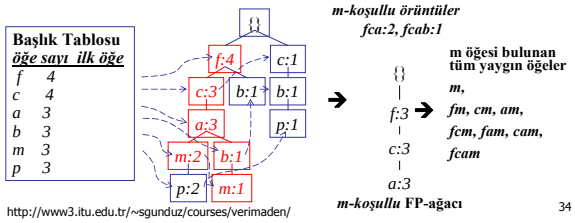
Öğe Koşullu Örüntü Oluşturma

- 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*)

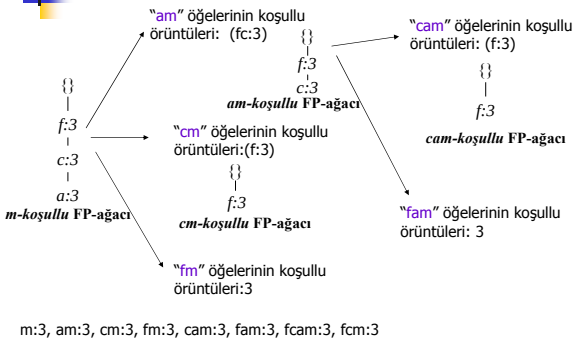


Öğe Koşullu Örüntülerden Koşullu FP-Ağaçları Oluşturma

- 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

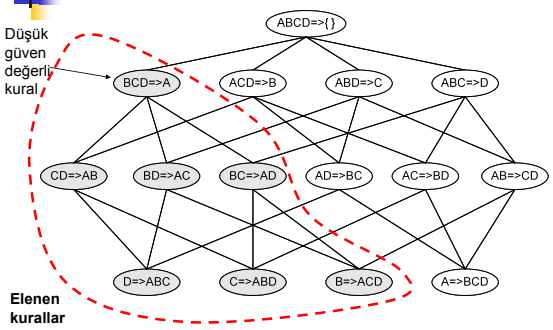
- L yaygın öğelerden f⊂L altkümelerinin bulunması
  - f→L-f kurallarının en küçük güven değeri koşulunu sağlaması gerekir
- Eğer {A,B,C,D} yaygın öğeler ise olası ilişkilendirme kuralları

  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.

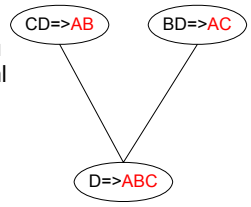
Apriori Algoritması için Kural Oluşturma



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