T.C.

PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VERİ KÜMELEME ALGORİTMALARININ PERFORMANSLARI ÜZERİNE KARŞILAŞTIRMALI BİR ÇALIŞMA

Mustafa Seçkin DURMUŞ

Yüksek Lisans Tezi

DENİZLİ – 2005

VERİ KÜMELEME ALGORİTMALARININ PERFORMANSLARI ÜZERİNE KARŞILAŞTIRMALI BİR ÇALIŞMA

Pamukkale Üniversitesi
Fen Bilimleri Enstitüsü
Tarafından Kabul Edilen
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

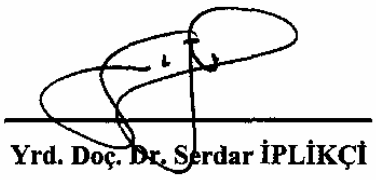
Mustafa Seçkin DURMUŞ

Tez Savunma Tarihi: 08.07.2005

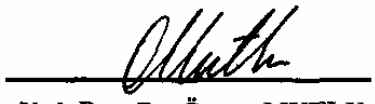
DENİZLİ – 2005

TEZ SINAV SONUÇ FORMU

Bu tez tarafımızdan okunmuş, kapsamı ve niteliği açısından Yüksek Lisans Tezi olarak kabul edilmiştir.



Yrd. Doç. Dr. Serdar İPLİKÇİ
(Yönetici)



Yrd. Doç. Dr. Özcan MUTLU
(Jüri Üyesi)

Yrd. Doç. Dr. Sezai TOKAT
(Jüri Üyesi)

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
..... tarih ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. M. Ali SARIGÖL
Müdür
Fen Bilimleri Enstitüsü

TEŞEKKÜR

Yüksek lisans tez çalışmalarım süresince kıymetli zamanını benden esirgemeyen, bilgi ve tecrübesi ile her konuda bana yön gösteren, düşünce ufkumu her yönüyle genişleten ve tezimin çok daha iyi olmasını sağlayan değerli danışman hocam Yrd. Doç. Dr. Serdar İPLİKÇİ'ye teşekkürlerimi sunarım.

Tez çalışması süresince fikirleri ile destek olan, yardımlarını esirgemeyen Öğretim Görevlisi Önder ÇİVRİL'e, Tel Aviv üniversitesinden Prof Dr. David HORN'a, Ofer PASTERNAK'a, California üniversitesinden Dr. Eamonn KEOGH'a, Bilgisayar Mühendisi Dursun GÜNDOĞAN'a, değerli bölüm öğretim üyelerim ve çalışma arkadaşlarım, Remzi ARSLANALP'a, Hilal Ezercan KAYIR'a, Ö. Önder KARAKILINÇ'a, Engin ÇETİN'e, Hüsnü ŞENTÜRK'e, Adem ÜKTE'ye, Ahmet ÇİFTÇİ'ye ve Mehmet ÜNAL'a, tezimin daha da mükemmel olmasını sağlayan Yrd. Doç. Dr. Özcan MUTLU ve Yrd. Doç. Dr. Sezai TOKAT'a teşekkürü bir borç bilirim.

Hayatımın her alanında olduğu gibi tez çalışma sürecinin her safhasında da yanımda olan anneme, babama, kardeşime sonsuz teşekkür ederim. Ayrıca çalışma tempoma hız katan, motivasyonumu artıran kıymetlime şükranlarımı sunarım.

Mustafa Seçkin DURMUŞ

ÖZET

Bu tezde, Veri Madenciliği metotlarından biri olan kümeleme tekniklerinden farklı veri kümeleme algoritmaları performanslarına göre karşılaştırmalı olarak incelenmiştir. Sık kullanılan kümeleme algoritmaları tanımlanmış ve bu algoritmalar arasından, kümeleme işlemi sonunda oluşacak küme sayısının ve hangi verinin hangi kümeye yerleştirileceğinin önceden bilinmediği (öğreticisiz öğrenme) algoritmalar karşılaştırma yapmak için seçilmiştir.

Seçilen bu algoritmalar farklı üç veri seti üzerinde (MATLAB ortamında oluşturulan rasgele veri seti, iris çiçeği veri seti ve Avustralya yengeçlerinden oluşturulmuş veri seti) gürültüye dayanıklılık, işlemler için kullanılan hafıza, işlem süresi ve işlemler esnasında kullandıkları flop sayılarına göre karşılaştırılmışlardır.

Tüm kümeleme algoritmaları veriye bağlıdır ve herhangi bir kümeleme algoritması tüm veri setleri için her zaman en iyi kümeleri oluşturmamaktadır. Bu nedenle, seçilen veriye en uygun algoritma belirlenmelidir.

Bu karşılaştırmalı çalışma için seçilen algoritmalar MATLAB simülasyon programı kullanılarak gerçekleştirilmiş ve her üç veri seti için seçilen tüm algoritmalar farklı eşik değerleri için denenmiştir. Sonuçlar arasında karşılaştırmalar yapılmıştır.

Anahtar kelimeler: Veri Madenciliği, veri kümeleme, kümeleme algoritmaları, çizge yapıları, en yakın komşu, en küçük tarama ağacı, karşılıklı komşuluk, destek vektörleri.

ABSTRACT

In this study, a comparative study on performances of different data clustering algorithms which is a way of data mining method is considered. Commonly used clustering algorithms are defined and among these algorithms in which resulting cluster number and which data is going to be placed in which cluster (unsupervised learning) are not to be known before clustering, were chosen for comparative study.

These algorithms are examined on three different data sets (A random data set generated by MATLAB, the iris data set and the Australian crab data set) for their endurance of noise, memory used for processes, process time and flop numbers.

All clustering algorithms are data dependent and an algorithm is not being always capable for all data sets. Therefore, the most suitable algorithm must be determined for the chosen data set.

Algorithms for this comparative study are realized by MATLAB and all algorithms are tested for different threshold values. Comparisons were made between different results.

Keywords: Data mining, data clustering, clustering algorithms, graph structures, nearest neighbor, minimum spanning tree, mutual neighborhood, support vectors.

İÇİNDEKİLER

Birinci Bölüm
GİRİŞ

İkinci Bölüm
TANIMLAR

Üçüncü Bölüm

KÜMELEME YÖNTEMLERİ VE ALGORİTMALARI

Dördüncü Bölüm

TEZDE KULLANILAN ALGORİTMALAR

Beşinci Bölüm

BENZETİM SONUÇLARI

Altıncı Bölüm

SONUÇLAR VE YORUMLAR

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER DİZİNİ

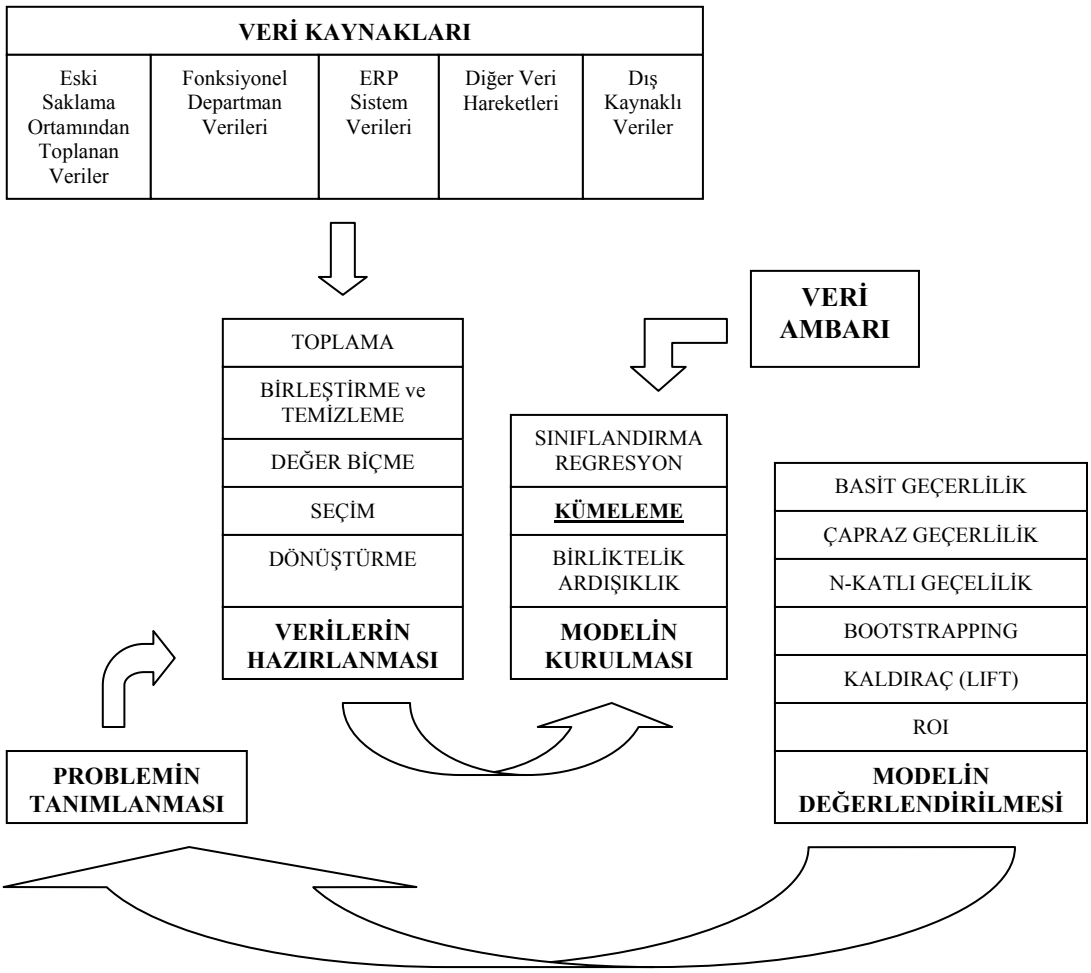
x	Skaler gösterimi
$\mathbf{x}$	Vektör gösterimi
$\mathbf{X}$	Matris gösterimi
$\mathbf{D}(i,j)$	Yakınlık matrisi
$d\left(\mathbf{x}_i,\mathbf{x}_j\right)$	İki nokta arasındaki uzaklık
$\mathbf{C}$	Kovaryans matrisi
$\boldsymbol{G}$	Çizge gösterimi
q	Gaussian kernel fonksiyonunun genişlik değeri
$K\left(\mathbf{x}_i,\mathbf{x}_j\right)$	Kernel fonksiyonu
$\left\| \cdot \right\|^2$	Öklit uzaklığı
Φ	Noktaları daha büyük boyutlu uzaya taşımak için kullanılan dönüşüm
L	Lagrange ifadesi
ξ_i	Gevşek değişkenler
μ_i , β_i	Lagrange katsayıları
W	Lagrange ifadesinin ikincil durumu
σ_v^2	Veri setlerinin bileşenlerinin değişkesi
σ_η^2	Veri setlerinin bileşenlerine eklenen gürültünün değişkesi

BİRİNCİ BÖLÜM
GİRİŞ

1. GİRİŞ

Günümüzde firmaların veri tabanı boyutları terabaytlar cinsinden ifade edilmektedir. İstenilen ise bu büyük veri tabanından amaç doğrultusunda bilginin elde edilmesidir. *Bilgi* ise herhangi bir amaca yönelik veri olarak tanımlanmaktadır. Veriyi bilgiye çevirme işine de *Veri Analizi* denilmektedir. *Bilgi* kelimesi aynı zamanda herhangi bir soruya yanıt verebilmek için veriden çıkarılan sonuç olarak da tanımlanmaktadır.

Tüm bu açıklamalar doğrultusunda *Veri Madenciliği*, "*büyük miktarda veri içerisinden, önceden bilinmeyen fakat potansiyel olarak kullanışlı bilginin bilgisayar programları kullanılarak aranmasıdır*" şeklinde tanımlanabilir. Veri madenciliği de *kümeleme*, veri özetleme, sınıflandırma, değişikliklerin analizi, sapmaların tespiti, karar ağaçları gibi belli sayıda teknik yaklaşımın kullanılmasıyla gerçekleştirilmektedir. Veri madenciliğinde *verinin önemi* (ne kadar çok örnek toplanırsa o kadar iyi sonuçlar elde edilmektedir), *uzmanın önemi* (algoritma seçiminde ve elde edilen sonuçların değerlendirilmesinde önemli rol oynamaktadır) ve *sabrın önemi* (çok büyük ver tabanları ile işlem yapıldığından dolayı sonuçlara ulaşmak için zamana ihtiyaç duyulmaktadır) kavramları dikkat edilmesi gereken üç önemli hususdur. Şekil 1.1'de veri madenciliği kavramında izlenmesi gereken yol görülmektedir. Veriler hazır olarak veri ambarlarından alınabileceği gibi dış veri kaynaklarından da alınabilmektedir. Fakat dışarıdan alınan veriler doğrudan modelin kurulması amacıyla kullanılamamaktadır. Bu nedenle modelde kullanılmak üzere hazırlanmaktadırlar. Model kurulduktan sonra farklı kriterlere göre incelemeler yapılıp önceden tanımlanmış probleme bu kurulan model uygulanmaktadır.

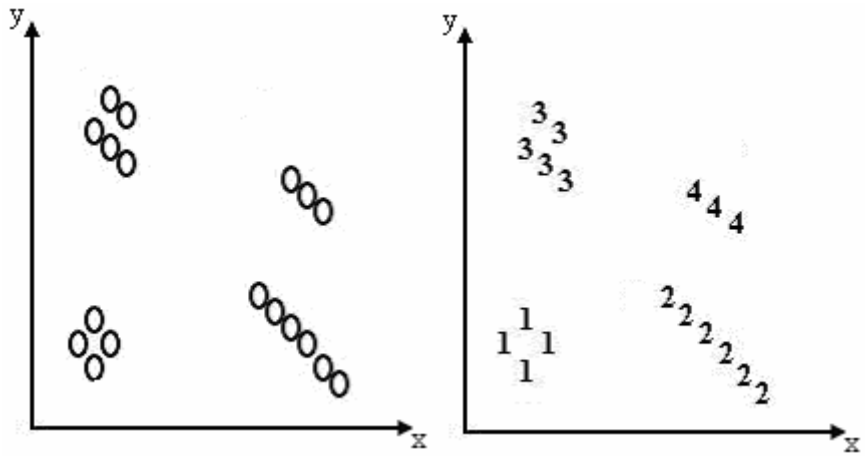


Şekil 1.1 Veri Madenciliği

Kümeleme, bir çeşit sınıflandırmadır (Jain ve Dubes, 1988). Diğer bir ifadeyle, önceden elde edilmiş nesnelerin (öznitelik vektörleri, gözlemler veya veri öğeleri), farklı gruplara (kümelere), herhangi bir öğretici olmadan (Öğreticisiz Öğrenme) sınıflandırılmasıdır. Oluşturulmuş herhangi bir kümede bulunan nesneler, farklı kümelerde bulunan nesnelere göre birbirlerine daha fazla benzemektedir. Nesneler arasındaki ilişkiler, satır ve sütunları nesnelerden oluşan yakınlık matrisi ile ifade edilmektedir. Bu nesneler "örnek (pattern)" olarak tanımlanmış ise, aralarında bulunan yakınlık, uzaklıklar şeklinde ifade edilebilmektedir (Öklit Uzaklığı-Euclidean Distance gibi). Nesneler arasında herhangi bir uzaklık ölçümü yapılamıyorsa veya yakınlık değerleri bulunamıyorsa, kümelendirme yapılması imkansız olmaktadır. *Küme Analizi* (Clustering Analysis) örneklerin (örnekler genellikle çok boyutlu uzayda bulunan noktalar veya bir ölçüm sonunda elde edilen noktalar şeklinde ifade edilmektedir),

birbirlerine olan benzerlikleri göz önünde bulundurularak, bir araya toplanması olarak tanımlanmaktadır (Jain ve diğerleri, 1999).

Kümelendirme ve *Diskriminant Analiz* birbirine karıştırılmamalıdır. Diskriminant Analiz yapılırken, önceden sınıflandırılmış örnekler ile işlem yapılmaktadır. Amaç, yeni karşılaşılan ve önceden sınıflandırılmamış örneklerin sınıflandırılmaya dahil edilmesidir. Özetle, önceden sınıflandırılmış örnekleri kullanarak yeni örneklerin bu sınıflardan herhangi birine dahil edilmesi işlemidir. Kümelendirme ise, verilen örneklerin ön sınıflandırma yapılmadan Şekil 1.2'deki gibi anlamlı kümelere ayrılmasıdır, burada doğrudan veriler üzerinde işlem yapılmaktadır.



Şekil 1.2 Verilerin Kümelere Ayrılması

1.1 Literatür Özeti ve Kümelemeye Genel Bir Bakış

İlk olarak 70'li yılların başında, verimliliği arttırmak amacı ile çalışmalara başlanmıştır. İlerleyen zamanlarda veri madenciliği ve kümeleme üzerine yazılan ilk kitaplardan biri olan fakat tek bir yaklaşımın kullanıldığı bir kitap yazılmıştır "Cluster Analysis" (Tryon&Bailey, 1970). Buna ek olarak, daha çok kümeleme işleminin matematik kısmının incelendiği "Mathematical Taxonomy" (Jardine&Sibson, 1971), veri madenciliği ve kümeleme üzerine yazılmış olan en kapsamlı kitap "Cluster Analysis for Applications" (Anderberg, 1973), sıradüzensel kümelemenin anlatıldığı bir çalışma, "Numerical Taxonomy" (Sneath&Sokal, 1973), değişik projelerin toplandığı

bir kitap olan, “Clustering Algorithms” (Hartigan-1975), “Algorithms for Clustering Data” (Jain&Dubes, 1988) geniş kapsamlı bir kitap ve 1990’dan itibaren konu ile ilgili olarak çok farklı uygulamalar ve kitaplar ortaya konulmuştur.

Farklı uygulamalarda kullanılabilen çok çeşitli kümeleme algoritmaları bulunmaktadır. Literatürde birçok yeni kümeleme algoritmaları ortaya çıkmaya devam etmektedir. Genel olarak bu algoritmalar iki başlıkta toplanmaktadır: Geleneksel Algoritmalar ve Yeni Nesil Algoritmalardır. Geleneksel algoritmalar da Sıradüzensel ve Paylaştırmalı algoritmalar şeklinde iki alt gruba ayrılmaktadır.

Sıradüzensel algoritmalar, veriyi iç içe sıralı diziler haline getirmektedir ve bu sıralı diziler dendrogramlar (sıradüzensel yapıyı gösteren çizimler) ile gösterilmektedir. Bu gösterimden kümeleri elde edebilmek için örnekler arasında bulunan yakınlık değerlerine göre bir eşik değeri (threshold) seçilmeli ve dendrogramlar bu seçilen eşik değerlerine göre kesilerek, kümeler belirlenmelidir. Farklı toplayıcı sıradüzensel algoritmalar, örnek ve küme arasında veya iki küme arasında bulunan yakınlık değerlerinin tanımlanmasına göre birbirlerinden farklılık göstermektedirler.

Paylaştırmalı algoritmalarda ise küme içi dağılımı en aza indiren veya kümeler arası dağılımı en yüksek değere çıkaran paylaşımlar elde edilmektedir. Genel en uygun bir sonuç elde edebilmeyi garantilemek için, uygun olmayan olası tüm paylaşımlar tespit edilmelidir. Sıradüzensel yöntemler, biyolojik, sosyal ve davranışsal bilim dallarında yaygın olarak kullanılmaktadır. Paylaştırmalı yöntemler ise, daha çok mühendislik alanlarında kullanılmaktadır (En Küçük Tarama Ağacı (Minimum Spanning Tree), Karesel-Hata Metodu (Squared Error Method), K-Yol algoritması (K Means), En Yakın Komşu Algoritması (Nearest Neighbor), PAM, CLARANS, Genetik Algoritmalar, Yapay Sinir Ağları, v.b.).

Yeni nesil algoritmalarda veritabanı, boyutundan bağımsız olarak sıkıştırılabilen veya budanabilen veri belleğine yerleştirilir. Geniş veritabanlarında kümeleme yapmak için bazı ölçütler belirlenmiştir. Bunlar, veritabanının bir kez veya daha az taranması, çevrimiçi çalışabilme özelliği, askıya alınabilme, durdurulabilme ve geri dönülebilir

olma özellikleri, veri ekleme veya çıkarma sonucunda güncelleme imkanı, kısıtlı bellek ile çalışabilme, tarama sırasında farklı teknikler kullanabilme ve bir kaydın sadece bir kez işlenmesi şeklindedir (BIRCH, DBSCAN, CURE, ROCK, v.b.).

Mevcut verileri kümelere ayıracak tek bir algoritma bulunmamaktadır bu nedenle çeşitli algoritmalar denenmelidir. Küme analizi keşifsel veri analizinde kullanılan araçlardan sadece bir tanesidir. Verilerin toplanması ve sunulması, kümeleme sonuçlarının değerlendirilmesi ve bulunan kümelerin tanımlanması en az kümeleme stratejisinin seçimi kadar önemlidir.

Uygulama alanlarına kısaca değinecek olursak, pazar bölümlerinin ayrılması, müşteri değerlendirme ve çapraz satış analizleri (pazarlama), risk analizleri, usulsüzlüklerin tespiti, müşteri kazanma ve mevcut müşterileri elde tutma analizleri (bankacılık, daha çok veri madenciliğinin bir alt koludur), ana giderlerin azaltılması, poliçe fiyatlarının belirlenmesi (sigortacılık), satış noktası veri analizleri, alış-veriş sepeti analizleri (perakendecilik), hisse senedi fiyat tahmini, genel piyasa analizleri, en iyi alım-satım stratejilerinin belirlenmesi (borsa), hatların yoğunluk tahminleri (haberleşme), test sonuçlarının tahmini, ürün geliştirme, ilaçlarda kullanılan maddelerin sınıflandırılması (ilaç sanayi), tıbbi teşhis, uygun tedavi sürecinin belirlenmesi (sağlık), kalite kontrol, lojistik, üretim süreçlerinin en iyileştirilmesi (endüstri) gözlemsel veriler üzerinde modeller kurularak bilimsel ve teknik problemlerin çözümlenmesi, çeşitli tahminler ve sınıflandırma problemlerinin ayrıştırılarak çözümlenmesi (bilim ve mühendislik) şeklinde sıralamak mümkündür (Hartigan, 1975).

Kümeleme çeşitli keşifsel örnek analizlerinde (Exploratory Data Analysis) de kullanılmaktadır. Bunlar, veri madenciliği, dokümanların tekrar kazanılması, örnek sınıflandırma ve görüntülerin parçalara ayrılması konularının da içinde bulunduğu, gruplama, makine öğrenmesi ve karar verme sistemleridir. Her araştırma sahasının kendine özgü terimleri, metotları ve algoritmaları bulunmaktadır.

1.2 Tez Tanıtımı

Bölüm 2'de kümeleme işlemlerinde dikkat edilmesi gereken hususlar, sıkça kullanılan tanımlamalar ve karşılıkları bulunmaktadır. Bölüm 3'te literatürde bulunan kümeleme algoritmaları anlatılmaktadır. Bölüm 4'te tezde kullanılan kümeleme algoritmaları tanımlanmaktadır. Bölüm 5'te kullanılan veri setleri ve simülasyon sonuçları bulunmaktadır. Son olarak, Bölüm 6'da elde edilen sonuçlar, değerlendirmeler ve bir önceki bölümde elde edilen simülasyon sonuçlarına ilişkin yorumlar bulunmaktadır.

İKİNCİ BÖLÜM
TANIMLAR

2. TANIMLAR

2.1 Örnek

Örnek (Pattern), kümeleme algoritması tarafından kullanılan veri öğeleridir ve genellikle yapılan ölçümlerin sonuçlarını içermektedir. Örnek vektörünün her bir sayısal elemanı da (x_i), yani verilerin uzaklık bileşenleri, öznitelik (attribute) olarak tanımlanmaktadır.

$$\mathbf{x_i} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \ (i = 1,\ 2,\ ...,\ n) \tag{2.1}$$

Buradaki d örnek uzayının boyutunu, n örnek sayısını ifade etmektedir. Örnek kümesi,

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \end{bmatrix} \Rightarrow \mathbf{X} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{n1} \\ x_{12} & x_{22} & \cdots & x_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1d} & x_{2d} & \cdots & x_{nd} \end{bmatrix} \tag{2.2}$$

ile gösterilmektedir. Örnek dizisi $n \times d$ örnek matrisi (pattern matrix) olarak da ifade edilebilmektedir. Bu matrisin her satırı örnekleri ve her sütunu da öznitelikleri veya ölçümleri ifade etmektedir. Kullanılan veriler buna benzer olarak tanımlanmıştır fakat ayrı ayrı öznitelikler şeklinde tanımlanmamıştır (Jain ve Dubes, 1988).

2.2 Yakınlık Matrisleri

Kümeleme metotları veri çiftleri arasındaki yakınlıkları, benzerlikleri veya ilişkileri oluşturulabilmek için bir göstergeye ihtiyaç duymaktadır. Yakınlık Matrisi (Proximity Matrix), $\left[\mathbf{D}(i,j)\right]$, ile ifade edilen, satır ve sütunlarında örnek numaralarının bulunduğu ve köşegen üzerinde bulunan tüm değerlerin sıfır olduğu simetrik bir matristir. Tüm yakınlık matrisleri simetriktir. Yakınlık değeri ne kadar büyük olursa o değere karşılık gelen satır ve sütunda bulunan örneklerin birbirlerine olan benzerlikleri de büyük olmaktadır (Jain ve diğerleri, 1999).

$$\left[\mathbf{D}(i,j)\right]=\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1m} \\ d_{21} & d_{22} & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nm} \end{bmatrix}=\begin{bmatrix} 0 & d_{12} & \cdots & d_{1m} \\ d_{21} & 0 & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix} \tag{2.3}$$

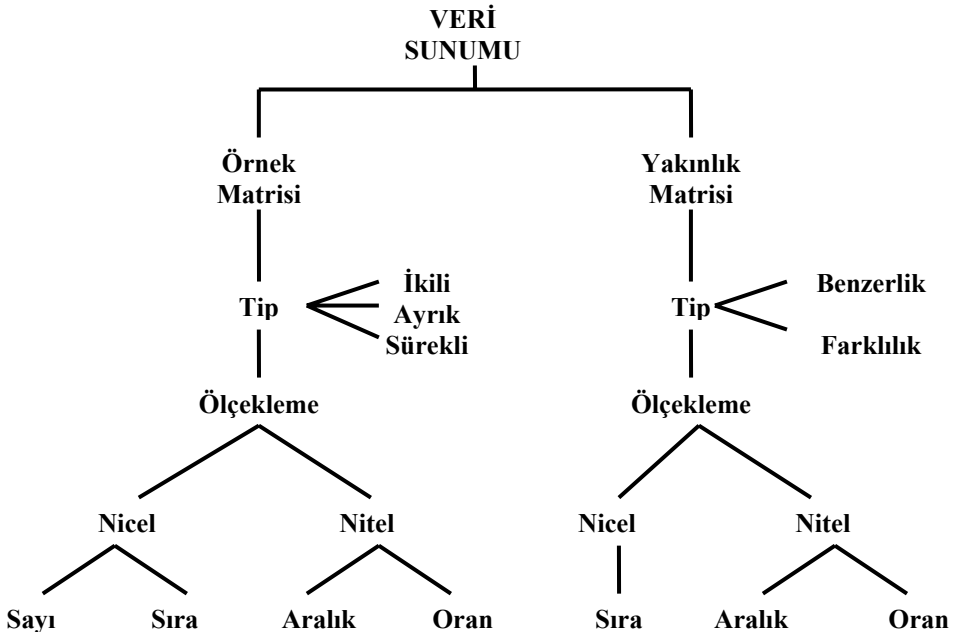
2.3 Veri Tipleri

Elde bulunan mevcut verinin tipi (Data Type) ve değişim aralığı kümeleme algoritmasının belirlenmesinde kullanılmaktadır. Veri tipi ile belirtilmek istenen veri miktarının derecesidir. Öznitelik, ikili (binary), ayrık (discrete) veya sürekli (continuous) şekilde tanımlanabilir. İkili öznitelikler evet-hayır sorularında olduğu gibi iki değere sahiptir. Ayrık veriler genellikle küçük ve sonlu değerlerdir. Sürekli veriler ise belirli sınırlar dahilinde gerçek değerlere sahiptirler. Yakınlık matrislerinde bulunan değerler, Şekil 2.1'de de görüldüğü gibi, yakınlık matrisinde bulunan değerler yukarıda tanımlanan üç farklı tipte de olabilmektedir.

İkinci bir özellik ise verilerin sayıların karşılıklı ilişkilerini gösterecek şekilde ölçeklenmesidir. Veriler nitel (qualitative) ve nicel (quantitative) olmak üzere iki şekilde ölçeklenmektedir. Nitel özellikler bir sıra (ordinal) veya sayı (nominal) değeri gösterirken, nicel ifadeler ise bir aralık değeri (interval) veya bir oran (ratio) ifade etmektedir.

Örneğin evet-hayır sorusu (0-1) veya (50-100) şeklinde ifade edilebilmektedir (nominal ölçekleme). Sayıların kendileri anlamsızdır. Sıraya göre ölçekleme de ise sayılar arasındaki ilişkilere dikkat edilir (1-2-3 veya 10-20-30 veya 1-100-200 gibi). Aralık değerleri ile ölçekleme yapılırken istenilen aralık değerleri veriler üzerine uygulanır mesela 100 kişilik bir gruptaki insanlara boylarına göre 45-55 arası puan verilmesi bu tip ölçeklemeye örnektir. En çok kullanılan ölçekleme tipi oransal ölçeklemedir, bu tip ölçeklemede sayılar tam değerlere sahiptir. Mesela iki şehir arasında bulunan mesafe metre, mil ve inç cinsinden ayrı ayrı ölçülebilmektedir, bu iki şehrin birinden diğerine araba ile giden bir insana göre gittiği yol değişmemektedir, benzer şekilde bir insanın gelirinin iki katına çıkarılması hangi para birimi kullanılırsa kullanılsın satın alma gücünü iki katına çıkaracaktır.

Veri tipinin belirlenmesi, yakınlık matrisinin oluşturulmasında ve küme analizi sonuçlarının gösterilmesinde önemli rol oynamaktadır. İnsanlar ikili, nitel verileri üretmede iyi iken aygıtlar sürekli nicel verilere ihtiyaç duymaktadır. Verinin güvenilirliği veri tipine ve ölçeklenmesine bağlıdır (Jain ve Dubes, 1988).



Şekil 2.1 Veri Tipleri

2.4 Yakınlık İfadeleri

Bu bölümde çok kullanılan yakınlık ifadelerine yer verilmektedir. *i*. ve *k*. örnekler arasında bulunan yakınlık değeri $d(i,k)$ ile gösterilmektedir ve aşağıdaki şartları sağlamalıdır:

(i) Tüm i değerleri için, $d(i,i)=0$

(ii) Tüm (i,k) değerleri için, $d(i,k)=d(k,i)$

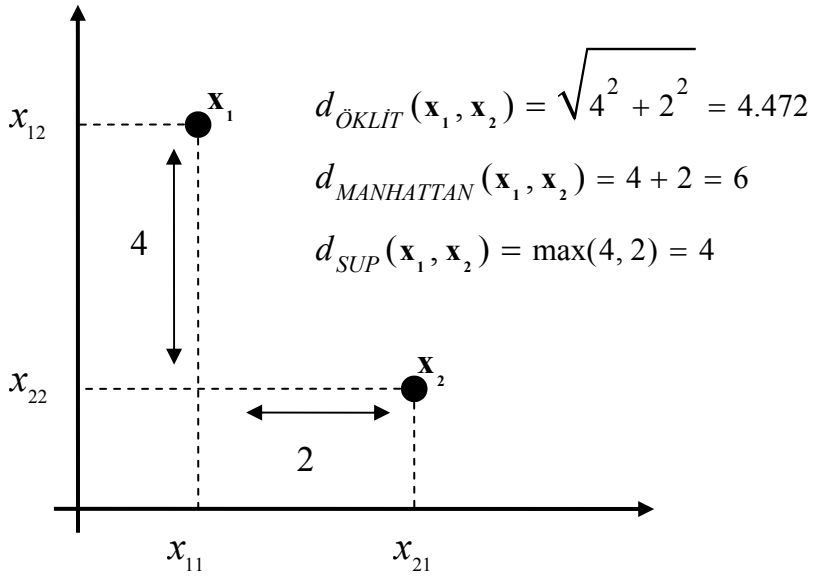
(iii) Tüm (i,k) değerleri için, $d(i,k)\geq 0$

olmalıdır.

Yakınlık değerleri çeşitli şekillerde ifade edilmektedir. En çok kullanılan yakınlık değerleri aşağıda gösterilmektedir. Bunlara *Minkowski Ölçütleri* (Minkowski Metrics)'de denilmektedir (Şekil 2.2). Tüm Minkowski Ölçütleri yukarıda bulunan üç şarta ek olarak aşağıdaki iki şartı da sağlamalıdır:

(iv) Sadece $x_i = x_k$ durumunda $d(i,k)=0$ olur.

(v) Tüm *i*,*k* ve *m* değerleri için, $d(i,k)\leq d(i,m)+d(m,k)$



Şekil 2.2 Minkowski Ölçekleri

Minkowski ölçütlerinin genel hali (2.5)'de görülmektedir:

$$d(i,k)=\left(\sum_{j=1}^{d}\left|x_{ij}-x_{kj}\right|^{r}\right)^{1/r} \quad ; \quad r\geq 1 \tag{2.4}$$

Denklem 2.5'de $r=2$ seçilirse, Öklit Uzaklığı (Euclidean Distance) elde edilir ($\|\bullet\|$ ile de gösterilmektedir):

$$d(i,k)=\left(\sum_{j=1}^{d}\left|x_{ij}-x_{kj}\right|^{2}\right)^{1/2}=\sqrt{\left(\mathbf{x}_i-\mathbf{x}_k\right)^{\mathrm{T}}\left(\mathbf{x}_i-\mathbf{x}_k\right)} \tag{2.5}$$

Denklem 2.5'de $r=1$ seçilirse, Manhattan Uzaklığı elde edilir:

$$d(i,k)=\sum_{j=1}^{d}\left|x_{ij}-x_{kj}\right| \tag{2.6}$$

Denklem 2.5'de $r=\infty$ seçilirse, Supremium Uzaklığı (Sup distance) elde edilir:

$$d(i,k)=\max_{1\leq j\leq d}\left|x_{ij}-x_{kj}\right| \tag{2.7}$$

Bunlar arasında en sık kullanılan uzaklık ölçümü öklit uzaklık ölçümüdür. Eğer tüm uzaklık değerleri ikili (binary) ise Manhattan Uzaklığı'na *Hamming Uzaklığı* denilmektedir. Mahalanobis Uzaklığı ise kullanılan ortak kovaryans matrisi (covariance matrix)'nin kullanılmasıyla öklit uzaklığından farklılık göstermektedir ve aşağıdaki gibi hesaplanır:

$$d(i,k)=\sqrt{\left(\mathbf{x_i}-\mathbf{x_k}\right)^{\mathrm{T}}\mathbf{C^{-1}}\left(\mathbf{x_i}-\mathbf{x_k}\right)} \tag{2.8}$$

Bu tanımlanan uzaklık yöntemleri dışında daha pek çok uzaklık ölçüm yöntemi bulunmaktadır (http://mathworld.wolfram.com).

2.4.1 Ortak Kovaryans Matrisi

Herhangi bir **M** matrisinin kovaryans matrisi (Covariance Matrix) aşağıdaki formül ile elde edilmektedir. Aşağıdaki ifadede bulunan $\bar{\mathbf{x}}$ vektörü ortalama değerlerin bulunduğu vektördür ve her sütunun (örneğin) ortalama değerini içermektedir (http://planetmath.org).

$$\mathbf{C} = \frac{1}{n-1}\sum_{i=1}^{n}\left[\left(\mathbf{M_i} - \bar{\mathbf{x}}\right)\left(\mathbf{M_i} - \bar{\mathbf{x}}\right)^{\mathbf{T}}\right] \tag{2.9}$$

$$\mathbf{M} = \begin{bmatrix} 4 & 2 & 0.6 \\ 4.2 & 2.1 & 0.59 \\ 3.9 & 2 & 0.58 \\ 4.3 & 2.1 & 0.62 \\ 4.1 & 2.2 & 0.63 \end{bmatrix} \tag{2.10}$$

$$\bar{\mathbf{x}} = \begin{bmatrix} 4.10 & 2.08 & 0.604 \end{bmatrix} \tag{2.11}$$

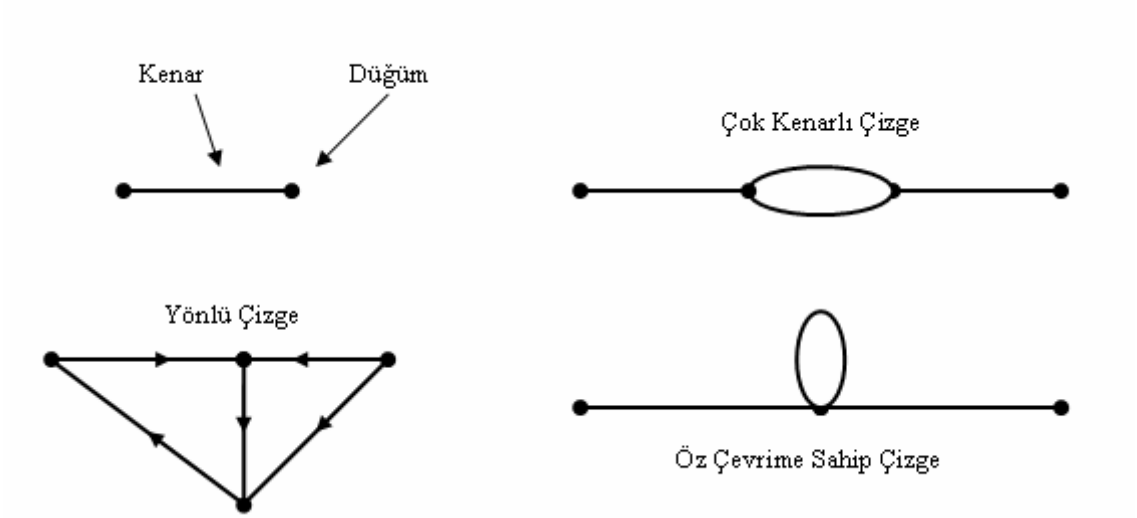
$$\mathbf{C} = \begin{bmatrix} 0.025 & 0.0075 & 0.00175 \\ 0.0075 & 0.007 & 0.00135 \\ 0.00175 & 0.00135 & 0.00043 \end{bmatrix} \tag{2.12}$$

2.5 Çizge Kuramı

Çizge (Graph), kümeleme işlemlerinde çok farklı uygulama alanlarına sahip matematiksel bir yapıdır. Bu bölümde kısaca bu tanımlamalar incelenmiştir. Bir çizge G ile gösterilmektedir (Şekil 2.3). G çizgesi, düğümlerden (V), kenarlardan (E) ve bunların birbirleri ile olan ilişkilerini gösteren bir fonksiyondan (f) oluşmaktadır ve G=<V',E',f'> ile gösterilmektedir. Kümeleme işlemlerinde kullanılan çizgilerin öz çevrimlere sahip olmadıkları kabul edilmektedir. Ayrıca kenarların yönleri önemli olmadığından dolayı yönsüz (undirected) çizgeler olarak da nitelendirilmektedirler.

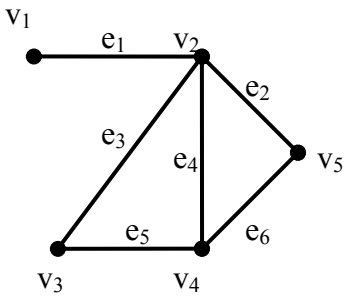
Kümeleme işlemlerinde kenarlar iki düğüm (örnek nokta) arasında bulunan uzaklık değerini ifade etmektedir.

Alt çizge, asıl çizgeden elde edilmektedir ve G=<V',E',f> ile ifade edilmektedir. Bir alt çizge asıl çizgenin tüm noktalarını içermelidir. Şekil 2.4'de alt çizge olan (b, d, e) ve olmayan (c) çizgeler verilmektedir. Diğer çizgenin alt çizge olmamasının sebebi asıl çizgede olmayan bir kenar içermesidir. Yine bu alt çizge tanımına benzer olarak *yol* ifadesi tanımlanmıştır. Yol ise, yine bir alt çizgedir fakat bu alt çizgede öz çevrim ve tekrarlanan kenar bulunmamaktadır (Şekil 2.5). Her yol bir alt çizgedir fakat her alt çizge bir yol değildir. Bir çizgede bulunan herhangi iki düğüm arasında bir yol bulunuyorsa *bağlı* (connected) ifadesi kullanılmaktadır. *Bileşen* (component) ise bağlı çizgenin azami (maximal) parçasıdır. Eğer bir kenar tüm düğümlere bağlı ise *tam* (complete) çizge olarak adlandırılmaktadır. n düğüme sahip bir tam çizge $n(n-1)/2$ kenar içermektedir.



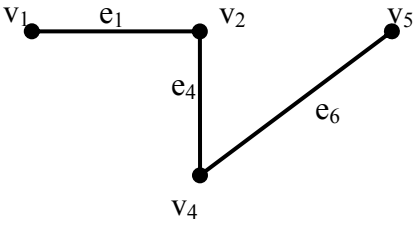
Şekil 2.3 Çizge Tanımlamaları

5 Düğümlü ve 6 Kenarlı bir Çizge



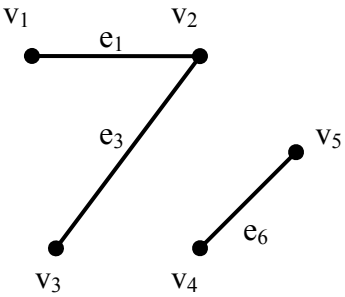
$f(e_4) = (v_4, v_2)$

a



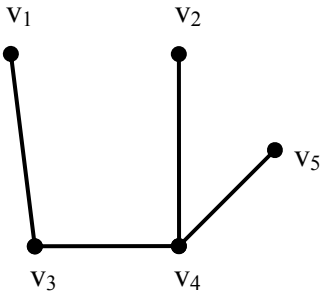
Bağlı Alt Çizge

b



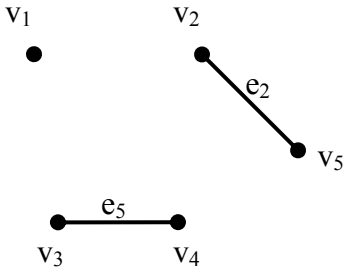
İki Bileşenli Alt Çizge

d



Alt Çizge Değil

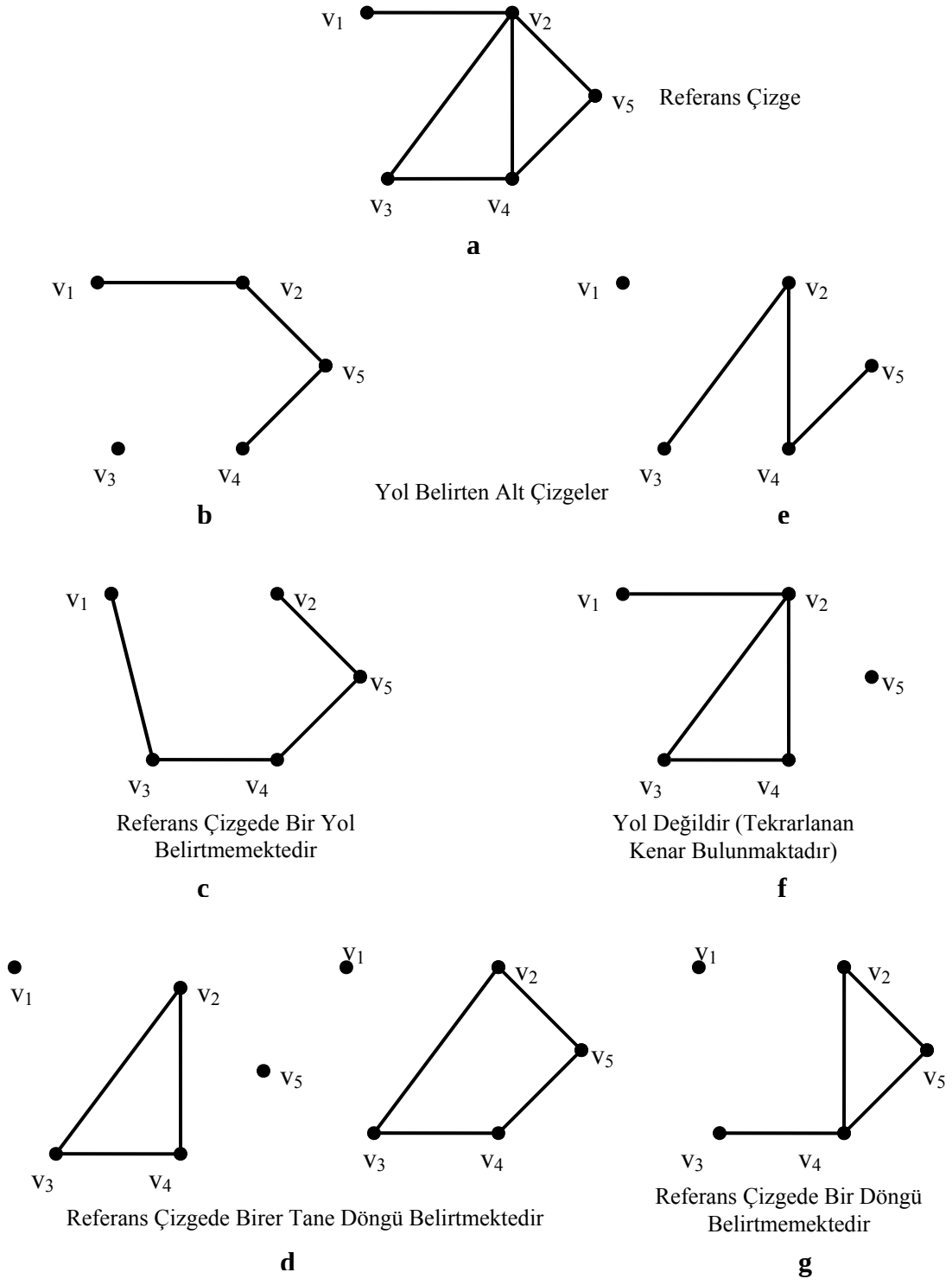
c



Üç Bileşenli Alt Çizge

e

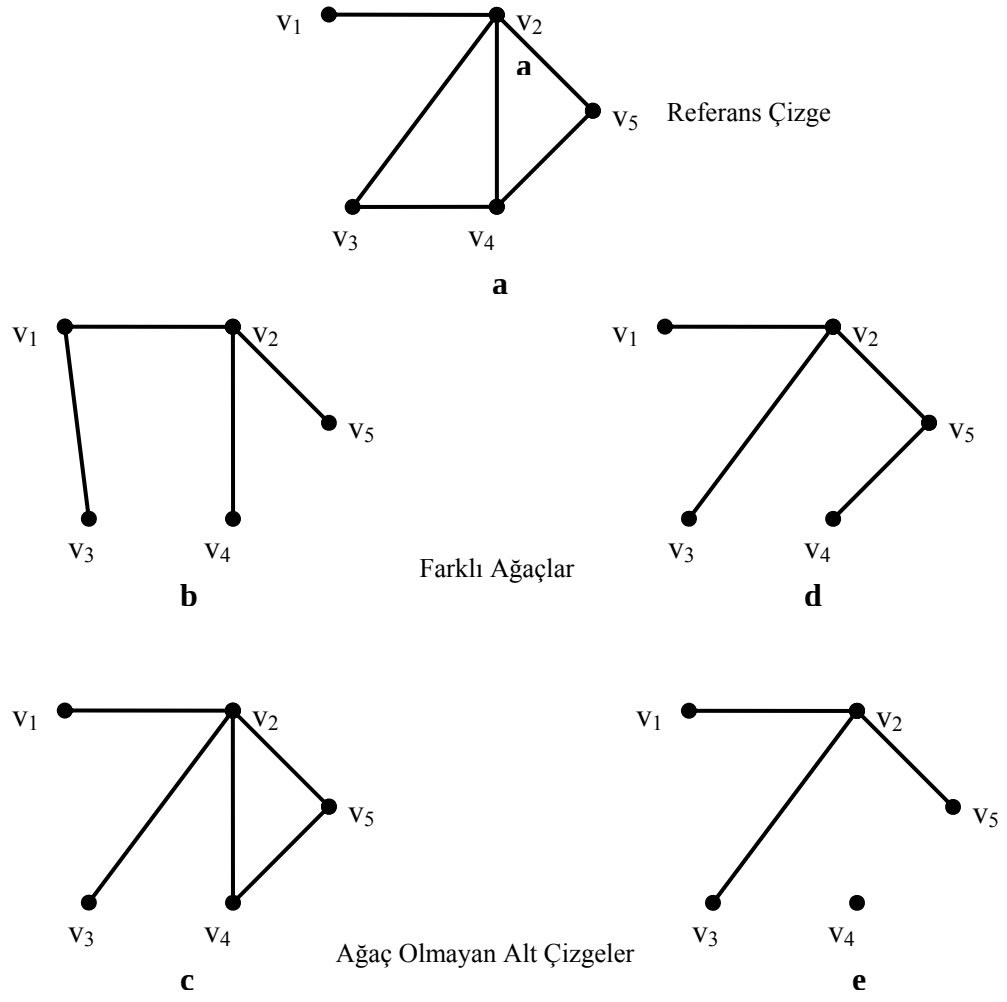
Şekil 2.4 Alt Çizgeler



Şekil 2.5 Çizgelerin Özellikleri

Döngü (cycle) tanımı ise yol tanımına benzemektedir fakat burada başlangıç düğümü ile bitiş düğümü aynıdır. *Ağaç*, çizge kuramında döngü içermeyen bağlı grafiklere

verilen isimdir. Eğer bir alt çizge m tane düğüme sahip ise, oluşturulan ağacın tam olarak m–1 tane kenarı bulunmaktadır. *Tarama Ağacı* (spanning tree), çizgede bulunan tüm düğümleri kapsayan ağaçtır. Çizgede bulunan kenarlara birer uzaklık (ağırlık) değeri verilirse, ağacın ağırlığı (uzunluklar toplamı) kenarlara verilen uzaklık değerlerinin toplanması ile elde edilmektedir. *En Küçük Tarama Ağacı* (minimum spanning tree - mst), çizgeden faydalanılarak oluşturulabilen ağaçlar arasında ağırlıklar toplamı en küçük olan ağaçtır.



Şekil 2.6 Ağaçlar

Şekil 2.6'da farklı ağaç yapıları görülmektedir (Zahn C. T., 1970, Jain ve Dubes, 1988, http://mathworld.wolfram.com/).

2.6 Ultrametrik Eşitsizlik

Üçgen eşitsizliğinin farklı bir uyarlaması olan aşağıdaki eşitsizliği tüm x, y, z değerleri için sağlayan uzunluk ölçütüne ultrametrik denilmektedir (http://mathworld.wolfram.com).

$$d(x,z) \le \max\left(d(x,y), d(y,z)\right) \tag{2.13}$$

($d(x,z), d(x,y)$ ve $d(y,z)$ çiftlerinden en az ikisi aynı olmalıdır).

2.7 Kophenetik Matris Ve Kophenetik Uzaklık

Kophenetik mesafe (cophenetic distance, d_C) denklem 2.13'de tanımlanan ultrametrik eşitsizliği sağlayan yakınlık değerleridir. Yani x_i ve x_j elemanlarının ilk olarak aynı kümeye yerleştirilme seviyelerini ifade etmektedir ($d_C(i,j) = L(k_{ij})$). Kophenetik matris (cophenetic matrix) ise bu yakınlık değerlerinden oluşan matrise verilen isimdir. Aşağıda sırasıyla verilen yakınlık matrisi için önce tek-bağ (single-lik) daha sonra tam-bağ (complete-link) algoritmalarına göre elde edilmiş kophenetik matrisler görülmektedir.

$$\mathbf{D} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 5.8 & 4.2 & 6.9 & 2.6 \\ & 6.7 & 1.7 & 7.2 \\ & & 1.9 & 5.6 \\ & & & 7.6 \end{bmatrix} \end{array} \tag{2.14}$$

$$\mathbf{D}_{Cs} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 4.2 & 4.2 & 4.2 & 2.6 \\ & 1.9 & 1.7 & 4.2 \\ & & 1.9 & 4.2 \\ & & & 4.2 \end{bmatrix} \end{array} \tag{2.15}$$

$$\mathbf{D}_{Cc} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 7.6 & 5.6 & 7.6 & 2.6 \\ & 7.6 & 1.7 & 7.6 \\ & & 7.6 & 5.6 \\ & & & 7.6 \end{bmatrix} \end{array} \tag{2.16}$$

Tek-hat ve tam-hat algoritmalarının her ikisi de bu oluşturulan matrisler için aynı dendrogramlara sahiptirler. Tam-hat $\mathbf{D}_{Cc}$ matrisi mükemmel bir sıradüzensel yapıya sahiptir (http://planetmath.org).

2.8 Kernel Fonksiyonları

4. bölüm'de anlatılan Destek Vektör Makineleri ile kümeleme yönteminde, veri setinde bulunan noktalar önce daha büyük boyutlu bir uzaya taşınmakta ve bu uzayda kümeleme işlemleri yapılmaktadır. Bu işlemler sırasında iç çarpımların hesabına ihtiyaç duyulmaktadır. Bu nedenle, iç çarpımları daha basit hale getirebilmek amacıyla kernel fonksiyonlarından yararlanılmaktadır.

$$K(\mathbf{x},\mathbf{z}) = \langle \Phi(\mathbf{x}) \cdot \Phi(\mathbf{z}) \rangle \tag{2.17}$$

$\Phi(\mathbf{x})$ ifadesi, $\mathbf{x}$ noktalarını daha büyük boyutlu uzaya taşımak için kullanılan dönüşümü ifade etmektedir. Kernel fonksiyonu simetrik olmalıdır. Bu şartı sağlayan kernel fonksiyonlarına literatürde *Mercer Kernel Fonksiyonları* da denilmektedir. Tüm bu koşulları sağlayan kernel fonksiyonları ile yeni kernel fonksiyonları da türetilebilmektedir (Cristianini ve Taylor, 2003). Bu tezde 2.18'de verilen Gaussian (RBF) kernel fonksiyonu kullanılmıştır.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-q\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \tag{2.18}$$

2.9 Gürültü Oranı (SNR)

Veri setlerine eklenecek olan gürültü oranını ifade etmektedir ve denklem 2.19'daki ifade ile belirlenir:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{\sigma_v^2}{\sigma_\eta^2}\right) \tag{2.19}$$

Bu ifadede, σ_v^2 veri setlerinin bileşenlerinin, σ_η^2 ise eklenen gürültünün değişkesidir (variance).

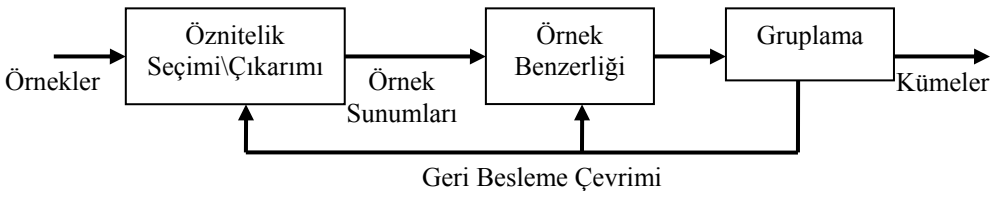
2.10 Kümeleme Problemi

2.10.1 Kümeleme İşleminin Bölümleri

Klasik bir örnek kümeleme işleminde takip edilmesi gereken adımlar şunlardır:

1) Örneklerin sunulması,
2) Örneklerin uzaklık ölçümlerinin veri tabanına uygun olarak tanımlanması,
3) Kümeleme veya gruplama,
4) Veri ayıklama (gerekli olduğu durumlarda yapılmaktadır),
5) Çıkışın değerlendirilmesi (gerekli olduğu durumlarda yapılmaktadır).

Yukarıda bulunan ilk üç adım Şekil 2.7'de görülmektedir. Geri besleme, kümeleme sonucunda elde edilen çıkışın, örnek uzaklık ölçümlerine ve özniteliklerin çıkarılmasına etki etmektedir.



Şekil 2.7 Kümeleme Adımları

Örnek Sunumu, özniteliklerin sayısına, tipine ve kullanılabilecek örnek sayısı ile ilgilidir. Bu bilgilerden bazıları kullanıcı tarafından kontrol edilememektedir.

Öznitelik Seçimi, kümelemede kullanılacak olan özniteliklerden oluşan en etkili alt kümenin belirlenmesi işlemidir. *Öznitelik Çıkarımı*, yeni öznitelikler oluşturabilmek amacı ile giriş özniteliklerinin bir veya daha fazla dönüşümlerinin kullanılmasıdır. Bu iki teknik, kümeleme işlemlerinde en uygun özniteliklerin ortaya çıkarılmasını sağlamaktadır.

Örnek Yakınlıkları, örnek çiftlerine göre tanımlanmış bir uzaklık fonksiyonu ile belirlenmektedir. Bunlar arasında en yaygın olarak kullanılan uzaklık fonksiyonu Öklit fonksiyonudur.

Gruplandırma, birkaç farklı yöntem ile yapılabilmektedir. Çıkış kümelendirmeleri "zor" (verilerin gruplara ayrılması) veya "bulanık" (her bir verinin farklı kümelerde değişken üyelik derecesine sahip olması) olabilir. Sıradüzensel (Hierarchical) kümeleme algoritmaları sıralı bölümler serisinden meydana gelmektedir. Paylaştırmalı (Partitional) kümeleme algoritmaları ise herhangi bir kümeleme kriterini en iyi hale getiren bölümler belirlenmektedir. Bunlar dışında, olasılıksal, çizge tabanlı kümeleme algoritmaları da bulunmaktadır. İlerleyen bölümlerde bu konu daha detaylı biçimde incelenecektir.

Veri Çıkarımı, veri setinin basit ve öz gösteriminin çıkarılması işlemidir (Jain ve diğerleri, 1999).

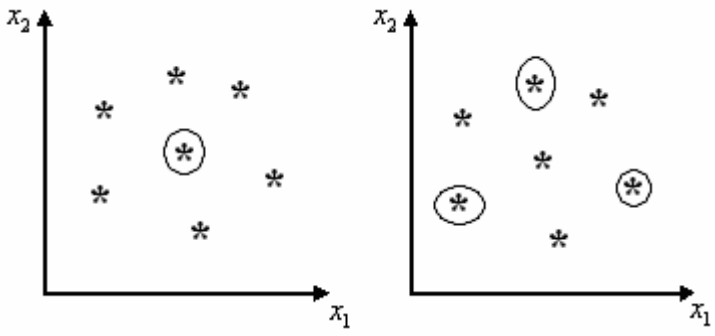
2.10.2 Uzmanın Önemi

Literatürde çok fazla kümeleme algoritması bulunduğundan dolayı, elde bulunan problemin çözümü için gerekli olan algoritmanın seçimi çok zor olmaktadır. Bu sorunu ortadan kaldırmak amacı ile algoritmaları birbirleriyle karşılaştırmada kullanılan çeşitli kriterler belirlenmiştir. Bu kriterler, (i) Kümeleri oluşturma tarzı, (ii) Verilerin yapısı (iii) Kümeleme tekniğinin verilerin yapısı üzerinde herhangi bir etkisi olmayan

değişiklikler karşısındaki hassasiyeti şeklindedir. Farklı yapılardaki veri setlerine uygulanabilecek tek bir kümeleme algoritması bulunmamaktadır. Bunun nedeni algoritmaların kümeleme yaparken izledikleri kriterlerden kaynaklanmaktadır (uzaklık ölçümleri, gruplandırma teknikleri gibi). Herhangi bir kümeleme tekniği kullanılırken, tekniğin işletilmesi dışında, verilerin elde edilme yöntemi ve uzman görüşleri de önemlidir. Kullanıcı ne kadar fazla bilgiye sahip ise kümeleme daha verimli ve etkili olmaktadır (Jain ve Dubes, 1988).

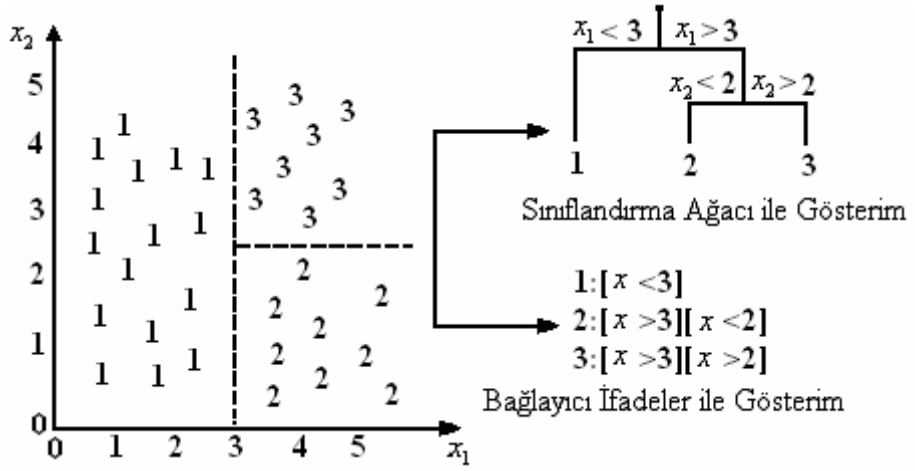
2.10.3 Kümelerin Gösterimi

Kümelerin veya sınıfların belirlenmesi gereken uygulamalarda, veri dizisinin paylaştırılması gerekmektedir. Bu paylaşım, veri noktalarının kümelere ayrılabilirliği hakkında bilgi vermektedir. Bunun yanı sıra, birçok uygulamada sonuç olarak ortaya çıkan kümeler, verilerin elde edilebilmesi amacıyla daha kısa ve öz olarak sunulmalı veya tanımlanmalıdır. Karar verme mekanizmalarında kümelerin gösterimi önemli bir adım olduğu halde araştırmacılar tarafından detaylı olarak incelenmemektedir. Buna göre, kümelerin gösterimi aşağıdaki üç şekilde yapılabilmektedir: (i) Noktaların bulunduğu kümeler, merkezleri ile veya kümede bulunan en dış noktalar ile temsil edilebilmektedir (Şekil 2.8).



Şekil 2.8 Kümelerin Noktalar İle Gösterimi (Merkez (center) ve En Dış Noktalar İle Gösterim)

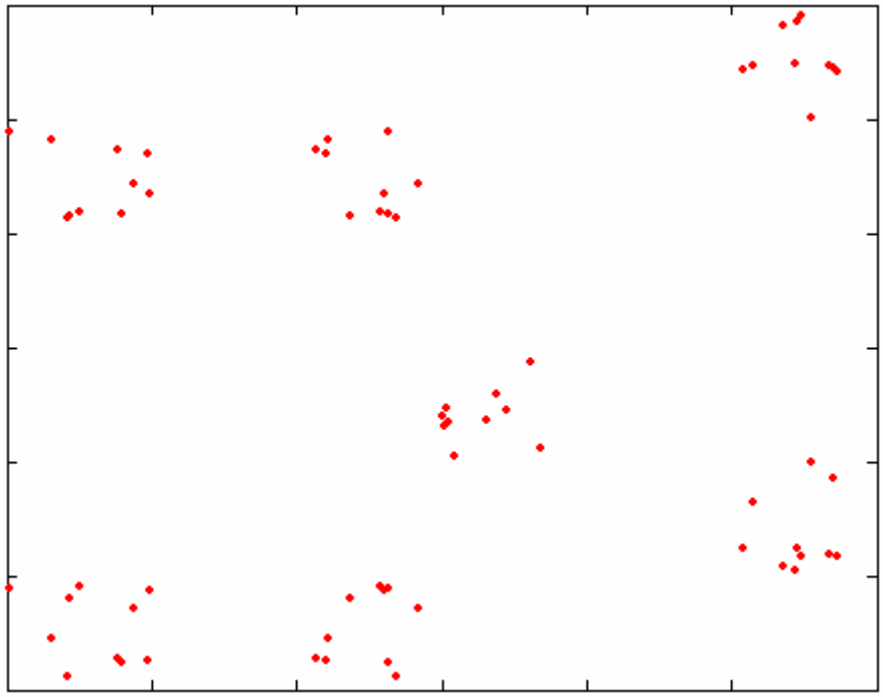
(ii) Kümeler, sınıflandırma ağacında bulunan düğümler ile ifade edilebilmektedir. (iii) Bağlayıcı mantık ifadeleri ile de kümeler temsil edilebilmektedir. Şekil 2.9'da bulunan $[x_1 > 3][x_2 < 2]$ ifadesi, "x_1 büyük 3" ve "x_2 küçük 2" anlamına gelmektedir.



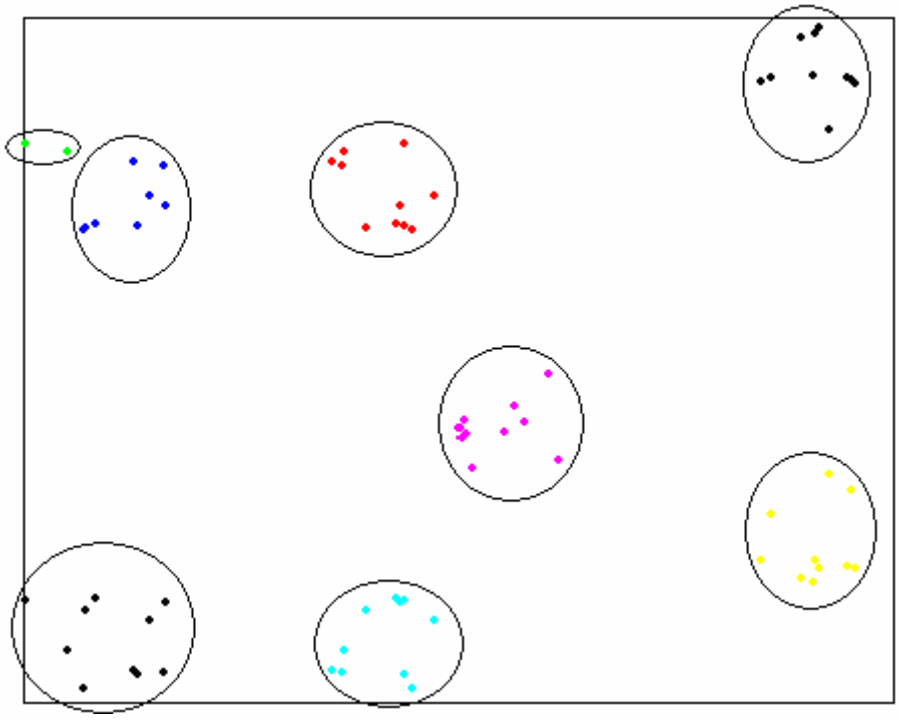
Şekil 2.9 Sınıflandırma Ağacı ve Bağlayıcı İfadeler ile Kümelerin Gösterilmesi

Kümelerin merkezlerine göre temsil edilmesi en çok kullanılan yöntemdir, bu yöntem kümelerin yoğun ve aynı karakteristiğe sahip (izotropik) olmaları durumunda daha verimli olmaktadır. Bununla birlikte, kümelerin farklı karakteristiğe sahip olmaları durumunda bu metot kümeleri tam anlamıyla ifade edememektedir. Böyle bir durumda, kümenin sınır noktaları ile ifadesi daha elverişlidir. Bir kümeyi temsil ederken kullanılan noktaların sayısı, küme şeklinin karmaşıklığının artmasıyla artmaktadır. Şekil 2.9'da gösterilen iki farklı gösterimde birbiriyle eşdeğerdir. Sınıflandırma ağacında, kök düğüm ile yaprak düğüm arasında bulunan her yol bağlayıcı ifadeyi temsil etmektedir.

Bu tez çalışmasında kullanılan veriler, küme sayısının önceden belli olmadığı farklı algoritmalar yardımı ile kümelere ayrılmış ve bu algoritmaların performansları, işlemler esnasında kullanılan bellek, gürültüye dayanıklılık ve işlemler esnasında kullanılan flop sayısı (MATLAB programının işlemleri gerçekleştirirken kullandığı adım sayısı) gibi kriterlere göre incelenmiştir. Verileri kümelere ayırmamızın asıl amacı, daha sonraki aşamalarda kullanılacak olan öğrenebilen sistemlere (yapay sinir ağı veya destek vektörleri makinesi modellerine) verileri daha kolay ve hızlı bir şekilde öğretebilmektir. Dağınık verileri öğrenmektense kümeler halindeki verileri öğrenmek daha hızlı ve verimli olmaktadır. Bu nedenle kümeleme işlemi yapılmaktadır. Şekil 2.10'da dağınık ve Şekil 2.11'de kümelenmiş veriler görülmektedir.



Şekil 2.10 Saçılmış Veriler



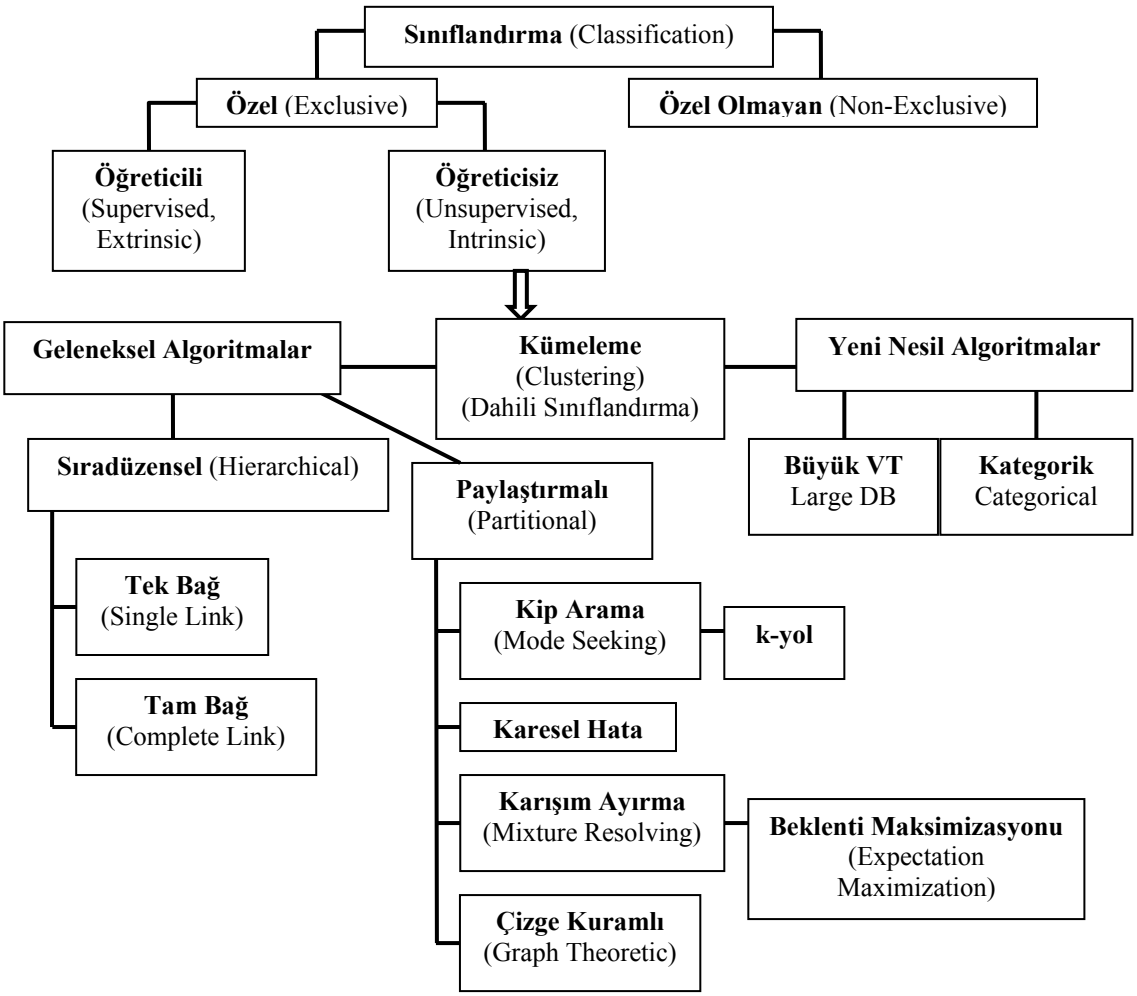
Şekil 2.11 Kümelere Ayrılmış Veriler

ÜÇÜNCÜ BÖLÜM
KÜMELEME YÖNTEMLERİ VE ALGORİTMALARI

3. KÜMELEME YÖNTEMLERİ ve ALGORİTMALARI

3.1. Kümeleme Yöntemleri

Verilerin kümelere ayrılmasında kullanılan çok çeşitli algoritmalar ve yöntemler bulunmaktadır. Bu bölümde en çok kullanılan algoritma çeşitleri ve yöntemleri tanımlanmaktadır. Birinci bölümde kümeleme işlemi, bir çeşit sınıflandırma olarak tanımlanmıştır.



Şekil 3.1 Sınıflandırma ve Kümeleme Çeşitleri

3.1.1 Özel ve Özel-Olmayan Sınıflandırma

Özel Sınıflandırma, nesnelerin bölümlere ayrılmasıdır. Her nesne, tamamen tek bir kümeye veya altkümeye aittir. Özel-Olmayan (örtüşen) Sınıflandırma ise, bir nesneyi birden fazla sınıfa dahil etmektedir. Örneğin, insanları boy veya göz renklerine göre sınıflandırma özel sınıflandırma iken, insanları geçirdikleri hastalıklara göre sınıflandırma özel-olmayan sınıflandırmadır (bir insan birden fazla hastalık geçirmiş olabilir). Bulanık kümeleme, nesnelerin birden fazla kümeye (derecelerine göre) üye olduğu bir çeşit özel-olmayan sınıflandırmadır (Jain ve Dubes, 1988).

3.1.2 Harici ve Dahili Sınıflandırma

Dahili Sınıflandırma da sadece yakınlık matrisi (proximity matrix) kullanılmaktadır. Aynı zamanda, sınıflandırma yapılırken ön bilgiye sahip olunmadığından dolayı, *Öğreticisiz Öğrenme* olarak da adlandırılmaktadır. Harici Sınıflandırma ise, yakınlık matrisi dışında, nesnelerin kategori niteliklerini de kullanmaktadır. Örneğin, sigara kullanan ve kullanmayan insanlardan oluşan bir topluluk göz önünde bulundurulursa, dahili sınıflandırma, bireyleri sağlık durumlarına göre ayırır ve sigara içmenin çeşitli hastalıklara yakalanmadaki etkisini inceler. Harici sınıflandırma ise, sigara içenleri ve içmeyenleri sağlık durumlarına göre sınıflandırır. Verilerin kümelere ayrılması da dahili sınıflandırmadır ve küme analizi konusunun özünü oluşturmaktadır.

3.1.3 Sıradüzensel ve Paylaştırmalı Sınıflandırma

Sıradüzensel sınıflandırma, iç içe sıralanmış bölümlerden oluşmaktadır. Paylaştırmalı Sınıflandırma ise tek bölümden oluşmaktadır. Bu nedenle sıradüzensel sınıflandırma, paylaştırmalı sınıflandırmanın özel bir durumudur. Bu iki konu daha detaylı olarak ilerleyen sayfalarda incelenmektedir.

3.1.4 Toplayıcı ve Bölücü Algoritmalar

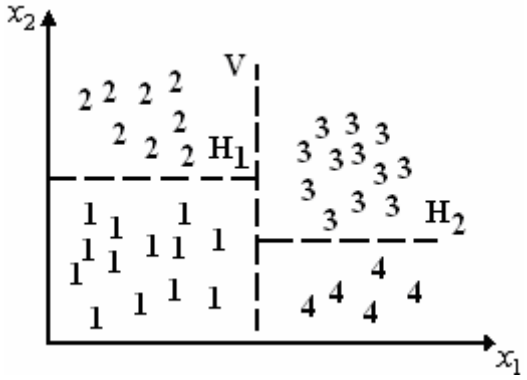
Toplayıcı (Agglomerative) algoritmalarda, başlangıçta her bir veri bir küme olarak kabul edilmektedir ve kademe kademe bu altkümeler, tek bir küme oluşturuluncaya kadar birleştirilir. Bölücü (Divisive) algoritmalarda ise, tüm veriler tek bir küme olarak kabul edilir ve bu küme kademe kademe alt kümelere ayrılır.

3.1.5 Seri ve Eşzamanlı Algoritmalar

Seri (Serial) algoritmalar veriler üzerinde tek tek işlem yapmaktadır. Eşzamanlı (Simultaneous) algoritmalar ise, tüm veriler üzerinde aynı anda işlem yapmaktadır.

3.1.6 Monothetic ve Polythetic Algoritmalar

Bu algoritmalar genellikle, nesnelerin örnekler veya belirli bir uzayda tanımlanmış noktalar olarak tanımlandığı, taksonomi (cinsine göre sınıflandırma) problemlerinde kullanılmaktadır. Monothetic algoritmalar öznitelikleri tek tek kullanmasına rağmen (tek bir karakteristiğe göre gruplandırmaktadır), Polythetic algoritmalar tüm öznitelikleri aynı anda kullanmaktadır (birden fazla karakteristiğe göre gruplandırmaktadır). Algoritmaların çoğu Polythetic'tir bunun nedeni, noktalar arasındaki uzaklıklar hesaplanırken kullanılan özniteliklerin hepsi bu uzaklıklara dayanmaktadır. Şekil 3.2'de noktalar x_1 özniteliği kullanılarak iki gruba ayrılmıştır, dikey V çizgisi ayırma düzlemidir. Bu iki küme daha sonra x_2 özniteliğini kullanarak iki gruba daha ayrılabilmektedir (H_1 ve H_2 yatay çizgileri).



Şekil 3.2 Monothetic Paylaştırmalı Kümeleme

3.1.7 Çizge Kuramı ve Matris Cebri

Bilgisayarda herhangi bir algoritma gerçekleştirilirken, sayısal verimliliğe dikkat edilmelidir. Çizge kuramında, *eksiksizlik (completeness)* ve *bağlanmışlık (connectedness)* gibi terimler göz önünde bulundurulmaktadır. Diğer algoritmalarda ise, karesel hata değeri gibi cebrik ifadeler yer almaktadır. Kümelenecek olan örneklerin düğümlerle ve örnekler arasındaki ilişkilerin kenarlar ile ifade edildiği yapılara çizge denilmektedir. Kenarların ağırlık değerleri örnekler arasındaki uzaklık değerlerini ifade etmektedir. Kümelemenin amacına uygun olarak; aynı kümede bulunan noktalar farklı kümelerde bulunan noktalara göre birbirlerine daha yakındır. Kümeleme metotları, uyuşmayan kenarları belirleyerek ve silerek çizgeleri bağlanmış elemanlar haline getirirler. Her eleman bir kümeyi temsil etmektedir. Çizge kuramına ait daha detaylı bilgi ikinci bölümde yer almaktadır.

3.1.8 Katı ve Bulanık Algoritmalar

Katı (Hard) kümeleme algoritmalarında her veri ayrı bir kümeye ait olmaktadır. Bulanık kümelemede ise bir veri birden fazla kümeye ait olabilmektedir. Bulanık (Fuzzy) kümeleme, noktaların en yüksek üyelik derecesine sahip kümelere yerleştirilmesiyle sert kümelemeye dönüştürülebilmektedir.

3.1.9 Artan ve Artmayan Algoritmalar

Veri setinin çok büyük olduğu, işlem zamanının ve kullanılan hafızanın algoritma üzerinde kısıtlamalara sebep olduğu durumlarda kullanılmaktadır. Kullanılan ilk algoritmalar büyük boyutlu veriler için tasarlanmamıştır, fakat veri madenciliğinin gelişmesi algoritmalarında geliştirilmesine yardımcı olmuştur.

3.2 Sıradüzensel Kümeleme Algoritmaları

Sıradüzensel kümeleme (Hierarchical Clustering), verilerin birbirlerine olan uzaklıklarından oluşturulmuş yakınlık matrisini iç içe sıralı bölümler şekline dönüştüren bir metottur. Sıradüzensel kümeleme algoritması ise, sıradüzensel kümeleme yapabilmek için gerekli olan adımların tanımlanmasıdır. Kümelenecek n adet veri $\mathbf{x}$ vektörü ile gösterilmektedir.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{\mathbf{T}} \tag{3.1}$$

B kümesinin tüm elemanları, L kümesinin bir alt kümesi ise, B kümesi, L kümesinin içine yuvalanmıştır. Aşağıdaki örnekte daha açık şekilde görülmektedir.

$$B = \left\{ \left(x_1, x_3, x_5, x_7\right), \left(x_2, x_4, x_6, x_8\right), \left(x_9, x_{10}\right) \right\} \tag{3.2}$$

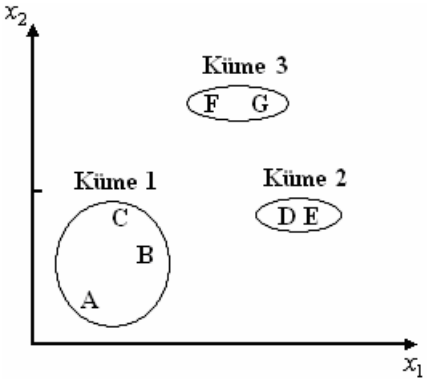
$$L = \left\{ \left(x_1, x_3\right), \left(x_5, x_7\right), \left(x_2\right), \left(x_4, x_6, x_8\right), \left(x_9, x_{10}\right) \right\} \tag{3.3}$$

$$M = \left\{ \left(x_1, x_2, x_3, x_4\right), \left(x_5, x_6, x_7, x_8\right), \left(x_9, x_{10}\right) \right\} \tag{3.4}$$

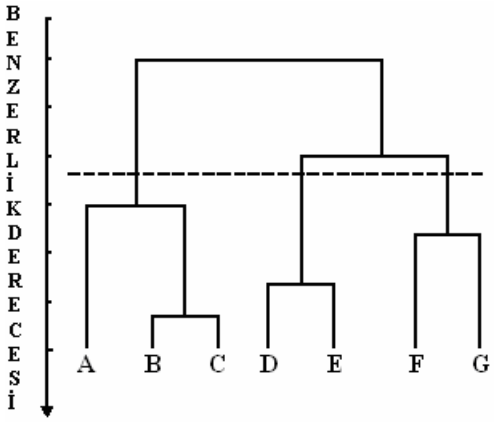
B ve L kümelerinden hiçbiri M kümesinin içine yuvalanmamıştır, benzer şekilde M kümesi de B ve L kümelerinden hiçbirinin içine yuvalanmamıştır. Şekil 3.3'de iki boyutlu bir veri setinin kümelenmesi görülmektedir. Burada A, B, C, D, E, F ve G

noktaları üç ayrı kümeye ayrılmıştır. Bu iç içe sıralı grupları gösterebilmek amacı ile *Dendrogram* olarak adlandırılan gösterim şekli kullanılmaktadır. Sıradüzensel kümelemenin en büyük özelliği, verilerin çok rahat bir şekilde gözlemlenebilmesini sağlayan, Şekil 3.4'deki gibi şekillerin olmasıdır. Dendrogram, sıradüzensel kümelemeye ait özel bir gösterim şeklidir ve kümelerin ifade edildiği, farklı seviyelerde bulunan düğümlerden oluşmaktadır. Düğümleri birleştiren çizgiler ise birbiri içine yuvalanmış kümeleri ifade etmektedir. *Dendrogram* istenilen seviyelerden kesilebilmekte ve o seviyede kaç farklı küme olduğu görülebilmektedir.

Yakınlık çizgesi (proximity graph), her kenarın birbirine olan yakınlıklarına göre belli ağırlıklara sahip olduğu eşik çizgeleridir (threshold graph). Yakınlık matrisine göre çizilen dendrogramlar, yakınlık dendrogramları olarak adlandırılmaktadır ve aynı anda hem kümeleri ve yakınlık değerlerini içermektedir. Yakınlık dendrogramları, yakınlık değerlerinin aralık değerleri veya oransal değerler olduğu durumlarda daha kullanışlıdır.



Şekil 3.3 Noktaların Farklı Kümelere Ayrılması

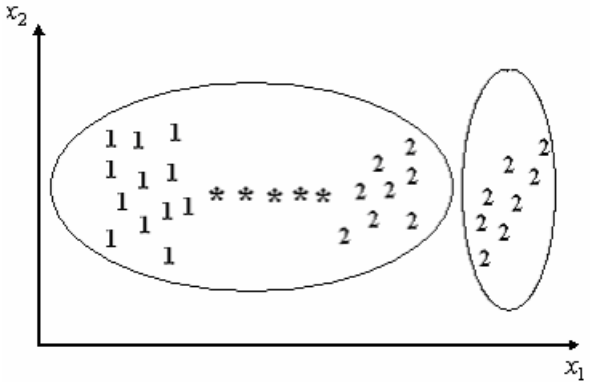


Şekil 3.4 Tek-Bağ Algoritmasına Göre Belirlenmiş Dendrogram

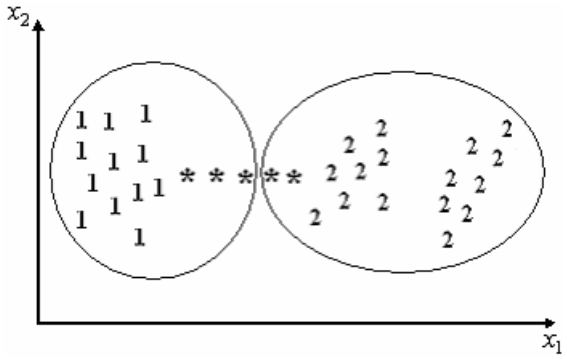
Sıradüzensel kümeleme algoritmalarının çoğu, tek-bağ (single-link), tam-bağ (complete-link) veya minimum-değişke (minimum-variance) algoritmalarından oluşmaktadır. Bunlardan en çok kullanılan algoritmalar tek-bağ ve tam-bağ algoritmalarıdır. Bu iki algoritma, küme çiftleri arasındaki benzerlikleri modelleme yapılarına göre farklılık göstermektedir.

Tek-Bağ algoritmalarında, iki küme arasında bulunan uzaklık, Şekil 3.5'deki gibi tüm veri çiftleri arasındaki uzaklıklardan *en küçük* değerde olanıdır. Tam-Bağ algoritmalarda ise Şekil 3.6'daki gibi, uzaklıklar arasından *en büyük* değerde olanıdır. Her iki durumda da kümeler, en kısa uzaklık kriterine bağlı olarak daha büyük bir küme oluşturabilmek amacıyla birleştirilmektedir.

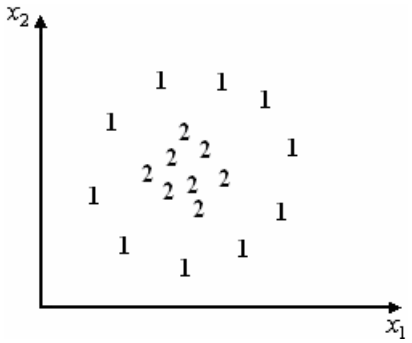
Tam-Bağ algoritmasıyla oluşturulan kümelere Tek-Bağ ile oluşturulan kümelere göre daha kısa ve yoğun olmaktadır. Tek-Bağ algoritması, Tam-Bağ algoritmasına göre çok yönlüdür. Şekil 3.7'de görülen eşmerkezli iki küme Tek-Bağ algoritmaları ile kümelere ayrılabilir fakat Tam-Bağ algoritmaları ile işlem yapılamamaktadır. Bununla birlikte, sebep-sonuç ilişkisine dayalı çalışmalarda Tam-Bağ algoritması daha kullanışlı hiyerarşiler meydana getirmektedir.



Şekil 3.5 Tek-Bağ Kümeleme (1, 2 ve gürültü örnekleri,*)



Şekil 3.6 Tam-Bağ Kümeleme (1, 2 ve gürültü örnekleri, *)



Şekil 3.7 Eşmerkezli İki Küme

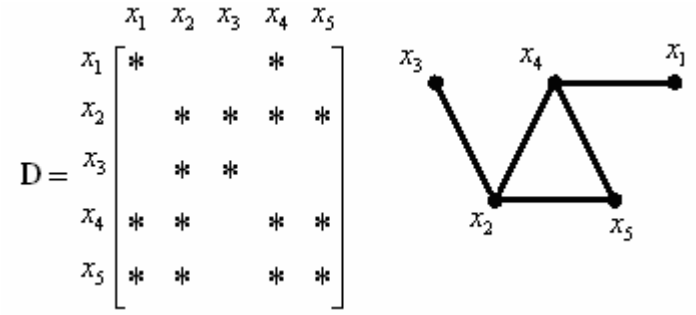
3.2.1 Tek-Bağ, Tam-Bağ ve Grup Ortalama Algoritmaları

Bu algoritmaların uygulanabilmesi için öncelikle, simetrik, nxn boyutlu yakınlık matrisinin elde edilmesi gerekmektedir $\left(\mathbf{D}=\left[d\left(i,j\right)\right]\right)$. Aşağıda örnek bir yakınlık matrisi ve başlangıç çizgesi (eşik çizgesi) gösterilmektedir.

$$\mathbf{D}=\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array}\begin{array}{c} \begin{array}{ccccc} x_1 & x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 0 & 6 & 8 & 2 & 7 \\ 6 & 0 & 1 & 5 & 3 \\ 8 & 1 & 0 & 10 & 9 \\ 2 & 5 & 10 & 0 & 4 \\ 7 & 3 & 9 & 4 & 0 \end{bmatrix} \end{array};\ n=5 \tag{3.5}$$

Başlangıç çizgesi, yönü kesin olarak belli olmayan, ağırlıkları bulunmayan ve iç çevrime sahip olmayan çizgedir ve $\boldsymbol{G}\left(v\right)$ ile ifade edilmektedir (v, farklılık değerini

belirtmektedir). $\boldsymbol{G}(v)$, ikili ilişkileri tanımlamaktadır. Eğer (i,j) değeri, v değerinden düşük ise i ve j arasına kenar (edge) yerleştirilmektedir. Kısaca, $(i,j) \in \boldsymbol{G}(v)$ sadece ve sadece $d(i,j) \le v$ durumu için geçerlidir. Şekil 3.8'de $v = 5$ eşik değeri için yakınlık matrisi ve başlangıç çizgesi görülmektedir.



Şekil 3.8 İkili İlişkiler ve Başlangıç Grafiği

Bu eşik çizgesine bağlı olarak çok farklı algoritmalar türetilmiştir bunlar ilerleyen sayfalarda anlatılmaktadır. Her iki çeşit algoritma da, derece belirten farklılık matrisine dayanarak işlem yapmaktadır ve sonuç olarak, dendrogramlarla ifade edilebilen, iç içe sıralanmış kümeler meydana getirmektedir.

3.2.1.1 Toplayıcı Algoritma (Tek-Bağ Kümeleme)

1) $\boldsymbol{G}(0)$ eşik çizgesi ile belirtilen, her noktayı kendi kümesine yerleştiren ve kenarları bulunmayan ayrık kümeler ile başlanır ve $k = 1$ olarak belirlenir.

2) $\boldsymbol{G}(k)$ eşik çizgesi oluşturulur. Eğer, $\boldsymbol{G}(k)$'nın elemanlarının sayısı (en fazla bağlı alt çizge – maximally connected subgraph) o anda bulunan kümelerin sayısından az ise, $\boldsymbol{G}(k)$'nın her bir elemanını ayrı bir küme olarak belirlenir ve kümeleme işlemini tekrar tanımlanır.

3) Eğer $\boldsymbol{G}(k)$, tekli bağlanmış çizge içeriyorsa işleme son verilir. Değilse, $k \leftarrow k+1$ yapılır ve 2. adıma dönülür.

3.2.1.2 Toplayıcı Algoritma (Tam-Bağ Kümeleme)

1) $\boldsymbol{G}(0)$ eşik çizgesi ile belirtilen, her noktayı kendi kümesine yerleştiren ve kenarları bulunmayan ayrık kümeler ile başlanır ve $k=1$ olarak belirlenir.

2) $\boldsymbol{G}(k)$ eşik çizgesi oluşturulur. $\boldsymbol{G}(k)$'da, o anda bulunan herhangi iki küme küçük bir grup (clique) oluşturuyorsa (en fazla tamamlanmış alt çizge – maximally complete subgraph), bu iki küme tek bir kümede birleştirilerek kümeleme işlemine devam edilir.

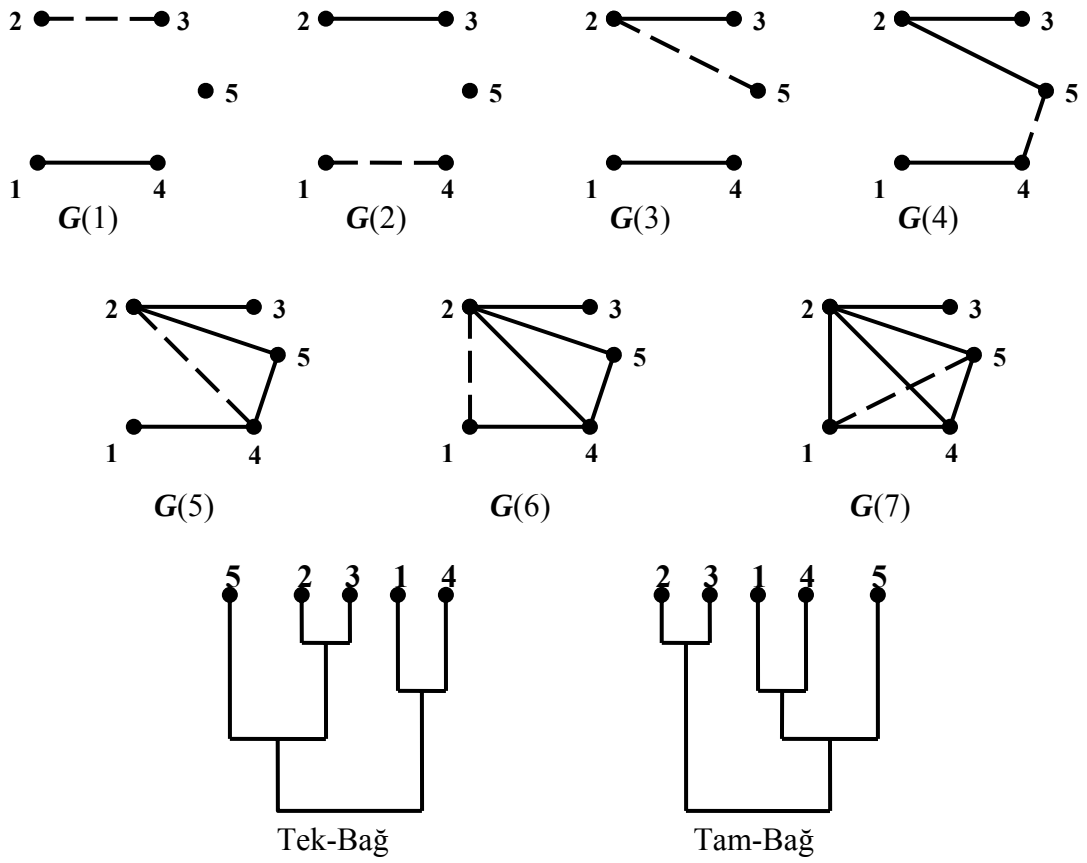
3) Eğer, $k=n(n-1)/2$ ise (böylece $\boldsymbol{G}(k)$, n tane düğüm üzerinde tam çizge oluşturmaktadır), kümeleme işlemine son verilir. Değilse, $k \leftarrow k+1$ yapılır ve 2. adıma dönülür.

Eşik değerlerine göre belirlenen dendrogram (threshold dendrogram), kümeleri yakınlık derecelerinden bağımsız olarak, oluşturuldukları şekilde ifade etmektedir. Yakınlık değerlerine göre belirlenen dendrogram (proximity dendrogram) ise, kümelerin oluşturulduğu seviyeleri ifade etmektedir. Şekil 3.9'da (3.5) ile ifade edilen matristen faydalanılarak oluşturulmuş (eşik değerlerine göre belirlenmiş) tek-bağ ve tam-bağ dendrogram'lar ve çizgeleri görülmektedir. Tek-Bağ kümeleme birleşen alt çizgelere göre, tam-bağ kümeleme ise tam alt çizgelere göre tanımlanmaktadır. Bunun yanı sıra bütün tam alt çizgeler tam-bağ kümeleme oluşturmayabilir.

Tek-Bağ metoduna göre belirlenmiş kümeler, en fazla bağlanmış alt çizgeler olarak tanımlanırken, tam-bağ metoduna göre belirlenmiş kümeler, en fazla tamamlanmış alt çizgeler olarak tanımlanmaktadır. Tek-Bağ kümeler kolayca birleştirilebilir ve genelde düzensizdirler, bu metoda göre iki kümeyi birleştirebilmek amacıyla tek bir kenar koymak yeterlidir. Tam-Bağ kümelerde ise durum daha farklıdır, tam-bağ kümeler kolayca birleştirilemezler ve *tamlık* özelliği *bağlanmışlık* özelliğine nazaran daha ön plandadır.

Bir eşik çizgesinde bulunan birbirine bağlı her alt çizge tek-bağ küme oluşturur fakat her grup tam-bağ küme oluşturamaz. Ayrıca, tek-bağ metodu, *minimum* metodu, tam-bağ metodu ise *maksimum* metodu olarak da adlandırılmaktadır. Bununla birlikte, tam-

bağ metodu *çap* metodu olarak da tanımlanır çünkü, tam bir alt çizgenin çapı alt çizgede bulunan tüm yakınlık değerleri arasında en büyük yakınlık değerine sahip olmasıdır (Jain ve Dubes, 1988).



Şekil 3.9 Eşik Çizgeleri ve Sıradüzensel Kümeleme İçin Dendrogramlar

Şekil 3.9'da, ilk dört çizge tüm tek-bağ işlemlerini içermektedir. Bununla birlikte, tam-bağ gösterimini yapabilmek için yedi tane çizge çizilmiştir. Şekil 3.9'da $\{x_2, x_5, x_4\}$ noktaları bir grup (clique) oluşturmaktadır fakat bu üç nokta tam-bağ kümesi değildir. $\{x_2, x_3\}$ ve $\{x_1, x_4\}$ tam-bağ kümeleri bulunduktan sonra, $\{x_5\}$ noktası bu bulunan iki kümeden birine birleşmek zorundadır, kümeler bir kere oluşturulduktan sonra tekrar bozulamamaktadır. Dendrogramlar ise, farklı seviyelerdeki kümelemelerden ortaya çıkarılmaktadır. Şekil 3.9'da bulunan tek-bağ kümelemeyi gösteren dendrogram $\boldsymbol{G}(3)$ çizgesinden, tam-bağ kümelemeyi gösteren dendrogram da $\boldsymbol{G}(7)$ çizgesinden

çizilmiştir. Burada karşımıza, $\{x_5\}$ noktasının hangi kümeye dahil edilmesi gerektiği ile ilgili bazı sorular çıkmaktadır.

Grup Ortalama (Average-Link) Metodu bir küme içerisinde çiftler arasındaki ortalama değerlere bağlıdır, yani tek-bağ veya tam-bağ algoritmalarında olduğu gibi en fazla veya en az benzerliklere bağlı değildir. Bir küme içerisindeki tüm nesneler küme içi benzerliğe katkıda bulunduğu için ortalama olarak kendi kümesi içerisindeki eleman gibidir, diğer kümelerdeki elemanlar gibi değildir. Eğer iki kümenin elemanları arasındaki karşılıklı uzaklıkların ortalaması eşik uzaklığından az ise, kümeler birleştirilir.

3.2.2 Çizge Kuramı Algoritmaları

Tek-Bağ metoduna göre kümeleme yapılırken, $\boldsymbol{G}(\infty)$ için "En Küçük Tarama Ağacı (Minimum Spanning Tree - MST)" ile işlemlere başlanır. Aşağıda tek-bağ algoritması görülmektedir:

1) Ayrık kümeler ile işleme başlanır (her veri ayrı bir küme olarak kabul edilmektedir). $\boldsymbol{G}(\infty)$ üzerinde bir MST bulunur. Tüm veriler tek bir kümeye dahil oluncaya kadar 2. ve 3. adımlara devam edilir.

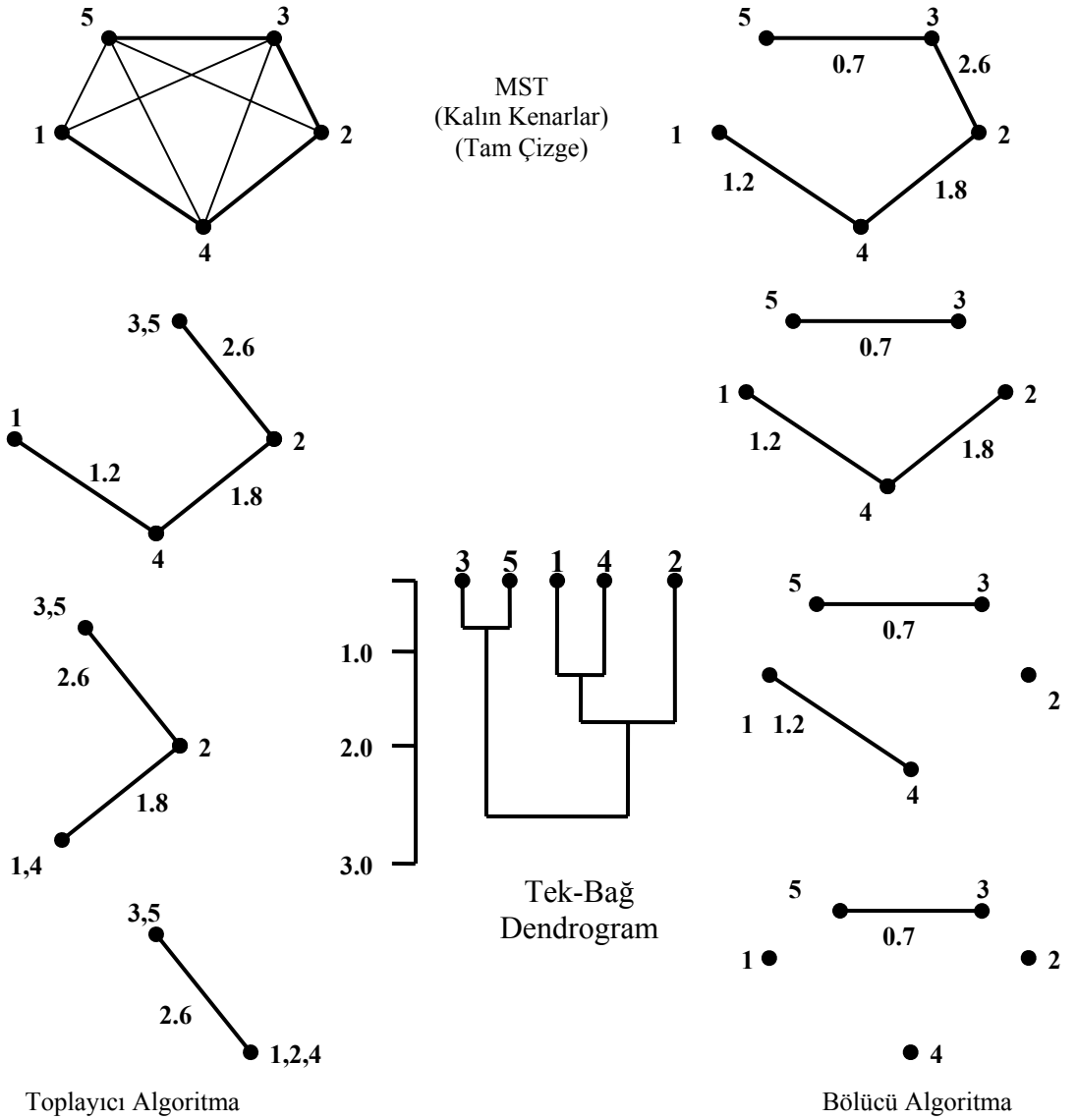
2) Sonraki kümeyi oluşturabilmek için, en küçük ağırlığa sahip kenar MST ile birleştirilir.

3) 2. adımda seçilen kenarın ağırlığı, en büyük yakınlık değerinden daha büyük bir değer ile değiştirilir.

Bu açıklamalar doğrultusunda, tek-bağ kümeleme için bölücü algoritmayı da basitçe tanımlamak mümkündür. MST'de bulunan kenarlar ağırlıklarına göre kesilir (en büyük ağırlıktan başlayarak). Her kesim yeni bir kümenin oluşmasını sağlamaktadır. Bu algoritmalar daha önce tanımlanmış olan tek-bağ algoritmaları ile aynı kümeleri oluşturmaktadır (yakınlıklar arasında herhangi bir bağ olmadığı müddetçe). Bu algoritmaya ait bir örnek Şekil 3.10'da görülmektedir. $\mathbf{D}$ yakınlık matrisini ifade etmektedir. "Düğüm Boyama" olarak adlandırılan yöntem, n tane düğüme farklı

renklerin atanmasıdır. $\mathbf{G}(v)$'de bulunan ve aynı kenara bağlı iki düğüm aynı renge boyanmaktadır. Düğüm boyama ve tam-bağ kümeleme metodu arasındaki ilişki, MST ile tek-bağ kümeleme metodu arasındaki ilişki kadar basit değildir (Zahn, 1970).

$$\mathbf{D} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccc} x_2 & x_3 & x_4 & x_5 \end{array} \\ \begin{bmatrix} 2.3 & 3.4 & 1.2 & 3.7 \\ & 2.6 & 1.8 & 4.6 \\ & & 4.2 & 0.7 \\ & & & 4.4 \end{bmatrix} \end{array} \tag{3.6}$$



Şekil 3.10 MST Prensibine Göre, Tek-Bağ Kümeleme Metodu İçin Toplayıcı ve Bölücü Algoritmaların Uygulanması

3.2.3 Matris Güncelleme Algoritmaları

Bu algoritma, tek-bağ ve tam-bağ kümeleme metotlarında kullanılan yakınlık matrisinin güncellenmesi prensibine dayanmaktadır. Algoritma, toplayıcı karakteristik göstermektedir, eski kümeler birleştirilirken, yakınlık matrisinin satır ve sütunları silinmektedir. İşlemleri basitleştirmek için yakınlık matrisinde herhangi bir bağın (tie) olmadığı varsayılmaktadır. Şekil 3.11'de bu algoritmanın (3.6) ile verilen yakınlık matrisine uygulanması görülmektedir. $\mathbf{D}=\left[d(i,j)\right]_{n\ \mathrm{x}\ n}$ yakınlık matrisini, $\boldsymbol{L}(k)$, k. kümelemenin seviyesini belirtmektedir. Kümeler 0, 1…, $(n-1)$ sayıları ile, sıra numarası m olan küme ise (m) ile ve son olarak (r) ve (s) gibi iki küme arasındaki uzaklık da $d\left[(r),(s)\right]$ ile ifade edilmektedir.

3.2.3.1 Johnson Algoritması

1) Başlangıçta her nokta ayrı küme olarak kabul edilir. $L(0)=0$ ve sıra numarası $m=0$ seçilerek başlanır.

2) O andaki kümelemede birbirine en benzer küme çifti bulunur ve $d\left[(r),(s)\right]=\min d\left[(i),(j)\right]$ eşitliğine göre (minimum, o andaki kümeleme esnasında tüm küme çiftlerinin incelenmesiyle seçilir), $\left\{(r),(s)\right\}$ kümesi olarak adlandırılır.

3) Sıra numarası arttırılır ($m\leftarrow m+1$). (r) ve (s) kümeleri, bir sonraki kümelemeyi (m) elde edebilmek amacıyla tek bir kümede birleştirilir. Kümelemenin seviyesi $\boldsymbol{L}(m)=d\left[(r),(s)\right]$ olarak belirlenir.

4) Yakınlık matrisi, (r) ve (s) kümelerinin bulunduğu satır ve sütunlar silinerek ve yeni oluşturulan kümeye uygun satır ve sütunlar eklenerek güncellenir. Yeni oluşturulan (r,s) kümesi ile eski (k) kümesi arasındaki yakınlık değeri tek-bağ ve tam-bağ kümeleme için farklı şekillerde bulunmaktadır.

Tek-Bağ için, $d\left[(k),(r,s)\right]=\min\left\{d\left[(k),(r)\right],d\left[(k),(s)\right]\right\}$ (3.7)

Tam-Bağ için, $d\left[(k),(r,s)\right]=\max\left\{d\left[(k),(r)\right],d\left[(k),(s)\right]\right\}$ (3.8)

5) Tüm nesneler tek bir kümede toplanmış ise işlem tamamlanır. Toplanmamışsa 2. adıma geri dönülür.

$$\mathbf{D}=\begin{array}{c} \\ 1\\2\\3\\4\\5\end{array}\begin{array}{c}\begin{array}{ccccc}1&2&3&4&5\end{array}\\\begin{bmatrix}0&2.3&3.4&1.2&3.7\\&0&2.6&1.8&4.6\\&&0&4.2&0.7\\&&&0&4.4\\&&&&0\end{bmatrix}\end{array}$$

$$\begin{array}{c} \\ 1\\2\\3,5\\4\end{array}\begin{array}{c}\begin{array}{cccc}1&2&3,5&4\end{array}\\\begin{bmatrix}0&2.3&3.4&1.2\\&0&2.6&1.8\\&&0&4.2\\&&&0\end{bmatrix}\end{array}$$

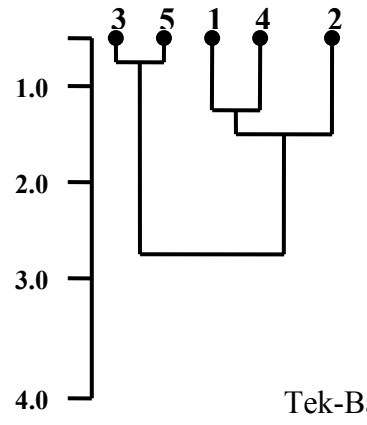
$$\begin{array}{c} \\ 1,4\\2\\3,5\end{array}\begin{array}{c}\begin{array}{ccc}1,4&2&3,5\end{array}\\\begin{bmatrix}0&2.3&3.4\\&0&2.6\\&&0\end{bmatrix}\end{array}$$

$$\begin{array}{c} \\ 1,2,4\\3,5\end{array}\begin{array}{c}\begin{array}{cc}1,2,4&3,5\end{array}\\\begin{bmatrix}0&2.6\\&0\end{bmatrix}\end{array}$$

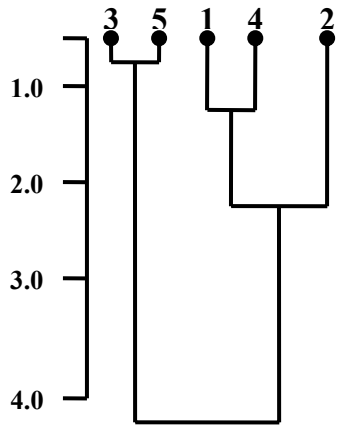
$$\begin{array}{c} \\ 1\\2\\3,5\\4\end{array}\begin{array}{c}\begin{array}{cccc}1&2&3,5&4\end{array}\\\begin{bmatrix}0&2.3&3.7&1.2\\&0&4.6&1.8\\&&0&4.2\\&&&0\end{bmatrix}\end{array}$$

$$\begin{array}{c} \\ 1,4\\2\\3,5\end{array}\begin{array}{c}\begin{array}{ccc}1,4&2&3,5\end{array}\\\begin{bmatrix}0&2.3&4.4\\&0&4.6\\&&0\end{bmatrix}\end{array}$$

$$\begin{array}{c} \\ 1,2,4\\3,5\end{array}\begin{array}{c}\begin{array}{cc}1,2,4&3,5\end{array}\\\begin{bmatrix}0&4.6\\&0\end{bmatrix}\end{array}$$



Tek-Bağ



Tam-Bağ

Şekil 3.11 Tek-Bağ ve Tam-Bağ Metotları İçin Matris Güncelleme Algoritmasının Kullanımı

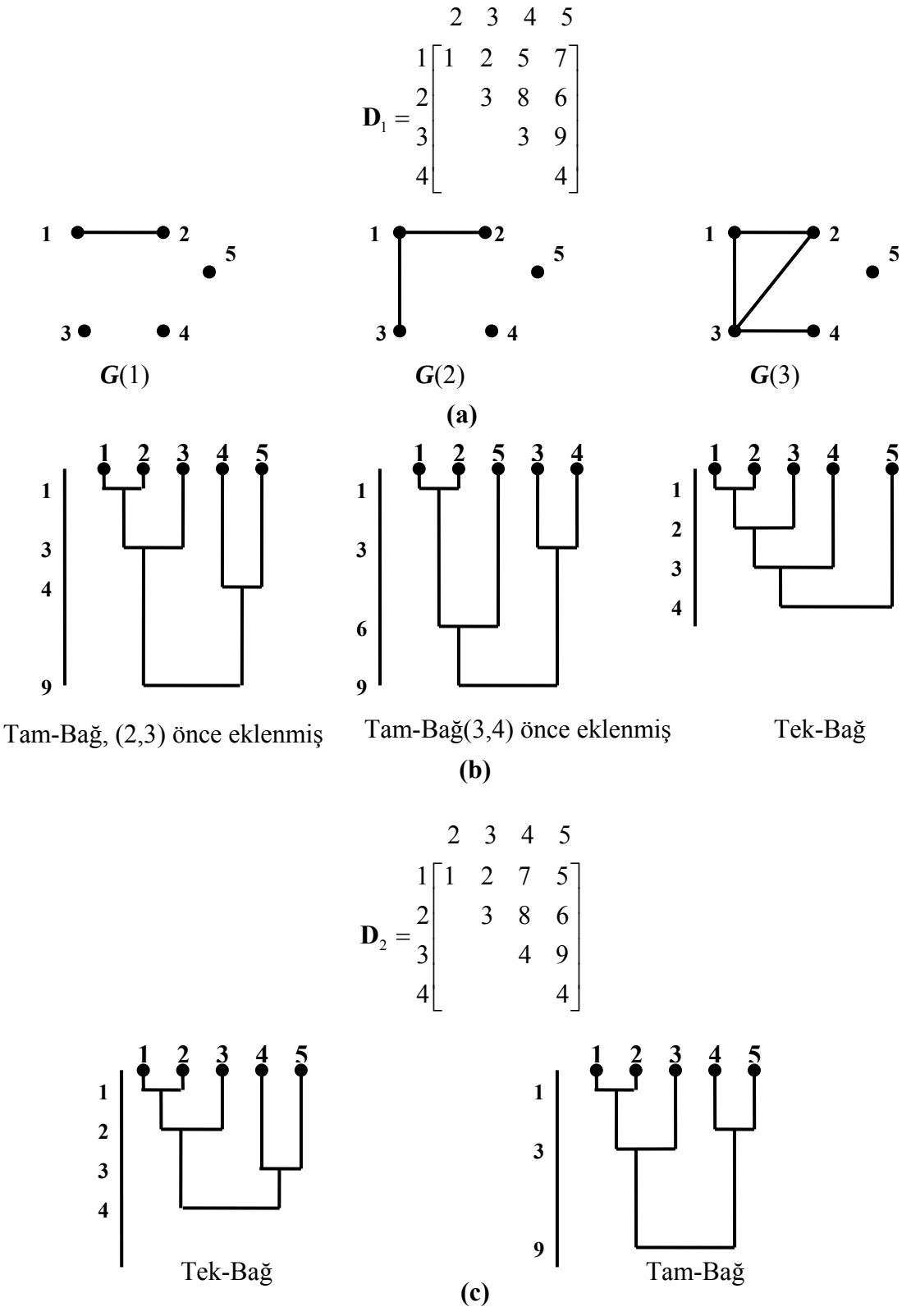
Yukarıda tanımlanan algoritma için, *Depolanmış Matris*, *Sıralanmış Matris* ve *Sıralanmış Veri* olmak üzere, üç farklı yaklaşım kullanılmıştır. Bu üç yöntem birbirinden, yakınlık matrisinin rasgele erişimli hafızada veya disk gibi yardımcı hafıza elemanlarında saklanmasına göre değişmektedir.

3.2.4 Yakınlık Matrisinde Bulunan Bağlar

Tek-Bağ ve Tam-Bağ metotları arasında seçim yapmak, bu metotlarda kullanılacak olan algoritmaların seçiminden çok daha zor olmaktadır. Bu bölümde, bu iki metot hakkında, özellikle de yakınlık matrisinde bulunan bağların etkisi hakkında, bilgi verilecektir. *Bağ* kelimesi aynı yakınlık değerine sahip veriler için kullanılmaktadır.

Tek-Bağ ve Tam-Bağ metotları, güncellemenin yapılış şekli, çizge yapılarının tanımlanması gibi birçok yönden birbirlerinden farklılık göstermektedir. Bu iki metot, yakınlık matrisinin, ikinci bölümde anlatılan ultrametrik eşitsizliği sağladığı durumlarda, sonuç olarak aynı kümeleri vermektedir. Bu iki metot, yakınlık matrisinde bulunan bağlar nedeniyle birbirinden farklılık göstermektedir. Şimdiye kadar gösterilen örneklerde bulunan yakınlık matrislerinin herhangi bir bağ içermediği varsayılmıştır. Böylece, aynı seviyede iki yeni küme oluşmamaktadır ve şimdiye kadar tanımlanan algoritmalar birbirinden farklı kümeler oluşturmuştur.

Bağ, yakınlık çizgesine bir kerede iki veya daha fazla kenarın eklenmesini ifade etmektedir. Tek-Bağ metodu süreklilik özelliğine sahip olduğundan, yakınlık matrisindeki bağlardan kaynaklanan belirsizliklerden etkilenmemektedir. Eğer yakınlık matrisinde bulunan bağlar, çok küçük miktardaki yakınlık değerlerinin çıkarılmasıyla veya eklenmesiyle bozulursa, sonuç olarak ortaya çıkan tek-bağ dendrogramlar, eklenen miktar sıfıra yaklaşırken, bağların bozulma şeklinden bağımsız olarak, aynı dendrograma hiçbir problem çıkmadan birleşmektedir (Jain ve Dubes, 1988).



Şekil 3.12 Yakınlık Matrisinde Bulunan Bağların Tek-Bağ ve Tam-Bağ Kümeleme Üzerindeki Etkisi, (a) Eşik Çizgeleri, (b) Yakınlık Dendrogramları, (c) Değiştirilmiş Yakınlık Matrisi ve Dendrogramlar

Şekil 3.12'de görüldüğü gibi, (a) gösteriminin $\boldsymbol{G}(3)$ aşamasında iki kenar aynı anda eklenmiştir. Tek-Bağ sıradüzeni içinde aynı durum söz konusudur. (b) gösteriminde bulunan ve tek-bağ metoduna göre oluşturulmuş dendrogram da aynı seviyede birden fazla küme oluşturulabilmesine rağmen benzersizdir. MST veya matris güncelleme algoritmaları da aynı sonuçları vermektedir. Tam-Bağ metodu için durum çok farklıdır, (b) gösterimindeki sıradüzenler, $(2,3)$'ün ilk olarak eklenmesi ve $(3,4)$'ün ilk olarak eklenmesi ile iki farklı şekilde oluşturulmuştur. Bu iki kümeleme yapısı çok farklıdır. $(2,3)$ ve $(3,4)$ kenarlarının aynı zamanda eklenmesiyle problem çözülememektedir. Çünkü elde edilen dört kenarlı çizge, tam çizge değildir. Aslında, bir sonraki tam çizge tüm beş düğümü de kapsamaktadır ve sıradüzensel kümeleme sadece üç seviyede gerçekleşecektir. Şekil 3.12 (c) gösterimi ise bağların önemini göstermektedir. (a) gösterimindeki yakınlık matrisi $(3,4)$ ve $(4,5)$ arasındaki değerlerin yer değiştirilmesiyle yeniden yazılmıştır. Bu durumda, benzersiz bir tam-bağ yapısı elde edilmiştir çünkü aynı yakınlık değerine sahip iki kenar rasgele eklenebilmektedir. (c)'de bulunan tam-bağ ve tek-bağ dendrogramları (b)'de bulunan dendrogramlara göre birbirine daha çok benzemektedir. Sıradüzensel yapı, yakınlıklardaki çok küçük değişimlerden büyük oranda etkilenmektedir. Yakınlık matrisinde bulunan benzerlikler arttıkça problem daha karmaşık bir yapıya dönüşmektedir. Tam-Bağ metodu yakınlıklar arasındaki bağların artmasıyla daha da karmaşıklaşsa da, tek-bağ metoduna göre birçok uygulamada daha kullanışlıdır.

3.2.5 Genelleştirilmiş Matris Güncelleme Algoritmaları

Bu bölümde, bölüm 3.2.2'de anlatılan matris güncelleme algoritmalarının genelleştirilmiş şekli ve bu algoritmaların uygulamaları anlatılmaktadır. Matris güncelleme algoritmalarındaki dördüncü adımın genelleştirilmesiyle elde edilen bu algoritma SAHN (Sequential (Sıralı), Agglomerative (Toplayıcı), Hierarchical (Sıradüzensel), Nonoverlapping (Örtüşmeyen)) olarak adlandırılmaktadır. Bölüm 3.2.2'de bulunan dördüncü adımda, yeni oluşturulan küme (r,s) ile n_k elemanlı eski küme (k) arasında bulunan farklılığın nasıl güncelleneceği anlatılmıştır. Tek-Bağ ve

tam-bağ metotları sırasıyla $\{(k),(r)\}$ ve $\{(k),(s)\}$ çiftleri arasındaki minimum ve maksimum yakınlık değerlerini kullanmıştır. Bu adımın genel şekli aşağıda görülmektedir.

$$\begin{aligned} & d\left[(k),(r,s)\right] \\ & = \alpha_r, d\left[(k),(r)\right] + \alpha_s, d\left[(k),(s)\right] + \beta d\left[(r),(s)\right] + \gamma \left| d\left[(k),(r)\right] - d\left[(k),(s)\right] \right| \end{aligned} \tag{3.9}$$

Bu formül Lance ve Williams (1967) tarafından bulunmuştur. Çizelge 3.1'de sık kullanılan algoritmalar için bu formülün parametre değerlerleri görülmektedir. Çizelge 3.1'deki "PGM (Pair Group Method)" çift grup metodu, "U (Unweighted)" ağırlıksız ve "W (Weighted)" ağırlıklı anlamına gelmektedir. Ağırlıksız metotta, dendrogramların yapılarından bağımsız olarak kümelerde bulunan nesnelere eşit işlemler yapılır. Ağırlıklı metot ise, tüm kümelere eşit ağırlık değeri vermektedir, böylece küçük kümelerde bulunan nesneler büyük kümelerde bulunan nesneler göre daha ağırlıklı hale gelmiş olur. Çizelge 3.1'de bulunan "A (Arithmetic Average)" ve "C (Centroid)" harfleri sırasıyla, aritmetik ortalama ve kitle merkezini ifade etmektedir. Bu açıklamalar ışığında, "UPGMA (Unweighted Pair Group Method Using Artihmetic Averages)" aritmetik ortalamaları kullanan ağırlıksız çift grup metodu anlamına gelmektedir. Ayrıca, UPGMC metodu aynı zamanda kitle merkezi (centroid) metodu olarak, WPGMC metodu da kenarortay metodu olarak adlandırılmaktadır.

Aritmetik ortalamanın bulunması tek-bağ ve tam-bağ metotlarının dezavantajlarını ortadan kaldırmaktadır. Varolan küme ile oluşması beklenen küme arasındaki farklılıkların ölçülmesinde tek-bağ metodu her iki kümede bulunan birbirine en yakın nesneleri bulurken, tam-bağ metodu en uzak nesneyi bulmaktadır, UPGMA ve WPGMA metotları ise aritmetik ortalamaları kullanmaktadır. Kitle merkezi metotları ise (UPGMC ve WPGMC) iki küme arasındaki farklılıkları kitle merkezlerinin uzaklıklarına göre belirlemektedir. UPGMC metodunda uzaklıklar, her kümede bulunan nesnelerin uzaklıklarının kitle merkezleri cinsinden hesaplanmasıyla elde edilmektedir, WPGMC metodunda ise uzaklıklar, yeni bir küme oluşturmak için bir araya gelmiş iki kümenin kitle merkezlerinin hesaplanmasıyla elde edilmektedir. Kitle merkezi metotları

sadece, nesnelerin yakınlık ölçümlerinin öklit uzaklığının karesi ile ifade edildiği durumlarda kullanılmaktadır.

Çizelge 3.1 SAHN Matris Güncelleme Algoritması İçin Farklı Parametre Değerleri

Kümeleme Metotları	α_r	α_s	β	γ
Tek-Bağ	$1/2$	$1/2$	0	$-1/2$
Tam-Bağ	$1/2$	$1/2$	0	$1/2$
UPGMA (grup ortalama)	$\frac{n_r}{n_r+n_s}$	$\frac{n_s}{n_s+n_r}$	0	0
WPGMA (ağırlıklı ortalama)	$1/2$	$1/2$	0	$1/2$
UPGMC (ağırlıksız kitle merkezi)	$\frac{n_r}{n_r+n_s}$	$\frac{n_s}{n_s+n_r}$	$\frac{-n_r n_s}{(n_s+n_r)^2}$	0
WPGMC (ağırlıklı kitle merkezi)	$1/2$	$1/2$	$-1/4$	$1/2$
Ward Metodu (minimum değişme)	$\frac{n_r+n_k}{n_r+n_s+n_k}$	$\frac{n_s+n_k}{n_r+n_s+n_k}$	$\frac{-n_k}{n_r+n_s+n_k}$	0

Şekil 3.13'de Çizelge 3.1'de bulunan matris güncelleme algoritmaları ile elde edilmiş dendrogramlar görülmektedir.

Aşağıda üç boyutlu uzayda tanımlanmış altı tane vektör bulunmaktadır.

$$\begin{aligned} \mathbf{x}_1 &= \begin{bmatrix}1 & 2 & 2\end{bmatrix}^T & \mathbf{x}_4 &= \begin{bmatrix}3 & 4 & 3\end{bmatrix}^T \\ \mathbf{x}_2 &= \begin{bmatrix}2 & 1 & 2\end{bmatrix}^T & \mathbf{x}_5 &= \begin{bmatrix}0 & 3.5 & 3.5\end{bmatrix}^T \\ \mathbf{x}_3 &= \begin{bmatrix}0 & 1 & 3\end{bmatrix}^T & \mathbf{x}_6 &= \begin{bmatrix}2 & 2.5 & 2.5\end{bmatrix}^T \end{aligned} \tag{3.10}$$

Bu vektörler arasında bulunan uzaklıklar, karesel öklit fonksiyonu ile hesaplanarak aşağıdaki matris elde edilmiştir.

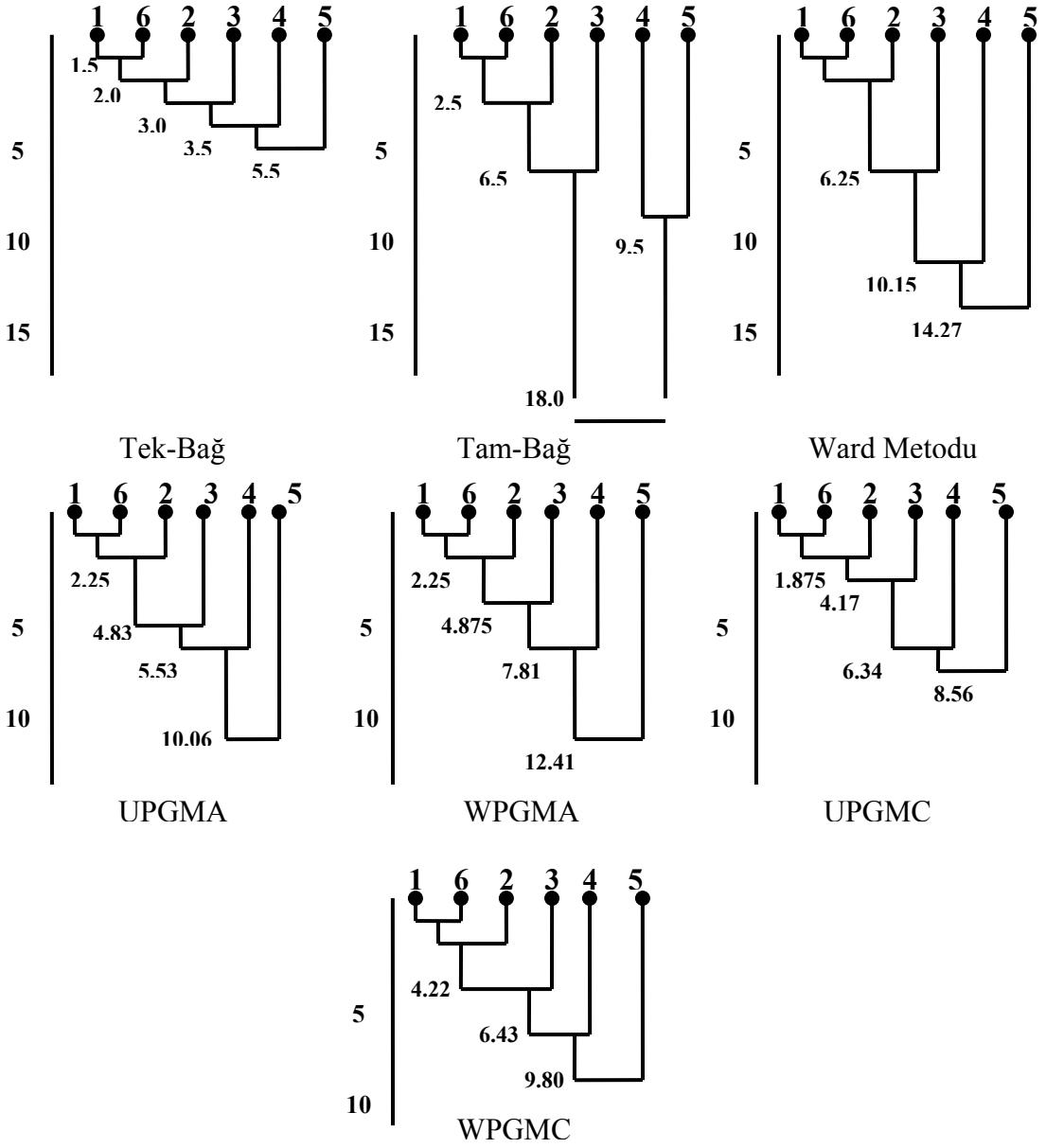
$$d(\mathbf{x},\mathbf{y}) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2} \tag{3.11}$$

$$\mathbf{D} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 2 & 3 & 9 & 5.5 & 1.5 \\ & 0 & 5 & 11 & 12.5 & 2.5 \\ & & 0 & 18 & 6.5 & 6.5 \\ & & & 0 & 9.5 & 3.5 \\ & & & & 0 & 6 \\ & & & & & 0 \end{bmatrix} \end{array} \tag{3.12}$$

$(\mathbf{x}_3, \mathbf{x}_5)$ ve $(\mathbf{x}_3, \mathbf{x}_6)$ çiftleri arasında bulunan benzerlik (bağ), dendrogramların oluşturulmasında herhangi bir belirsizliğe sebep olmamaktadır. Çizelge 3.1'de bulunan algoritmalardan Ward metodu ile iki aritmetik metot ve iki kitle merkezli metot arasında, kümelerin oluşma seviyesi dışında, yapı bakımından bir fark bulunmamaktadır. Bu algoritmalar arasında en iyi performansı Ward metodu sağlamaktadır. Bu metot karesel hata prensibine göre işlem yapmaktadır, bu prensibi kullanan paylaştırmalı kümeleme metotları da bulunmaktadır. d boyutlu bir uzayda bulunan n elemanlı bir küme için küme merkezi aşağıdaki formül ile elde edilmektedir:

$$m_j^{(k)} = (1/n_k) \sum_{i=1}^{n_k} x_{ij}^{(k)} \tag{3.13}$$

$x_{ij}^{(k)}$, k kümesinde bulunan, i. örneğin j. elemanının değerini ifade etmektedir. k kümesinin karesel hatası her elemanın merkeze olan uzaklıklarının karelerinin toplamı ile elde edilmektedir.



Şekil 3.13 Matris Güncelleme Algoritmaları İçin Oluşturulan Dendrogramlar

$$e_k^2 = \sum_{i=1}^{n_k} \sum_{j=1}^{d} \left[x_{ij}^{(k)} - m_j^{(k)} \right]^2 \tag{3.14}$$

K tane kümenin bulunduğu tüm kümeleme işlemi için karesel hatanın değeri her bir kümenin karesel hatasının toplamı ile elde edilmektedir.

$$E_K^2 = \sum_{k=1}^{K} e_k^2 \tag{3.15}$$

Ward metodu ΔE_{pq}^2'yi en aza indirgeyen küme çiftlerini bir araya getirmektedir, E_K^2'deki değişimin sebebi p ve q kümelerinin bir sonraki kümelemede t kümesini oluşturabilmek amacıyla birleştirilmesinden kaynaklanmaktadır. Tüm kümelerin karesel hatası, üç küme hariç, değişmemektedir.

$$\Delta E_{pq}^2 = e_t^2 - e_p^2 - e_q^2 \tag{3.16}$$

Değerlerin yerine konulmasıyla, karesel hatadaki değişmenin sebebinin sadece kitle merkezlerine bağlı olduğu görülmektedir.

$$\Delta E_{pq}^2 = \frac{n_p n_q}{n_p + n_q} \sum_{j=1}^{d} \left[m_j^{(p)} - m_j^{(q)} \right]^2 \tag{3.17}$$

Birleştirilmek için seçilen p ve q kümeleri (3.18) eşitliğini en aza indirgemektedir. küme sayısı azaldıkça karesel hata değeri artmaktadır fakat bu artış Ward metodunda olabildiğince küçüktür. p ve q kümeleri, t kümesi için birleştiğinde diğer tüm kümeler ile olan yakınlıklar ile yeni oluşturulan t kümesinin yakınlık değeri güncellenmelidir. t, q, ve p kümeleri dışında herhangi bir r kümesi için $d\left[(r),(t)\right]$ değeri aşağıdaki formül ile elde edilmektedir.

$$d\left[(r),(t)\right] = \frac{n_r + n_p}{n_r + n_t} d\left[(r),(p)\right] + \frac{n_r + n_q}{n_r + n_t} d\left[(r),(q)\right] - \frac{n_r}{n_r + n_t} d\left[(p),(q)\right] \tag{3.18}$$

Uygulamalar için en uygun sıradüzensel kümeleme algoritmasının seçimi önemli bir problem olmaktadır. Karesel hata kuramı mühendislik uygulamalarında sıkça kullanılmaktadır, böylece Ward metodu gibi karesel hata değerini en küçük yapan herhangi bir algoritmanın kullanılması daha uygun olmaktadır. Bununla birlikte, küme analizinin asıl amacı verinin yapısını araştırmaktır, bu nedenle karesel hata gibi bir algoritmanın kullanılması uygun değildir çünkü verilerin tümü hazır bir şekilde (örnekler matrisi) bulunmamaktadır (Jain ve diğerleri, 1999).

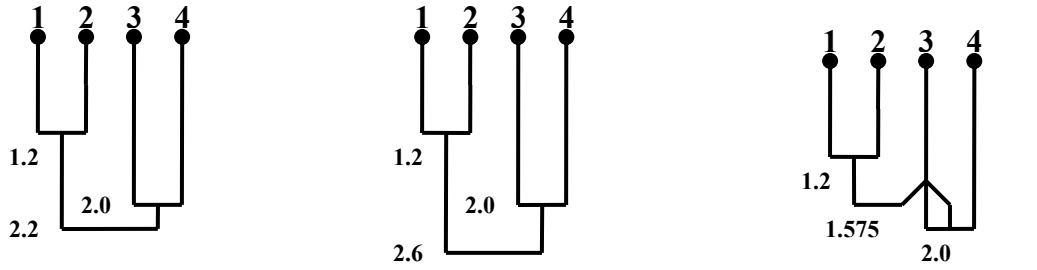
3.2.6 Dendrogramlarda Geçitler ve Monotonluk

Önceki bölümlerde anlatıldığı üzere, ultrametrik eşitsizliği sağlayan yakınlık matrisine sahip bir küme için, tek-bağ ve tam-bağ metotları ile oluşturulan dendrogramlar aynı yapıdadır. Kümeleme metotları farklı yapıda olsa da benzer yapıda dendrogramlara sahip olduklarından dolayı, yakınlık ve eşik çizgeleri ile oluşturulmuş bu iki metot monoton (Monotonicity) olarak kabul edilmektedir. Bu nedenle, bir sonraki oluşturulacak küme seviyesi varolan küme seviyesinden her zaman daha büyük olacaktır. Eğer herhangi bir metot (r) ve (s) kümelerini (r,s) kümesinde birleştiriyorsa (r) ve (s) kümesi hariç tüm kümeler için aşağıdaki şart sağlanmalıdır. Bunun nedeni, güncellenen matristeki yakınlık değerlerinden hiçbirisinin önceki matriste bulunan en küçük yakınlık değerinden daha küçük olamamasıdır. Diğer bir deyişle, bu iki metot ile oluşturulan kophenetik matrisi ultrametrik eşitsizliği sağlamaktadır.

$$d\left[(k),(r,s)\right] \geq d\left[(r),(s)\right] \tag{3.19}$$

Şekil 3.14'de verilen örnekte basit bir yakınlık matrisi ve farklı metotlar için oluşturulmuş dendrogramlar görülmektedir. UPGMC ve WPGMC metotları (kitle merkezi metotları) ile oluşturulan dendrogramlar monoton değildir ve üzerinde geçitler (Crossovers) ((x_1,x_2) ve (x_3,x_4) kümeleri (x_3,x_4) kümesinin ilk tanımlandığı seviyeden daha düşük bir seviyede birleşmiştir) bulunmaktadır.

$$\mathbf{D} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1.2 & 2.3 & 2.2 \\ & 0 & 2.4 & 2.6 \\ & & 0 & 2.0 \\ & & & 0 \end{bmatrix} \end{array} \tag{3.20}$$



Tek-Bağ Tam-Bağ WPGMC ve UPGMC

Şekil 3.14 Dendrogramlarda Bulunan Geçitler

Monotonluk kümeleme metotlarının bir özelliğidir ve yakınlık matrisi ile herhangi bir ilgisi bulunmamaktadır. Matris güncelleme algoritmasının bir avantajı, herhangi bir SAHN algoritmasının monotonluğunun katsayılar yoluyla tahmin edilebilmesidir. Şekil 3.14'deki gibi monoton sıradüzenlerin oluşturulması da mümkündür. Çizelge 3.1'de bulunan tüm metotlar monotondur, sadece tek-bağ ve tam-bağ metotları yakınlık değerlerinin monoton dönüşümlerinden etkilenmemektedir. Şekil 3.13'de görüldüğü gibi monoton olmayan kümelemelerde geçitler bulunmamaktadır. Geçitler, nesnelerin örnek uzayında, örnek matrisleri şeklinde bulunduğu ve uzaklık değerlerinin öklit uzaklığı ile hesaplandığı durumlar içinde dahi oluşabilmektedir (Jain ve Dubes, 1988).

3.3 Paylaştırmalı Kümeleme Algoritmaları

Sıradüzensel kümeleme teknikleri, verileri iç içe sıralı gruplar haline dönüştürmektedir. Sıradüzensel kümeleme metotlarının en önemli özelliği, kümelerin yakınlık seviyelerine göre birleşmelerini veya ayrılmalarını görsel bir şekilde ortaya koyan dendrogramlardır. Sıradüzensel olmayan kümele metotları, *paylaştırmalı* kümeleme metotları olarak adlandırılmaktadır. Bu metotlar, verilerin doğal olarak oluşturdukları grupları yeniden gruplandırmak amacıyla tek bir bölüm meydana getirmektedirler. Sıradüzensel kümeleme metotları genellikle nesneler arasında bulunan yakınlık değerleri ile oluşturulmuş yakınlık matrislerine gereksinim duymaktadır, oysa paylaştırmalı kümeleme teknikleri verileri örnek matrisleri olarak ele alıp işleme sokmaktadır, genellikle öznitelikler oransal değerlerdir.

Sıradüzensel teknikler sınıflandırmaların gerek duyulduğu biyoloji, sosyal ve davranışsal bilim alanlarında kullanılmaktadır. Paylaştırmalı teknikler ise, tek paylaşımların bulunduğu mühendislik uygulamalarında sıkça kullanılmaktadır. Paylaştırmalı kümeleme tekniklerinin büyük boyutlu veri tabanlarının etkili ve verimli bir şekilde gösteriminde ve sıkıştırılmasında kullanılması daha uygundur. Dendrogramlar yüzden fazla verilerin bulunduğu örnekler için elverişsizdir.

Paylaştırmalı kümelemenin basit olarak tanımı şu şekilde yapılmaktadır. d boyutlu bir uzayda verilen n örneğin, K tane kümeye paylaştırılmasıdır. Aynı kümede bulunan örnekler farklı kümelerdeki örneklere göre birbirlerine daha benzerdir. k değeri önceden belirlenmiş veya belirlenmemiş olabilmektedir. Tezde gerçeklenen algoritmalarda oluşacak küme sayısı önceden bilinmemektedir. Bu paylaştırma amacıyla, karesel-hata değeri gibi bir kümeleme kriteri belirlenmelidir. Kriterler, genel (global) veya yerel (local) olarak sınıflandırılmaktadırlar. Global kriterler, her kümeyi bir ilk örnek ile belirtmektedir ve örnekleri, kümelere seçilen prototipe göre atamaktadır. Yerel kriterler ise, verilerin yerel yapısından yararlanarak kümeleri oluşturmaktadır. Örneğin, kümeler örnek uzayında yüksek yoğunluklu bölgelerin belirlenmesiyle veya örneklerin ve k. en yakın komşularının (k nearest neighbours) aynı kümeye atanmasıyla oluşturulabilir.

Paylaştırmalı problemlerin kuramsal çözümü çok kolaydır. Basitçe, bir amaç fonksiyonu seçilir, k tane küme içeren olası tüm paylaştırmalar için değerlendirilir ve amaç fonksiyonunu en iyileyen paylaşım seçilir. Karşılaşılan ilk zorluk, "küme" ifadesini uygun bir matematiksel formüle çevirmektir. Seçilen amaç fonksiyonu problemin parametrelerine bağlıdır ve sayısal sebeplerden dolayı basit ve çeşitli veri yapılarına uygulanabilecek kadar karmaşık olmalıdır. Karşılaşılan ikinci zorluk ise, orta dereceli örnekler için dahi paylaşımların çok sayıda olmasıdır, böylece basit bir ama fonksiyonunun tüm paylaşımlara uygulanması elverişsiz olmaktadır.

Yukarıdaki açıklamalarda, amaç fonksiyonunun en iyilenmesinden bahsedilmişti, bu fonksiyonunun en iyilenmesinde en sık kullanılan tekniklerden bir tanesi tepe-tırmanma (hill-climbing) tekniğidir. Başlangıç olarak bir paylaşım seçilmektedir ve nesneler bir

kümeden diğerine amaç fonksiyonunun değerini yükseltmek amacıyla taşınmaktadır. Bu nedenle, her ardışık paylaşım bir önceki paylaşıma bağlıdır ve böylece sadece az sayıda paylaşım incelenmektedir. Bu tekniğe dayalı algoritmalar etkili ve verimli hesaplamalar yapmakta fakat genelde yerel minimumlara yakınsamaktadır. Başlangıç paylaşımının seçilmesi, örneklerin bir kümeden diğerine taşınması, kümelerin birleştirilmesi ve ayrılması ilerleyen bölümlerde anlatılacaktır.

Bir paylaşımı elde edebilmek için seçilebilecek tek bir *en iyi* yöntem bulunmamaktadır. Çünkü *küme* kelimesi için kesin ve uygulanabilir bir açıklama bulunmamaktadır. Kümeler çok boyutlu örnek uzaylarında rasgele şekillerde ve boyutlarda bulunmaktadır. Her kümeleme kriteri, veri üzerinde kesin bir yapı oluşturmaktadır ve eğer veri, yöntemin belirli kısmının gereksinimlerine uyuyorsa, doğru kümeler elde edilmiştir. Sadece az sayıda bağımsız kümeleme yöntemleri hem matematiksel hem de sezgisel olarak anlaşılabilmektedir. Bu nedenle literatürde önerilen yüzlerce karar fonksiyonu, birbiri ile ilişkilidir ve aynı yöntem çeşitli şekillerde bulunmaktadır. Örneğin, pek çok yöntem *karesel-hata (square-error)* tekniğinin versiyonlarıdır.

Bu bölümde anlatılacak olan paylaştırmalı kümeleme metotlarının çoğunda sürekli değerlere sahip öznitelik vektörleri kullanmıştır. Eğer öznitelikler sıra veya derece gösteriyorsa (ordinal) veya yazılı (nominal) şekilde ise öklit uzaklıkları ve küme merkezleri çok anlam ifade etmemektedir, böylece sıradüzensel metotlar çoğunlukla kullanılmaktadır. *Kavramsal kümeleme (Conceptual Clustering)* veya bir diğer adıyla örneklerden öğrenme, sayısal olmayan veya sembolik olarak tanımlanan nesnelere uygulanmaktadır. Burada amaç, nesneleri kavramsal basit kümelere ayırmaktır. Örneğin, arabaların kümelenmesinde tekerlek sayısı, jantın rengi, tekerleklerin rengi gibi özellikler veya bilgisayarların kümelenmesinde işlemci hızı, hafıza boyutu gibi özellikler göz önünde bulundurularak oluşturulan kümeler kavramsal kümelerdir. Kavramlar, özellikler cinsinden belirtilmektedir. Arabalar için oluşturulan sınıflandırmada, “mavi jantlı araba” kelimesi bir kavramdır. Nesneler kavramlar ile tanımlanan kümelere ayrılmaktadır.

3.3.1 Karesel-Hata Kümeleme Metotları

En sık kullanılan paylaştırmalı kümeleme teknikleri karesel-hata (Square-Error) kriterine dayanmaktadır. Bu yöntemde amaç, karesel-hata değerini (sabit sayıda küme için) en aza indirgeyen paylaştırmanın tespit edilmesidir. Önceki bölümde anlatılan Ward metodunda karesel-hata değeri farklı biçimde kullanılmıştır. Karesel-hatanın en aza indirgenmesi aynı zamanda kümeler arası değişimin en büyük hale getirilmesidir.

1) Başlangıçta, belirli sayıda küme merkezi ve küme sayısına sahip örnek paylaşımı seçilir.

2) Her örnek kendine en yakın küme merkezine atanır ve yeni küme merkezleri hesaplanır. Bu adım yakınsama gerçekleşinceye (örneğin, küme elemanları kararlı hale gelinceye kadar veya bir kümeden diğerine eleman aktarımı bitinceye kadar) kadar tekrarlanır.

3) Kümeler, sezgisel bilgilere göre birleştirilir ve ayrılır.

Karesel-hata kümeleme metotlarında gerçekleştirilen adımlar yukarıda belirtilmiştir. d boyutlu, n elemanlı bir dizi K kümeye $\{C_1, C_2, ..., C_K\}$ ayrılmış olsun. Buna göre, ve C_K kümesinde n_k tane örnek bulunmaktadır ve her örnek tamamen bir kümeye aittir.

$$\sum_{k=1}^{K} n_k = n \tag{3.21}$$

C_K kümesinin merkezi aşağıdaki Şekilde tanımlanmıştır.

$$\mathbf{m}^{(k)} = \left(1/n_k\right) \sum_{i=1}^{n_k} \mathbf{x}_i^{(k)} \tag{3.22}$$

Bu denklemde $\mathbf{x}_i^{(k)}$, C_K kümesinde bulunan $i.$ örneği ifade etmektedir. C_K kümesi için karesel-hata değeri, C_K kümesinde bulunan tüm elemanlar ile küme merkezleri

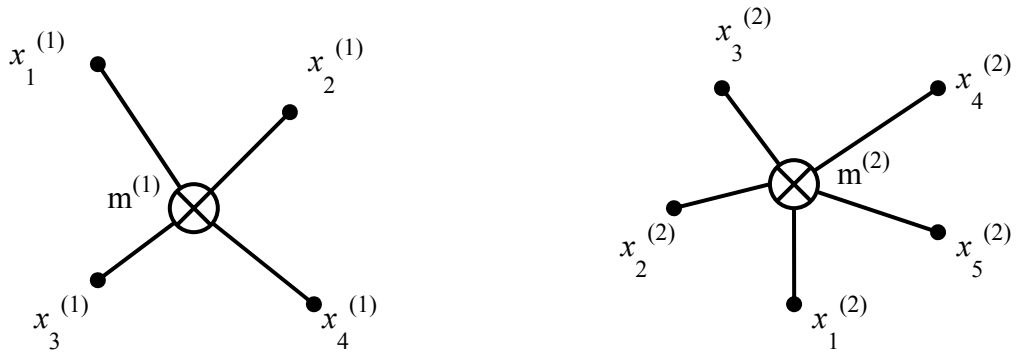
$\mathbf{m}^{(k)}$ arasında bulunan öklit uzaklıklarının toplamıdır. Bu karesel-hata küme içi değişim olarak da adlandırılmaktadır.

$$e_k^2 = \sum_{i=1}^{n_k} \left(\mathbf{x}_i^{(k)} - \mathbf{m}^{(k)} \right)^T \left(\mathbf{x}_i^{(k)} - \mathbf{m}^{(k)} \right) \tag{3.23}$$

Karesel-hata değerinin hesaplanmasında Mahalanobis uzaklık ölçümü gibi farklı bir ölçüt de kullanılabilir. K tane kümeyi de içeren tüm kümeleme için karesel-hata değeri küme içi değişimlerin toplamına eşittir.

$$E_K^2 = \sum_{k=1}^{K} e_k^2 \tag{3.24}$$

Önceden de belirtildiği gibi, karesel-hata kümeleme metodunda amaç, K tane kümeyi de içinde bulunduran ve sabit K değeri için E_K^2 değerini minimize eden paylaşımı bulmaktır. Şekil 3.15'de karesel-hata kriterinin küme merkezlerini ilk örnek olarak seçtiği görülmektedir. Hata ise, örnek noktaların küme merkezinden sapmalarını göstermektedir. Başka bir deyişle, örnekler K tane küresel kümenin toplamı gibi varsayılmaktadır. Karesel-hata, bu K tane küresel kümeyi olabildiğince yoğun ve ayrık hale getirmeye çalışmaktadır. FORGY (Forgy, 1965) ve CLUSTER (Dubes ve Jain, 1965) algoritmaları karesel-hata değerine göre işlem yapan algoritmalardır.

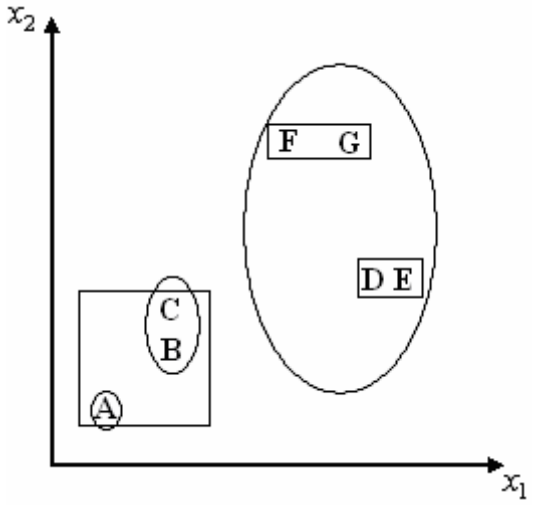


Şekil 3.15 Karesel-Hatanın Hesaplanmasında Kullanılan Uzaklıklar

3.3.1.1 k-yol Algoritması

k-yol (k-means) algoritması en basit ve çok kullanılan karesel-hata tabanlı algoritmadır. Rasgele bir paylaşım ile başlar ve örnekleri, küme merkezlerine (herhangi bir yakınsama gerçekleşinceye kadar) olan uzaklıklarına göre yeni kümelere tekrar atar. k-yol algoritmasının bu kadar sık kullanılmasının sebebi, gerçekleştirilmesinin kolay olması ve zaman karmaşıklığına yol açmamasından kaynaklanmaktadır. Bu algoritmanın kullanımında karşılaşılan en büyük problem başlangıç paylaşımının seçilmesi ve başlangıç paylaşımı düzgün bir şekilde seçilmediyse işlemler sırasında amaç fonksiyonunun yerel en küçük değerlere yakınsamasıdır.

Şekil 3.16'da yedi tane iki boyutlu örnek bulunmaktadır. Eğer A, B ve C örnekleri ile başlarsak, sonuç olarak elipsler ile gösterilen üç küme oluşmaktadır $\{A\}, \{B, C\}, \{D, E, F, G\}$. Karesel-hatanın değeri, elipsler ile gösterilen kümeler için daha büyüktür. En iyi paylaştırma, $, \{A,B, C\}, \{D, E\}, \{F, G\}$ şeklindedir ve dikdörtgenler ile gösterilmiştir. Bu paylaşım sonucunda karesel-hata fonksiyonun global minimum değeri elde edilmiştir ve başlangıç olarak A, D, F seçilmiştir.



Şekil 3.16 k-yol Algoritması ile Oluşturulan Kümeler

k- yol algoritmasının adımları şu şekildedir:

1) Örnek uzayından k tane rasgele seçilmiş örnek veya k tane rasgele belirlenmiş nokta ile aynı anda k tane küme merkezi seçilir.

2) Her örneği kendine en yakın küme merkezine atar.

3) Küme merkezleri varolan küme üyeliklerine göre yeniden hesaplanır.

4) Eğer herhangi bir yakınsama kriteri sağlanamazsa 2. adıma dönülür. En çok kullanılan yakınsama kriterlerinden bazıları şunlardır: Herhangi bir örneğin yeni küme merkezine atanması gerçekleşmiyorsa veya karesel-hata değerinde azalma gerçekleşmiyorsa işlemlere son verilir.

Başlangıç paylaşımı, k tane kaynak nokta belirlenerek oluşturulabilir. Bu kaynak noktalar örnek matrisindeki ilk k nokta veya bu matristen seçilecek rasgele k nokta olabilir. Bu ilk paylaşım veya kümeleme her noktayı kendine en yakın kaynak noktaya atayarak gerçekleştirilir. Oluşan kümelerin kitle merkezleri başlangıç küme merkezleridir. Başlangıç kümesinin oluşturulmasında sıradüzensel kümeleme metotları da kullanılabilmektedir. Farklı başlangıç kümesi seçimi sonuçlarında farklı olmasına sebep olmaktadır çünkü karesel-hata değeri ile işlem yapan algoritmalar yerel en küçük noktalara takılabilmektedir. Yerel en küçük noktalardan kurtulmanın bir yolu farklı başlangıç kümeleri oluşturmaktır.

Paylaşımın güncellenmesi, karesel-hata değerini azaltacak Şekilde örneklerin kümelere atanmasıdır. k-yol geçişi (pass) tüm örnekleri en yakın küme merkezine atamaktadır. Bunun dışında farklı metotlar da bulunmaktadır (Jain ve Dubes, 1988).

Küme sayılarının ayarlanması, eğer bir kümede çok fazla örnek bulunuyorsa bölünür, küme merkezleri birbirlerine çok yakınsa kümeler birleştirilir.

Yakınsama, paylaştırmalı algoritmalar eğer kriter fonksiyonunda herhangi bir geliştirme yapılamıyorsa sonlandırılır. Yinelemeli algoritmalar hiçbiri için global en küçük değere yakınsama garantisi yoktur. Bazı algoritmalar, yinelemeler sonucunda tüm örnekler için küme değerleri değişmiyorsa işlem sonlanmaktadır. Bunun yanı sıra yineleme sayısı algoritma uygulanmadan önce belirlenebilmektedir. Pratikte, k-yol algoritmaları çok kısa sürede yakınsamaktadır.

Literatürde çeşitli *k*-yol algoritmaları bulunmaktadır, bunlardan bazıları algoritmanın global en küçük değerini daha çabuk bulabilmesi amacıyla iyi bir başlangıç paylaşımının seçilmesine çalışmaktadır. Tipik olarak, bir kümenin değişimi daha önceden belirlenmiş eşik değerinin üzerinde ise ayrılır, benzer Şekilde, iki küme eğer kitle merkezleri arasındaki uzaklıklar önceden belirlenmiş eşik değerinin altında ise bir araya getirilir. Yaygın olarak kullanılan ISODATA algoritması bu tekniği kullanarak kümeleri bir araya getirir veya ayırır (Jain ve diğerleri, 1999).

3.3.2 Karışım-Ayırma ile Kümeleme

Verilerin kümelenmesi amacıyla sıkça kullanılan bu metot *karışım yoğunluğu (mixture density)* kavramına dayanmaktadır. Kümelenecek tüm noktalar, K bölgeden veya kümeden seçilmektedir. Burada amaç, her örneği doğru bölgeye yerleştirmektir. *Yoğunluk-Tahmini (Density Estimation)* veya *Yüksek Yoğunluğa Sahip Bölge-Arama (Mode Seeking)* algoritmalarının tersine, bölgelerin şekli ve sayısı biliniyor kabul edilmektedir. Örnek noktalar bölgelere göre etiketlenmemiştir. Eğer bölgelerin yoğunluk değerleri örnek noktalar yardımı ile tahmin edilebiliyorsa, her nokta tahmin edilen yoğunluk olasılıklarına göre kendi uygun bölgesine atanabilir.

3.3.3 Yoğunluk-Tahmini veya Durum-Arama

Kümeler, örneklerin yoğun olarak bulunduğu örnek uzayının bölgeleri olarak görülebilir. Ayrıca kümeler, durum olarak ifade edilen yüksek yoğunluğa sahip bölgelerin örnek uzayında aranması ile belirlenebilir. Her durum bir küme merkezi ile ilişkilidir ve noktalar, kümelere bulundukları en yakın merkeze göre atanır. Verilerde bulunan durumları belirlemenin en basit yolu, örnek uzayını birbiri ile çakışmayan bölgelere ayırarak, istatistiksel grafikler (histogram) oluşturmaktır. Örnek sayısı, yoğunluk fonksiyonunun iyi bir şekilde tahmin edilebilmesi amacıyla yeteri kadar büyük olmalıdır. Yüksek yoğunluğa sahip bölgeler muhtemel durumlar veya küme merkezleridir ve kümeler arasındaki sınırlar bu istatistiksel grafiklerde kullanılan vadi (valley) ile ifade edilmektedir. Çok sayıda örneğe ihtiyaç duyulsa da bu yaklaşımın başarılı olabilmesi iki şarta bağlıdır. İlk olarak, düşük boyutlu bölgeler kötü yoğunluk

tahminine neden olurken, yüksek boyutlu bölgeler yoğunluk tahmininde daha verimli olmaktadır. İkinci olarak, istatistiksel grafiklerde bulunan tepeleri (peaks) ve vadileri (valleys) belirleyebilmek için işlemler, boyutu bilinen bir komşu üzerinden gerçekleştirilmektedir. Bu nedenlerden dolayı boyut arttıkça işlemler zorlaşmaktadır.

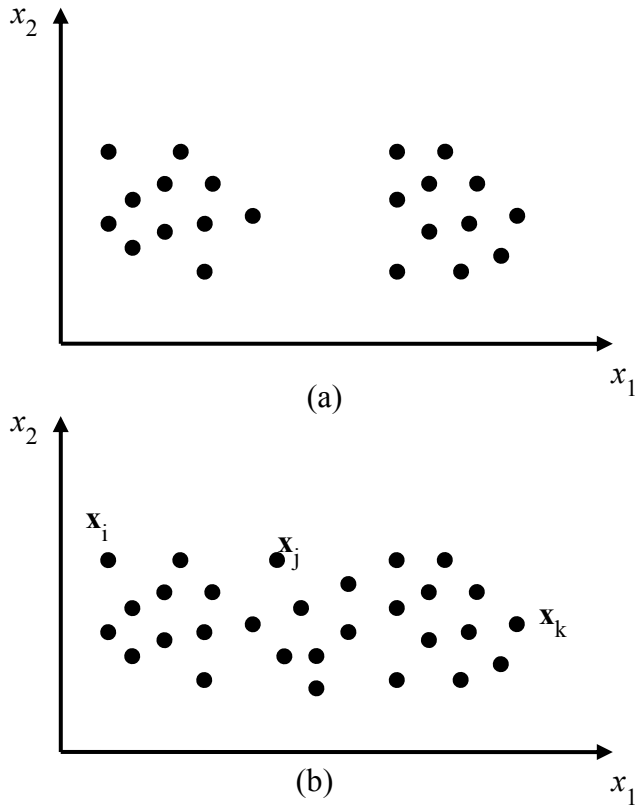
Çok boyutlu istatistiksel grafiklerin araştırılması ve saklanması için gerekli olan bellek ve işlem zamanı çok büyük olmaktadır.

3.3.4 Bulanık Kümeleme

Şu ana kadar anlatılan tüm algoritmalar örnekleri tek bir kümeye yerleştirmektedir. Diğer bir deyişle, örnekler ayrık dizilere ayrılmaktadır, aynı kümede bulunan örnekler diğer kümelerde bulunan örneklere göre birbirlerine daha benzerdir. Eğer kümeler yoğun ve birbirlerinden iyi şekilde ayrılmışlar ise, örnekleri kümelere yerleştirirken herhangi bir belirsizlikle karşılaşılmamaktadır. Şekil 3.17 (a)'da iyi şekilde ayrılmış, küme sınırları belirgin iki küme görülmektedir. Eğer küme sınırları belirgin değilse ve kümeler iç içe geçmiş ise (Şekil 3.17 (b)), örneklerin kümelere yerleştirilmesi zorlaşacaktır. $\mathbf{x}_i$ ve $\mathbf{x}_k$ örnekleri farklı kümelere yerleştirilmelidir fakat $\mathbf{x}_j$ örneği $\mathbf{x}_i$ veya $\mathbf{x}_j$'nin bulunduğu kümelerden birine yerleştirilebilir. Böyle kümeler bulanık sınırlara sahip olmalıdır.

Bulanık küme teorisi 1965 yılında Azerbaycanlı Lotfi Zadeh tarafından ortaya konulmuştur. Bu teoreme göre bir nesne üyelik derecesine göre farklı kümelere ait olabilmektedir. Üyelik derecesi [0, 1] arasında değerler alabilmektedir. Sıradan kümeler için (kesin kümeler), eğer $\mathbf{x}_i$ kümenin elemanı ise üyelik derecesi 1 değil ise 0'dır. Bulanık kümelerde ise $\mathbf{x}_i$, $f_q\left(\mathrm{X}_i\right) \geq 0$ gibi bir üyelik derecesine sahiptir veya $\sum_q f_q\left(\mathbf{x}_i\right) = 1$ olan $q.$ kümeye aitlik derecesine sahiptir. Eğer üyelik derecesi 1'e eşit ise örnek o kümeye tamamen dahildir. Üyelik derecesi arttıkça kümeye aitlik artmaktadır. Aristo mantığına göre bir eleman ya o kümeye dahildir ya da dahil değildir yani alacağı

değer ya 1'dir ya da 0'dır fakat bulanık küme teoremine göre $\mathbf{x}_i$ örneği aynı anda birden fazla kümeye ait olabilmektedir.



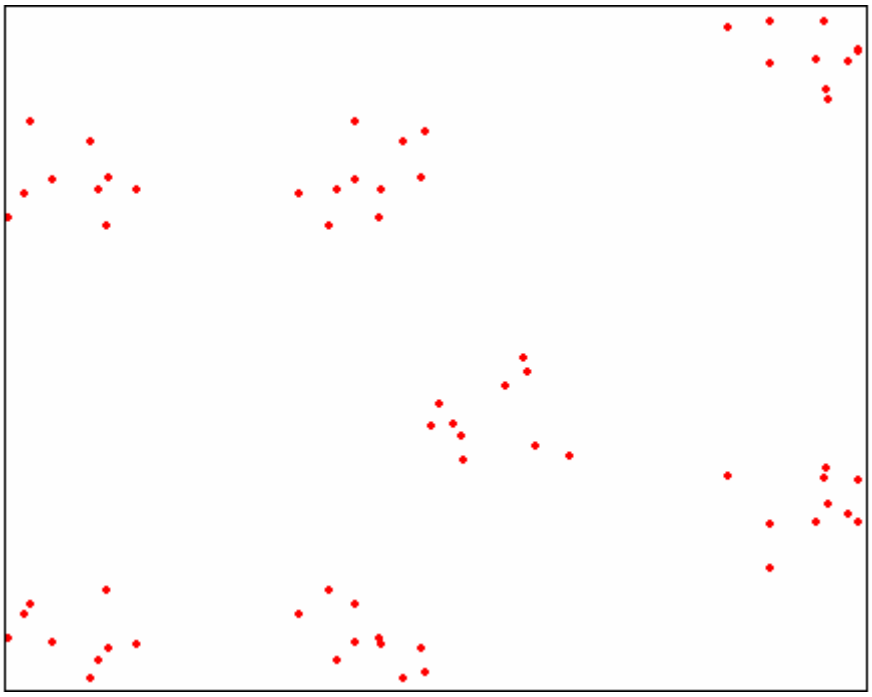
Şekil 3.17 (a) İyi Ayrılmış Kümeler, (b) İç içe Geçmiş Kümeler

Bulanık küme teorisine dayalı kümeleme algoritmalarının çoğu paylaştırmalı kümeleme yapmaktadır, çok az bir kısmı hiyerarşi oluşturmaktadır. Bulanık kümelemede en önemli adım üyelik fonksiyonlarının belirlenmesidir.

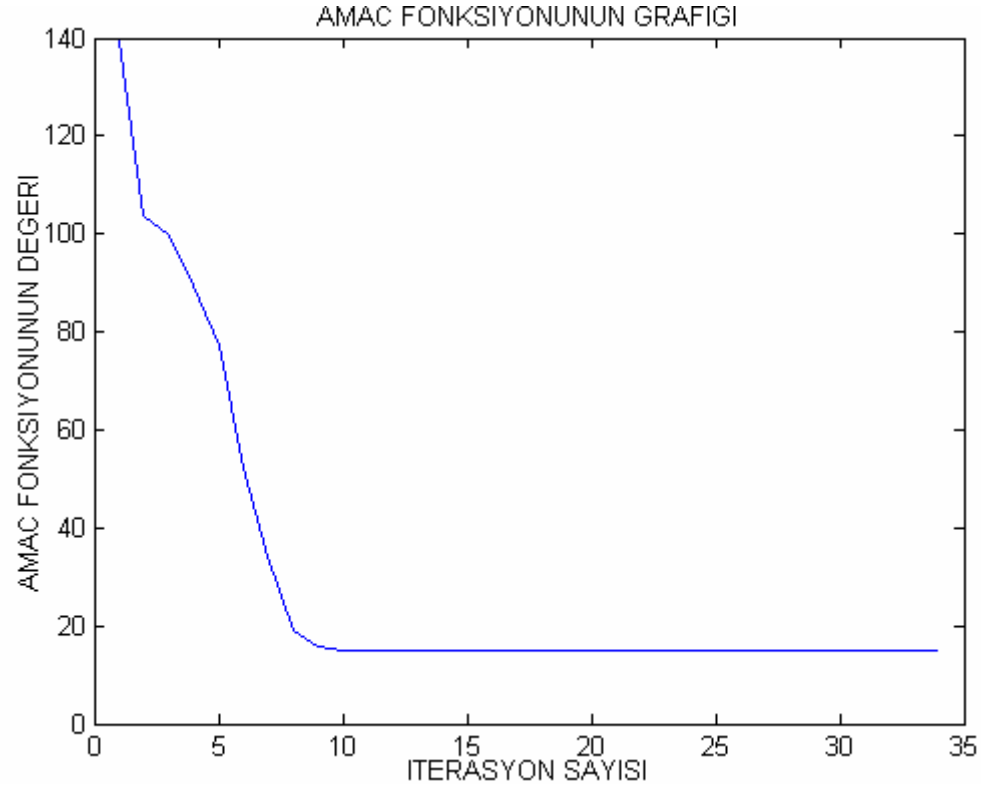
En fazla kullanılan bulanık kümeleme algoritması, Bulanık C-yol (Fuzzy C-Means - FCM) algoritmasıdır. Yerel minimumlara takılmama konusunda k-yol algoritmasına göre daha iyi olmasına rağmen FCM, karesel-hata değerinin yerel minimumlarına takılmaktadır. Önceden de ifade edildiği gibi üyelik fonksiyonlarının belirlenmesi bulanık kümelemede en önemli işlemdir. FCM, başlangıçta rasgele küme merkezleri belirlemektedir. Buna ek olarak FCM her noktayı üyelik derecelerine göre kümelere yerleştirmektedir. Tekrarlamalı olarak küme merkezleri ve her noktanın üyelik derecesi

belirlenerek küme merkezleri doğru noktaya yerleştirilmektedir. Bu tekrarlama işlemi, küme merkezi ile noktalar arasındaki uzaklığı noktaların üyelik dereceleri cinsinden ifade eden bir amaç fonksiyonunun (yukarıda belirtilen kriter fonksiyonu gibi) minimize edilmesi ile gerçekleşir.

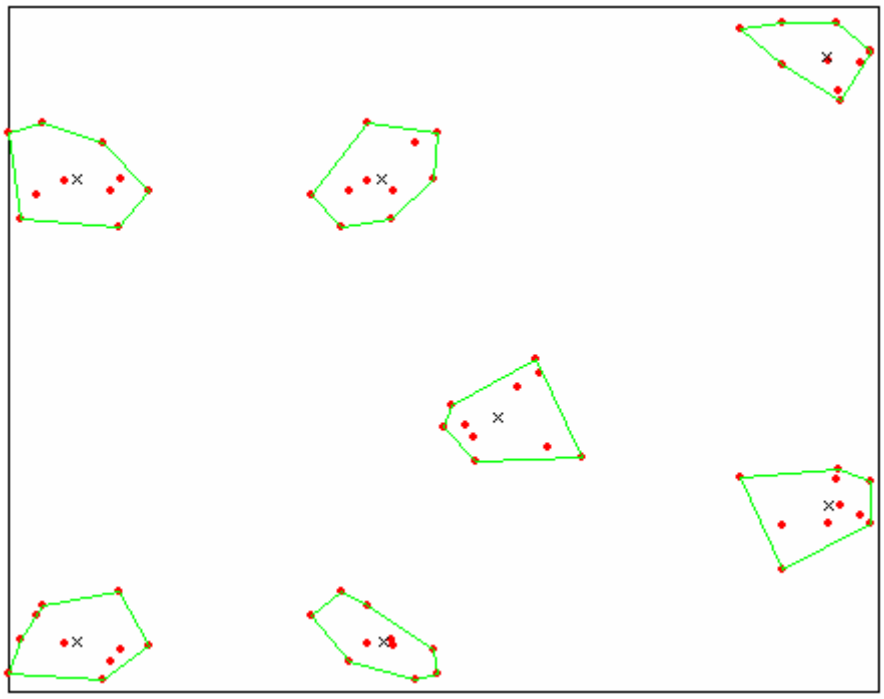
Şekil 3.18 (a)'da sırasıyla dağılmış veriler, Şekil 3.18 (b)'de amaç fonksiyonun grafiği ve Şekil 3.18 (c)'de bulanık kümeler görülmektedir (Şekilde bulunan x işaretleri küme merkezlerini göstermektedir).



Şekil 3.18 (a)Dağılmış Noktalar



Şekil 3.18 (b) Amaç Fonksiyonunun Grafiği



Şekil 3.18 (c) Bulanık Kümelere Ayrılmış Noktalar

3.3.5 Yapay Sinir Ağları ile Kümeleme

Yapay sinir ağları (YSA) biyolojik sinir ağlarından esinlenerek ortaya atılmıştır. YSA son otuz yıldır çok yaygın olarak kümeleme ve sınıflandırma işlemlerinde kullanılmaktadır. YSA aşağıdaki sebeplerden dolayı örnek kümelemede kullanılmaktadır.

1) YSA, sayısal vektörlerle işlem yapmaktadır ve bu nedenden dolayı kullanılacak örneklerin nicel özellikleri göz önünde bulundurulmalıdır.

2) YSA, paralel ve ayrık işlemler yapabilmektedir.

3) YSA, birbirleri arasındaki ağırlık değerlerini öğrenebilmektedir. Kısaca YSA, örnekleri düzgeleyebilen ve uygun ağırlıkların seçilmesiyle öznitelikleri belirleyebilen bir yapıya sahiptir.

Genelde giriş verilerini kümeleme amacıyla Rekabetçi Sinir Ağları (Competitive Neural Networks), kullanılmaktadır. Rekabetçi öğrenme işleminde, birbirine benzer özellikte bulunan veriler ağ tarafından gruplandırılır ve tek birim (nöron) ile ifade edilir. Bu gruplama işlemi verilerin karşılıklı ilişkilerine göre kendiliğinden yapılmaktadır. Kümeleme işlemlerinde kullanılan en yaygın YSA yapıları, Kohonen Öğrenmesi (Kohonen's Learning Vector Quantization – LVQ), SOM (Self-Organizing Map) ve Uyarlamalı Rezonans Teorisi (Adaptive Resonance Theory) yapılarıdır. Bu mimarilerin çalışma prensipleri çok basittir. Hepsi tek katmandan oluşmaktadır ve örnekler girişi temsil eder ve çıkış ile ilişkilendirilir. Giriş ve çıkış arasında bulunan ağırlık değerleri önceden belirlenmiş bir sonlandırma kriterine ulaşılıncaya kadar tekrarlamalı olarak değişmektedir (öğrenme bu şekilde gerçekleşmektedir) (Hagan ve diğerleri, 1996, Jain ve Dubes, 1988, Jain ve diğerleri, 1999).

Herhangi bir örnek değeri, farklı tekrarlamalar için farklı çıkış değerlerini tetikleyebilir, bu özellik öğrenebilen sistemlerde *kararlılık* olarak belirtilmektedir. Bir sistem, eğer girişlerden herhangi birisi tekrarlamalardan sonra çıkış üzerinde bir değişikliğe neden olmuyorsa *kararlıdır* denilmektedir. Kararlılık konusu, algoritmanın yeni verilere adapte olması özelliği olarak da bilinen *uyarlanabilme* konusu ile yakından

ilgilidir. Kararlılığın sağlanabilmesi için öğrenme oranı tekrarlamalar sonucunda sıfıra yaklaşmalıdır ve bu da uyarlanabilme özelliğini etkilemektedir. Tüm YSA modelleri, oluşturulacak küme sayısını sınırlayan sabit çıkış sayısına sahiptir.

Bu kümeleme metot ve algoritmalarının dışında daha birçok kümeleme metot ve algoritması bulunmaktadır günümüzde veri tabanlarının giderek daha da büyümesi dolayısıyla daha güncel algoritmalar ortaya çıkmaktadır. Tezde kullanılan algoritmalar dördüncü bölümde detaylı olarak anlatılmaktadır.

3.3.6 Medoidler Etrafında Gruplama (PAM)

Bu algoritmada verilerin giriş sırası sonucu etkilememektedir. İyi ölçeklenebilen bir algoritma değildir. Her bir küme *medoid* adı verilen bir eleman ile tanımlanır. Medoid, bir kümedeki tüm elemanlara olan uzaklıklarının ortalaması en küçük olan eleman olarak tanımlanmaktadır. K adet medoid'den oluşan set öncelikle rasgele seçilir ve diğer elemanlar kendisine en yakın medoid ile aynı kümeye dahil edilmektedirler. K-Medoids algoritması olarak da bilinmektedir (Partitioning Around Medoids). Medoid yaklaşımı kullanılması aykırı değerlerin (outlier) daha etkili değerlendirilmesini sağlamaktadır. Her yeni elemanın bir medoid olup olamayacağı değerlendirilerek kümeleme kalitesi artırılmaya çalışılmaktadır. Hesaplamalar karmaşık olduğundan dolayı geniş veri setleri için uygun değildir. (www.cs.ualberta.ca / ~zaiane / courses / cmput690 / slides / Chapter8 / sld023.htm, www.unesco.org / webworld / idams / advguide / Chapt7_1_1.htm).

3.3.7 CLARA Algoritması

Clara (CLustering LArge Applications) kümeleme tekniği veri setini örnekleyerek PAM algoritmasının zaman karmaşıklığı değerini düşürmektedir. Geniş veri setlerinde PAM algoritmasına göre daha etkili kümeleme yapmaktadır. Algoritmanın ne kadar etkili olduğu örnekleme boyutuna göre değişiklik göstermektedir (40+2k örnek alınmasının

iyi sonuçlar verdiği belirtilmiştir) (http://db.cs.sfu.ca / GeoMiner / survey / html / node9.html).

3.3.8 CLARANS Algoritması

CLARANS (Clustering LARge Applications based upon raNdomized Search) algoritmasında, PAM algoritmasının giriş değerlerine ek olarak en yakın komşular da göz önünde bulundurulmaktadır. Tüm verinin ana bellekte olması koşuluyla PAM ve CLARA algoritmalarına göre daha etkili kümeleme yapmaktadır (Raymond ve Han, 2002).

3.3.9 BEA Algoritması

BEA algoritması (Bond Energy Algorithms), veritabanı tasarımında, verinin nasıl gruplanacağını ve fiziksel olarak sabit diske ne şekilde yazılacağı belirlemekte kullanılmaktadır. Birlikte kullanılan özelliklerin kümelenmesine dayanmaktadır. Sonuç olarak ortaya çıkan her kümeye dikey parça (*vertical fragment*) adı verilir ve her bir parça ayrı yerlere kaydedilmektedir.

3.4 Büyük Veri Tabanlarında Kümeleme

Bu bölüme kadar anlatılan algoritmalar belleğe sığan ve dinamik olmayan veri setleri için tasarlanmıştır. Geniş veritabanlarında kümeleme yapmak için, veritabanının bir kez veya daha az taranması, çevrimiçi çalışabilme özelliği, askıya alınabilir, durdurulabilir ve geri dönülebilir olma, veri ekleme veya çıkarma sonucu güncelleme, kısıtlı bellek ile çalışabilme, tarama sırasında farklı teknikler kullanabilme, bir kaydın sadece bir kez işlenmesi gibi kriterlere dikkat edilmektedir.

Kümeleme algoritmalarının çoğu bellek yerleşimli büyük veri yapılarında işlem yapmaktadırlar. Kümeleme öncelikle bir örnek set üzerinde uygulanmaktadır ve daha sonra tüm veri tabanında uygulanır. En çok kullanılan kümeleme algoritmaları şunlardır: BIRCH, DBSCAN, CURE.

3.4.1 BIRCH Algoritması

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algoritması büyük boyutlu verilerin kümelenmesi için tasarlanmıştır. Artımlı ve sıradüzensel bir teknik kullanılmaktadır. Kümeleme için gerekli bilgiyi yakalama amacıyla bir ağaç oluşturulur. Kümeler hakkındaki bilgileri tutan *Kümeleme Özniteliği* (Clustering Feature) özelliğine dayanır. Hem bellek, hem de zaman bakımından doğrusaldır. Algoritmanın etkili çalışmasında eşik değerinin seçimi çok önemlidir. Eğer eşik değeri uygun değilse ağacın tekrar tekrar baştanbaşa oluşturulması gerekebilir (Zhang ve diğerleri, 1996).

3.4.2 DBSCAN Algoritması

DBSCAN (Density Based Spatial Clustering of Applications with Noise) algoritması minimum boyuta ve yoğunluğa sahip olan kümeler oluşturmayı hedeflemektedir. Yoğunluk, birbirlerinden belli uzaklıkta bulunan en az sayıda nokta olarak tanımlanmaktadır. Kümelenmemiş değerlerin yeni kümeler oluşturması engellenmektedir ve herhangi bir kümeye yerleştirilemeyen noktalar *gürültü* olarak değerlendirilmektedir. DBSCAN, CLARANS ile karşılaştırılmış ve veri setindeki tüm kümeleri ve gürültü değerlerini bulduğu gözlenmiştir (CLARANS daha etkisiz) (Grizaite ve Innerhofer-Oberperfler, 2005).

3.4.3 CURE Algoritması

CURE (Clustering Using REpresentatives) algoritmasında *Daralma Etkeni* (Shrinkage Factor) kullanılarak kümelenmemiş değerler kümelerin kitle merkezi değerlerine doğru çekilirler. BIRCH ve MST yaklaşımlarıyla karşılaştırılmış ve her ikisinden de daha kaliteli kümeler oluşturduğu görülmüştür (Guha ve diğerleri, 1998).

3.5 Kategorik Özellikler İle Kümeleme

Geleneksel algoritmalar kategorik veriler üstünde her zaman etkili çalışmamaktadırlar. Sorunun tanımlanması amacıyla çeşitli teknikler kullanılabilmektedir (Sıradüzensel algoritmalar gibi).

3.5.1 ROCK Algoritması

Hem *boolean*, hem de kategorik veriyi hedeflemektedir. Veriler arasındaki bağlantıların sayısına dayanan bir benzerlik ölçütü kullanılmaktadır. Bir eleman çifti arasındaki benzerlik değeri eşik değerini aşarsa komşu olarak kabul edilmektedirler. ROCK (Robust Clustering Using Links) algoritması dışında COOLCAT, LIMBO ve CLICK gibi algoritmalar da kullanılmaktadır (Guha ve diğerleri, 2000).

Tanımlanan bu algoritmalar dışında farklı amaçlar da kullanılmak üzere birçok kümeleme algoritması bulunmaktadır. Çizelge 3.2'de farklı kümeleme algoritmaları hakkında kısa bilgiler sunulmaktadır.

Çizelge 3.2 Kümeleme Algoritmalarının Karşılaştırılması

Algoritma	**Tür**	**Alan**	**Süre**
Tek Bağ	Sıradüzensel	$O(n^2)$	$O(kn^2)$
Ortalama Bağ	Sıradüzensel	$O(n^2)$	$O(kn^2)$
Tam Bağ	Sıradüzensel	$O(n^2)$	$O(kn^2)$
MST	Sıradüzensel/Paylaştırmalı	$O(n^2)$	$O(n^2)$
Karesel Hata	Paylaştırmalı	O(n)	O(tkn)
K-yol	Paylaştırmalı	O(n)	O(tkn)
En Yakın Komşu	Paylaştırmalı	$O(n^2)$	$O(n^2)$
PAM	Paylaştırmalı	$O(n^2)$	$O(tk(n-k)^2)$
BIRCH	Paylaştırmalı	O(n)	O(n) Tekrar oluşturma yok
CURE	Karma	O(n)	O(n)
ROCK	Toplayıcı	$O(n^2)$	$O(n^2 lgn)$
DBSCAN	Karma	$O(n^2)$	$O(n^2)$

DÖRDÜNCÜ BÖLÜM
TEZDE KULLANILAN ALGORİTMALAR

4. TEZDE KULLANILAN ALGORİTMALAR

Bu bölümde, tezde gerçeklenen algoritmalar yer almaktadır. Gerçeklenen algoritmaların adımları kısaca anlatılmakta ve uygulama alanları hakkında kısaca bilgi verilmektedir.

4.1 En Küçük Tarama Ağacı Algoritması

En küçük tarama ağacı (Minimum Spanning Tree - MST) algoritması, aynı zamanda ZAHN kümeleme algoritması olarak da ifade edilmektedir. Algoritmanın adımları aşağıdaki şekildedir:

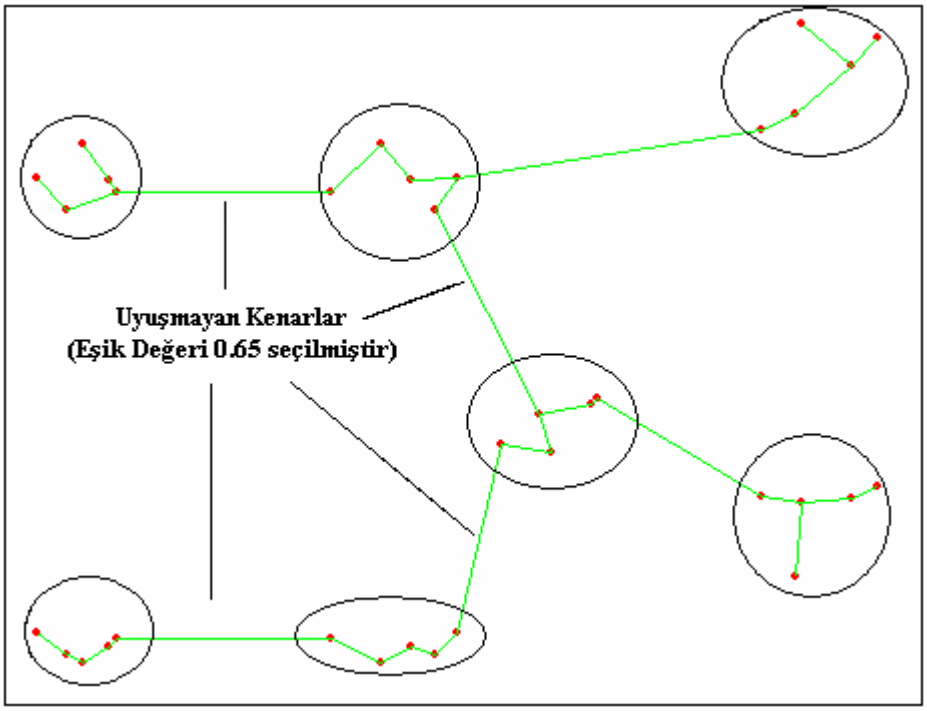
1) Verilen *n* örnek için MST oluşturulur (Öklit uzaklık ölçümleri kullanılarak kenarların ağırlıkları belirlenir).

2) MST'de bulunan uyuşmayan kenarlar tespit edilir.

3) Uyuşmayan kenarlar silinerek bağlı elemanlar elde edilir ve kümeler olarak adlandırılır.

Algoritmada bulunan en önemli adım uyuşmayan kenarların tespit edilmesidir. Bu kenarların tespit edilebilmesi amacıyla ZAHN çeşitli kriterler belirlemiştir. Mesela, eğer bir kenarın ağırlığı (Öklit uzaklığı) kendisine yakın olan kenarların ağırlıklarının ortalamasından oldukça büyük ise o kenar uyuşmayan kenar (inconsistent edge) olarak belirlenir. Böylece, uyuşmayan kenarlar küme ayırımları ile ilişkilendirilmiştir. Şekil 4.1'de uyuşmayan kenarlar ve belirlenen kümeler görülmektedir. Bu iki küme ikinci adımda E ve F arasında bulunan kenar ortadan kaldırılarak üç kümeye ayrılabilir.

Ayrıca belirlenen eşik değerinden daha büyük ağırlık değerlerine sahip olan kenarlarda uyuşmayan kenar olarak seçilebilmektedir.



Şekil 4.1 MST ile Kümelerin Oluşturulması

Öncelikle daha yoğun kümeler tespit edilmeli ve silinmeli, sonra kalan verilerin analizi yapılmalıdır. MST'nin iki boyuttan daha fazla boyutlu durumlarda uygulanması zorlaşmaktadır. Yine de MST ile kümeleme, karesel-hata değerini kullanan paylaştırmalı kümeleme tekniklerinin en önemlileri arasındadır.

4.2 Bağıl Komşuluk Değeri Ve Gabriel Çizge Algoritmaları

Küme analizinde MST dışında yukarıda da ifade edilen bağıl komşuluk değeri (Relative Neighborhood Value-RNG) ve gabriel çizge (Gabriel Graphs-GG) metotları da kullanılmaktadır. RNG ve GG algoritmaları örnekler üzerinde sezgisel gruplamayı elde edebilmek amacıyla uygulanmıştır. Bu çizgeler, tam çizgede bulunan $n(n-1)/2$ kenardan, yapıyı oluşturabilmek amacıyla bir alt küme seçerler. Ayrıca bu çizgeler sadece kenarların uzunluklarına bağlıdır.

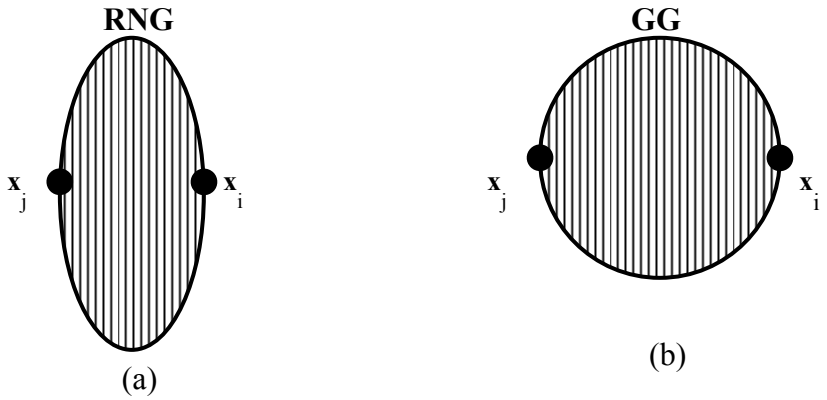
RNG metodunda $\mathbf{x}_i$ ve $\mathbf{x}_j$ örnekleri bağıl komşular olarak tanımlanmıştır ve sadece aşağıdaki koşul sağlanıyorsa birbirine bağlıdır.

$$d\left(\mathbf{x}_i,\mathbf{x}_j\right) = \max\left\{d\left(\mathbf{x}_i,\mathbf{x}_k\right), d\left(\mathbf{x}_j,\mathbf{x}_k\right)\right\};\ \textit{tüm k değerleri için},\ k \neq i, k \neq j \tag{4.1}$$

Denklem (4.1)'de bulunan $d\left(\mathbf{x}_i,\mathbf{x}_j\right)$ ifadesi, $\mathbf{x}_i$ ve $\mathbf{x}_j$ arasındaki öklit uzaklığını ifade etmektedir. Bir başka deyişle, diğer hiçbir nokta $LUNE\left(\mathbf{x}_i,\mathbf{x}_j\right)$'da bulunmuyorsa, $\mathbf{x}_i$ ve $\mathbf{x}_j$ örnekleri RNG'de birbirine bağlıdır. $LUNE\left(\mathbf{x}_i,\mathbf{x}_j\right)$ ifadesi, $\mathbf{x}_i$ ve $\mathbf{x}_j$'de merkezlenmiş ve $d\left(\mathbf{x}_i,\mathbf{x}_j\right)$ yarıçaplı iki diskin kesişimini ifade etmektedir. $LUNE\left(\mathbf{x}_i,\mathbf{x}_j\right)$, RNG'nin etki bölgesidir ve Şekil 4.2 (a)'da gösterilmiştir. GG ise şu şekilde tanımlanmaktadır, $\mathbf{x}_i$ ve $\mathbf{x}_j$ noktaları GG'de sadece aşağıdaki eşitliğin sağlandığı durumlarda birbirine bağlıdır.

$$d^2\left(\mathbf{x}_i,\mathbf{x}_j\right) < d^2\left(\mathbf{x}_i,\mathbf{x}_k\right) + d^2\left(\mathbf{x}_j,\mathbf{x}_k\right);\ \textit{tüm k değerleri için},\ k \neq i, k \neq j \tag{4.2}$$

Diğer hiçbir nokta $DISK\left(\mathbf{x}_i,\mathbf{x}_j\right)$'de bulunmuyorsa, $\mathbf{x}_i$ ve $\mathbf{x}_j$ noktaları GG'de birbirine bağlıdır. $DISK\left(\mathbf{x}_i,\mathbf{x}_j\right)$ ifadesi, $d\left(\mathbf{x}_i,\mathbf{x}_j\right)$ çaplı diski ifade etmektedir. DISK, GG'nin etki bölgesidir ve Şekil 4.2 (b)'de gösterilmiştir.

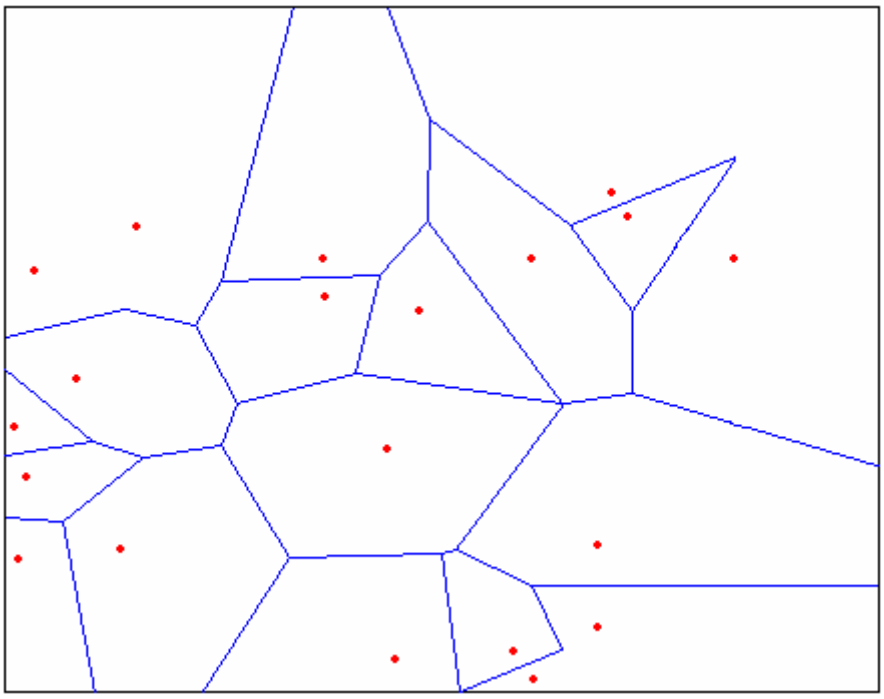


Şekil 4.2 RNG ve GG için Etki Bölgeleri

RNG ve GG algoritmalarının iki boyutlu veriler için oluşturulması gayet kolay olmasına rağmen çok boyutlu veriler için işlemler daha da zorlaşmaktadır (Jain ve Dubes, 1988).

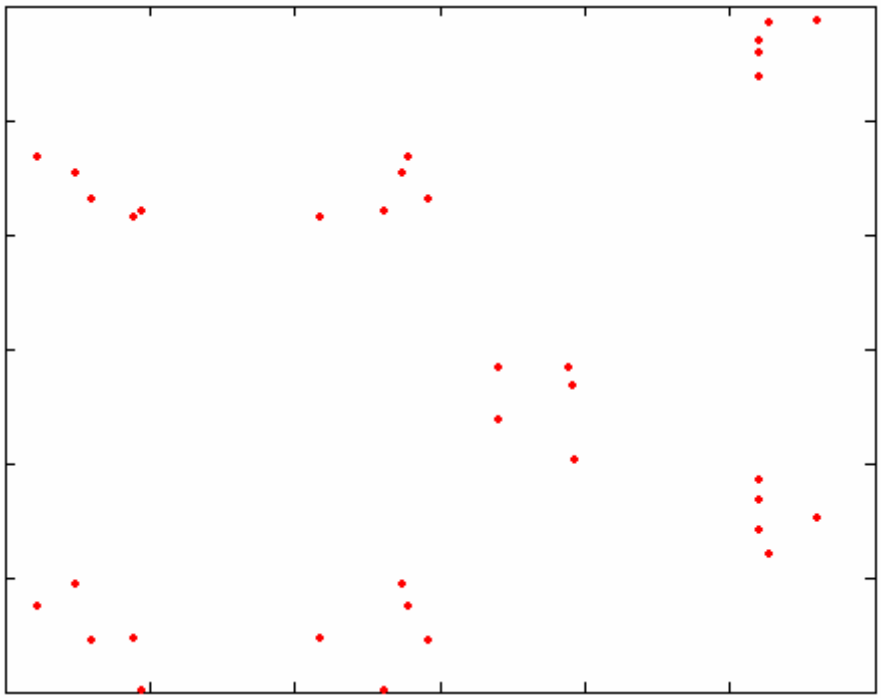
4.3 Delaunay Üçgen Metodu

Delaunay üçgen metodu (Delaunay Triangulation-DT) örnek analizinde kullanılan bir diğer çizge yapısıdır. DT'nin hesaplanabilmesi için etkili ve verimli birçok algoritma bulunmaktadır, ayrıca RNG ve GG, DT'de bulunan kenarlardan bazılarını silerek kolayca hesaplanabilmektedir. DT'nin en güzel tanımı ikili (dual) yapısı olan "Dirichlet Mozaikleri (Dirichlet Tessellation)"'nin açıklanması ile gerçekleştirilir. R^d (d boyutlu öklit uzayı)'de bulunan $\mathcal{H}$ gibi bir örnek dizisinin Dirichlet mozaiği (Şekil 4.3) ile kümelenmesi (aynı zamanda Voronoi diyagramı olarak da bilinmektedir), R^d'nin her $\mathbf{x}_i$ örnek vektörü civarında hücrelere bölünmesidir. Hücre sınırlarını, $\mathbf{x}_i$'yi diğer $(n-1)$ örnekle birleştiren çizgilerin dikey açıortaylarının kesişimleri belirlemektedir. Bu nedenle her hücre dışbükey bir çokgen şeklindedir.

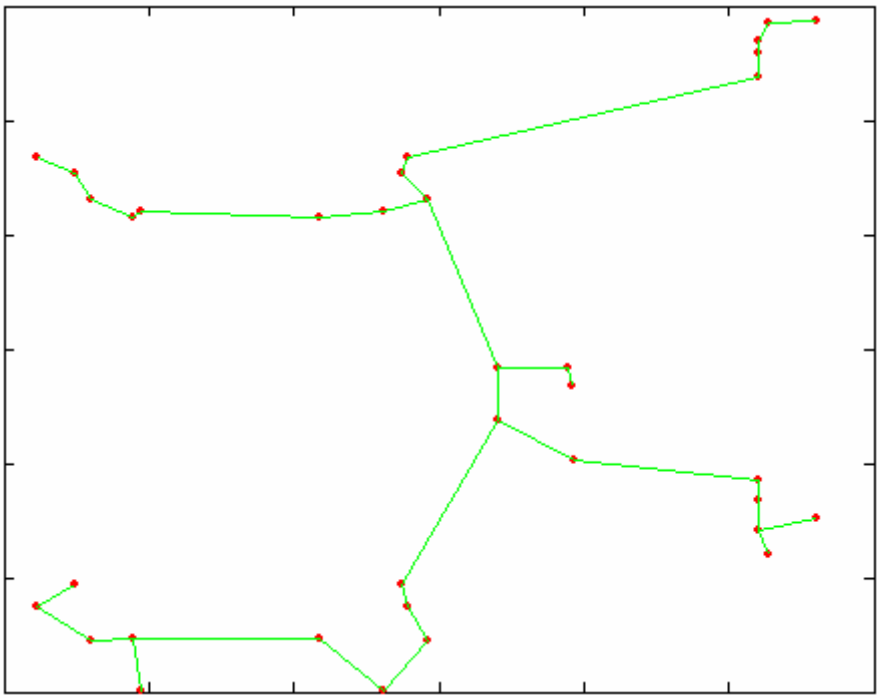


Şekil 4.3 Dirichlet Mozaiği (Voronoi Diyagramı)

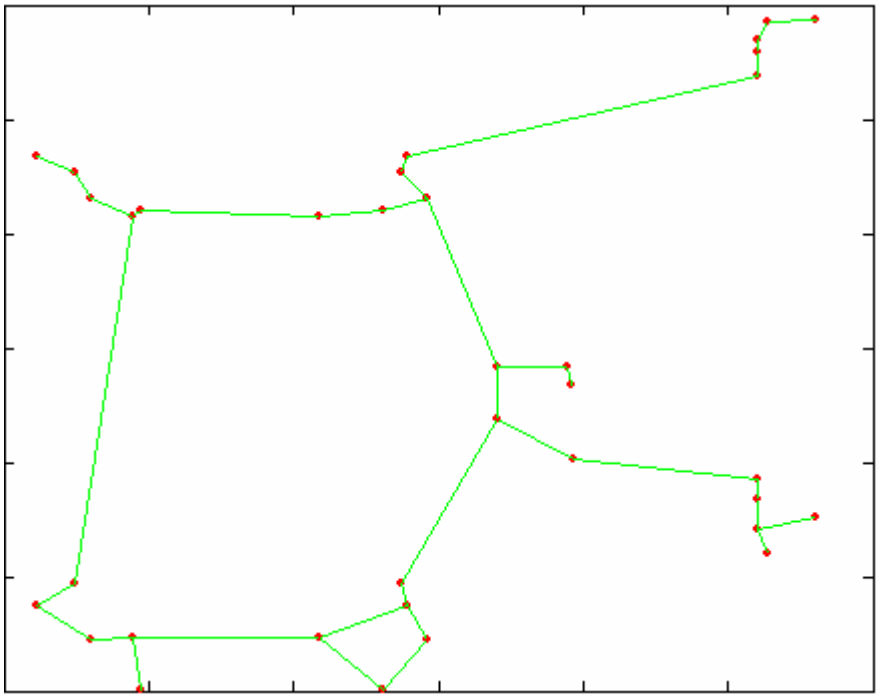
DT'nin bir diğer tanımı ise şu şekildedir. $\mathbf{x}_i$ ve $\mathbf{x}_j$ 'yi birleştiren kenar sadece, $\mathbf{x}_i$ ve $\mathbf{x}_j$ 'yi de içeren Dirichlet mozaiği ile oluşturulmuş iki hücre ortak bir sınırı paylaşıyorsa DT'ye dahildir. DT'nin sık olarak kullanıldığı uygulama alanları biyoloji (farklı kuş türlerinin üreme alanları) ve coğrafya (bölgelerin ağaç türlerine göre ayrılması)'dır. Fakat bu uygulamalar sadece iki boyutlu veriler için yapılmıştır. Aynı zamanda, MST'de olduğu gibi noktalar arasında bulunan kenarlar Voronoi diyagramından farklı olarak, eşik değerinden büyük kenarların kaldırılmasıyla da kümelenebilmektedir. Şekil 4.4'de görülen veri noktaları sırasıyla MST (Şekil 4.5), RNG (Şekil 4.6), GG (Şekil 4.7) ve DT (Şekil 4.8) için kümelere ayrılmıştır.



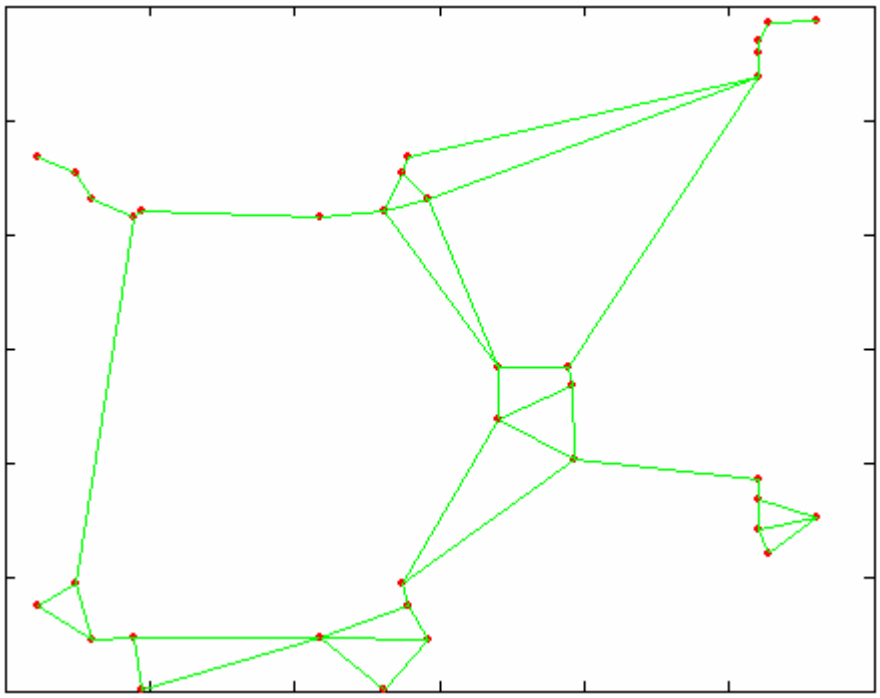
Şekil 4.4 Kümelenecek Olan Veri Noktaları



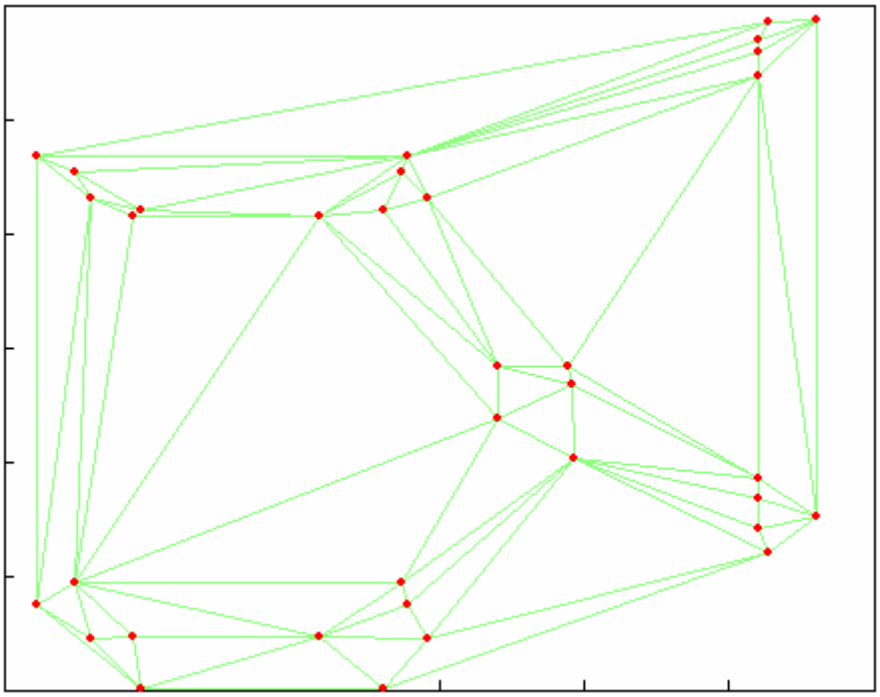
Şekil 4.5 MST



Şekil 4.6 RNG



Şekil 4.7 GG



Şekil 4.8 DT

RNG, GG ve DT tabanlı kümeleme algoritmaları ZAHN'ın MST tabanlı algoritmasına çok benzemektedir. Sadece bu algoritmada bulunan ilk adım, "Verilen *n*

örnek için MST oluşturulur", "Uygun çizge yapısı oluşturulur" şeklinde değiştirilmelidir. Çok fazla sayıda kenar içeren çizgelerde, uyuşmayan kenarın belirlenmesinde kullanılacak olan sezgisel yöntemin seçimi daha karmaşık hale gelmektedir. MST için bir kenarın kaldırılması her zaman iki küme oluşturmaktadır. Aynı durum RNG, GG ve DT için geçerli değildir çünkü iki nokta arasında birden fazla kenar (yol) bulunabilmektedir. RNG, GG ve DT uygulamaları, çok boyutlu durumlarda hesaplamaların karmaşıklaşmasından dolayı sınırlıdır.

Bazı bilim adamları MST yerine GG ve RNG'nin kullanımını daha uygun bulmaktadır. Bu sebeplerden ilki, RNG ve GG'nin örneklerin yerlerinin değişimine MST'ye göre daha az duyarlı olmalarıdır. İkincisi, GG ve RNG ile oluşturulan çizgeler MST ile oluşturulan çizgelerin alt çizgeleri olduğundan dolayı, GG ve RNG daha büyük birbirine bağlılık dereceleri göstermektedir ve böylece MST'ye göre küme yapılarının oluşturulmasında daha uygun olmaktadır. DT algoritması iki boyutlu örneklerin kümelenmesinde daha kullanışlıdır. Bunlara ek olarak unutulmamalıdır ki, algoritmaların performansları verilere bağlıdır (Jain ve Dubes, 1988).

4.4 En Yakın Komşu Kümeleme Algoritması

Kümelerin belirlenmesinde kullanılan en doğal yol, en yakın komşu (Nearest Neighbor) özelliğinin kullanılmasıdır. Bir örnek kendine en yakın komşusu ile aynı kümeye konulmaktadır. Eğer iki örnek aynı komşuyu paylaşıyorsa birbirine benzer olarak kabul edilir. Çizge kuramlı metotlar en yakın komşu özelliğiyle yakından ilişkilidir, bununla birlikte, kümelerin oluşturulma şekli ve en önemlisi en son paylaşımın belirlenmesi açısından birbirlerinden açık şekilde farklı özelliktedirler. $\mathcal{H} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ örnek dizisini K kümeye ayırmak amacıyla aşağıdaki adımlar yapılmalıdır. Kullanıcı en yakın komşuluk değeri için bir eşik değeri, t, belirlemelidir.

1) $i \leftarrow 1$ ve $k \leftarrow 1$ olarak belirlenir ve $\mathbf{x}_1$ örneği C_1 kümesine atanır.

2) $i \leftarrow i+1$ olarak belirlenir. Biraz önce kümelenen örneklerin arasından $\mathbf{x}_i$'nin en yakın komşusu bulunur. d_m ifadesi $\mathbf{x}_i$ ile en yakın komşusu arasındaki uzaklığı ifade etmektedir. En yakın komşunun m. kümede olduğu varsayılsın.

3) Eğer $d_m \leq t$ ise, $\mathbf{x}_i$ örneği C_m kümesine atanır. Değilse, $k \leftarrow k+1$ olarak belirlenir ve $\mathbf{x}_i$ örneği yeni oluşturulan C_k kümesine yerleştirilir.

4) Eğer tüm örnekler kümelere atanmışsa işlem bitirilir. Atama işlemi bitmemiş ise ikinci adıma dönülür.

Oluşturulan küme sayısı, K, t parametresinin bir fonksiyonudur ve t değeri arttıkça daha az küme oluşmaktadır.

4.5 Karşılıklı Komşuluk Değeri Kümeleme Algoritması

Karşılıklıl komşuluk değeri (Mutual Neighborhood Value-MNV) algoritması, *Karşılıklı Yakınlık (Mutual Nearness)* ile ifade edilen ve en yakın komşuluk algoritmasına benzer başka bir kümeleme algoritması da bulunmaktadır (Gowda ve Krishna, 1978). Bu algoritmada, $\mathbf{x}_j$, $\mathbf{x}_i$'nin p. yakın komşusu ve $\mathbf{x}_i$'de $\mathbf{x}_j$'nin q. yakın komşusu ise, $\mathbf{x}_i$ ve $\mathbf{x}_j$ arasındaki MNV değeri $(p+q)$ ile tanımlanır. MNV küçüldükçe örnekler birbirine daha benzer hale gelmektedir.

1) Her örnek için k tane en yakın komşular belirlenir.

2) Tüm örnekler için MNV hesaplanır. $\mathbf{x}_i$ ve $\mathbf{x}_j$ örnekleri verilen k değeri için karşılıklı komşu değillerse, $\text{MNV}(\mathbf{x}_i, \mathbf{x}_j)$ keyfi olarak seçilen büyük bir sayıya atanır.

3) MNV değeri 2 olan tüm örnek çiftleri belirlenir ve bu örnek çiftleri en küçük uzaklık değerine sahip çiftlerden başlanarak aynı kümede birleştirilir. Bu adım MNV değeri 3, 4, ..., $2k$ için tekrarlanarak hiyerarşi oluşturulur.

Komşulukları belirleyen k değeri algoritmanın performansında önemli rol oynamaktadır. Küçük k değerleri çeşitli güçlü kümelerin oluşmasına sebep olurken,

büyük k değerleri zayıf kümelerin oluşmasına sebep olmaktadır. k değeri, algoritmanın sonuç olarak tek bir küme oluşturabilmesi için her zaman yeterince büyük seçilmelidir (Jain ve Dubes, 1988)..

4.6 Destek Vektörleri İle Kümeleme

Kümelendirme metotları ve algoritmaları önceden belirlenmiş kurallara göre verileri gruplandırmaktadır. Destek vektörleri ile kümelendirme (Support Vector Clustering - SVC) işlemi yapılırken, elimizde bulunan veriler, uygun bir kernel fonksiyonu (Gaussian kernel fonksiyonu) ile farklı uzaya taşınmaktadır ve burada verilerimizi çevreleyen en küçük çaplı küreler hesaplanmaktadır. Bulunan küreler tekrar veri uzayımıza taşındığında verilerimiz çevreleyen dış hatlar ortaya çıkmaktadır. Bu dış hatlar, küme sınırları olarak adlandırılmaktadır. Bu hatlar üzerinde kalan veriler de destek vektörleri olmaktadır.

$\mathbf{x}_i$; $i = 1,2,...,l$ veri seti, lineer olmayan Φ dönüşümü ile daha büyük boyutlu öznitelik uzayına taşınmaktadır ve burada en küçük R yarıçaplı küre aranmaktadır. Problem aşağıdaki gibi tanımlanmaktadır:

$$\begin{aligned} &\textit{Amaç Fonksiyonu} \quad \min R^2 \\ &\textit{Kısıtlar} \quad \left\| \Phi(\mathbf{x}_i) - a \right\|^2 \le R^2 \;;\; i = 1,2,...,l \end{aligned} \tag{4.3}$$

a değeri kürenin merkezini ifade etmektedir. Probleme gevşek değişkenler eklenirse aşağıdaki problem elde edilir.

$$\begin{aligned} &\textit{Amaç Fonksiyonu} \quad \min R^2 \\ &\textit{Kısıtlar} \quad \left\| \Phi(\mathbf{x}_i) - a \right\|^2 \le R^2 + \xi_i \;;\; \xi_i > 0 \;;\; i = 1,2,...,l \end{aligned} \tag{4.4}$$

Lineer olmayan en iyileme probleminin çözümü için Lagrange fonksiyonu tanımlanmaktadır:

$$L = R^2 - \sum_{i=1}^{l}\left(R^2 + \xi_i - \left\|\Phi(\mathbf{x}_i) - a\right\|^2\right)\beta_i - \sum_{i=1}^{l}\xi_i\mu_i + C\sum_{i=1}^{l}\xi_i \tag{4.5}$$

$\beta_i \geq 0$ ve $\mu_i \geq 0$ katsayıları lagrage katsayılarını ifade etmektedir. C katsayısı bir sabittir ve problemi hatalara karşı biraz daha esnek olabilmesi için eklenmiştir. Birinci dereceden koşulları bulabilmek amacıyla lagrange fonksiyonunun sırasıyla R, a ve $\boldsymbol{\xi}_i$'ye göre türevleri aşağıda görülmektedir.

$$\frac{\partial L}{\partial R} = 0 \;\Rightarrow\; 2R - \sum_{i=1}^{l} 2R\beta_i = 0 \;\Rightarrow\; 2R\left(1 - \sum_{i=1}^{l}\beta_i\right) = 0 \;\Rightarrow\; \sum_{i=1}^{l}\beta_i = 1 \tag{4.6}$$

$$\frac{\partial L}{\partial a} = 0 \;\Rightarrow\; a = \sum_{i=1}^{l}\beta_i\Phi(\mathbf{x}_i) \tag{4.7}$$

$$\frac{\partial L}{\partial \xi} = 0 \;\Rightarrow\; \beta_i = C - \mu_i \tag{4.8}$$

Bunlara ek olarak KKT (Karush Kuhn Tucker) tamamlayıcı koşulları da göz önünde bulundurulmalıdır (Venkataraman, 2001).

$$\xi_i\mu_i = 0 \tag{4.9}$$

$$\left(R^2 + \xi_i - \left\|\Phi(\mathbf{x}_i) - a\right\|^2\right)\beta_i = 0 \;;\; i = 1, 2, ..., l \tag{4.10}$$

(4.10) denkleminden, herhangi bir $\mathbf{x}_i$ noktasının öznitelik uzayındaki görüntüsü, $\beta_i \geq 0$ ve $\xi_i \geq 0$ durumunda kürenin dışında kalmaktadır. (4.9) denkleminde eğer $\mu_i = 0$ alınırsa ve (4.8)'de yazılırsa $\beta_i = C$ elde edilmektedir. Bu durumları sağlayan noktalar *Sınırlandırılmış Destek Vektörleri* (Bounded Support Vectors-BSV) olarak adlandırılmaktadır. Eğer, $\xi_i = 0$ değerini alırsa, $\mathbf{x}_i$ noktasının öznitelik uzayındaki görüntüsü kürenin içinde veya sınırında kalmaktadır. Ayrıca, $0 < \beta_i < C$ durumunda (4.9) denkleminden yola çıkarak, $\mathbf{x}_i$ noktasının öznitelik uzayındaki görüntüsü $\Phi(\mathbf{x}_i)$, kürenin yüzeyinde bulunacaktır. Bu koşulu sağlayan noktalar ise *Destek Vektörleri*

(Support Vectors-SV) olarak adlandırılmaktadır. Eğer $C \geq 1$ olursa, (4.6) denklemine göre herhangi bir BSV bulunmamaktadır.

Bu açıklamalardan yola çıkarak R, a ve μ_i değişkenlerini elenebilir ve lagranj fonksiyonu, β_i değişkenlerinin bir fonksiyonu olarak tekrardan yazılabilir. Bu durumdaki lagranj fonksiyonu, *Wolfe İkincil Form* olarak adlandırılmaktadır.

$$W = \sum_{i=1}^{l} \Phi(\mathbf{x}_i)^2 \beta_j - \sum_{i,j=1}^{l} \beta_i \beta_j \Phi\left(\mathbf{x}_i\right) \cdot \Phi\left(\mathbf{x}_j\right) \tag{4.11}$$

(4.11) denkleminde bulunan iç çarpım yerine uygun bir kernel (Mercer Kernel) fonksiyonu yazıldığında, denklem aşağıdaki şekilde değişmektedir. Burada Gaussian kernel fonksiyonu kullanılmıştır. q değeri, fonksiyonun genişliğini ifade etmektedir ve eşik değeri yerine kullanılmaktadır.

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = e^{-q\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \tag{4.12}$$

$$W = \sum_{i=1}^{l} K(\mathbf{x}_i, \mathbf{x}_i)\beta_j - \sum_{i,j=1}^{l} \beta_i \beta_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) \tag{4.13}$$

Herhangi bir $\mathbf{x}_i$ noktası için kürenin yarıçapı aşağıdaki denklem ile hesaplanmaktadır.

$$R^2\left(\mathbf{x}\right) = \left\|\Phi\left(\mathbf{x}_i\right) - a\right\|^2 \tag{4.14}$$

(4.7) denklemi ve bölüm 2'deki kernel tanımından yol çıkarak yarıçap ifadesi tekrar yazılırsa,

$$R\left(\mathbf{x}\right) = K(\mathbf{x}, \mathbf{x}) - 2\sum_{i=1}^{l} \beta_j K(\mathbf{x}_j, \mathbf{x}) - \sum_{i,j=1}^{l} \beta_i \beta_j K\left(\mathbf{x}_i \mathbf{x}_j\right) \tag{4.15}$$

elde edilmektedir. Kürenin yarıçapı,

$$R(\mathbf{x}) = \left\{ R(\mathbf{x}_i) \,|\, \mathbf{x}_i \text{ noktası bir Destek Vektörü'dür} \right\} \tag{4.16}$$

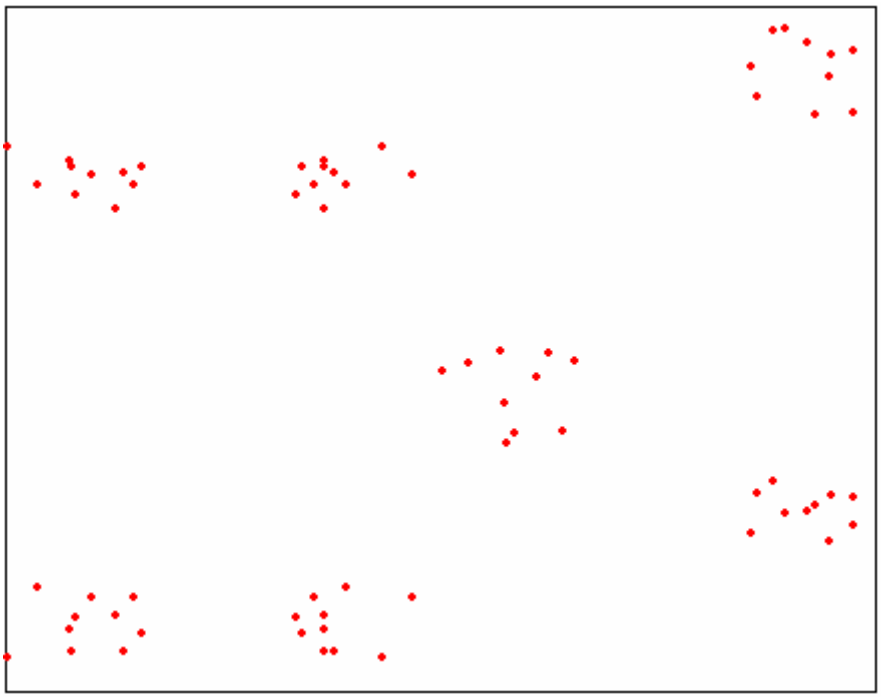
Veri uzayında bulunan noktaları çevreleyen kürenin dış hatlarını belirleyen set ise,

$$\left\{ \mathbf{x} \,|\, R(\mathbf{x}) = R \right\} \tag{4.17}$$

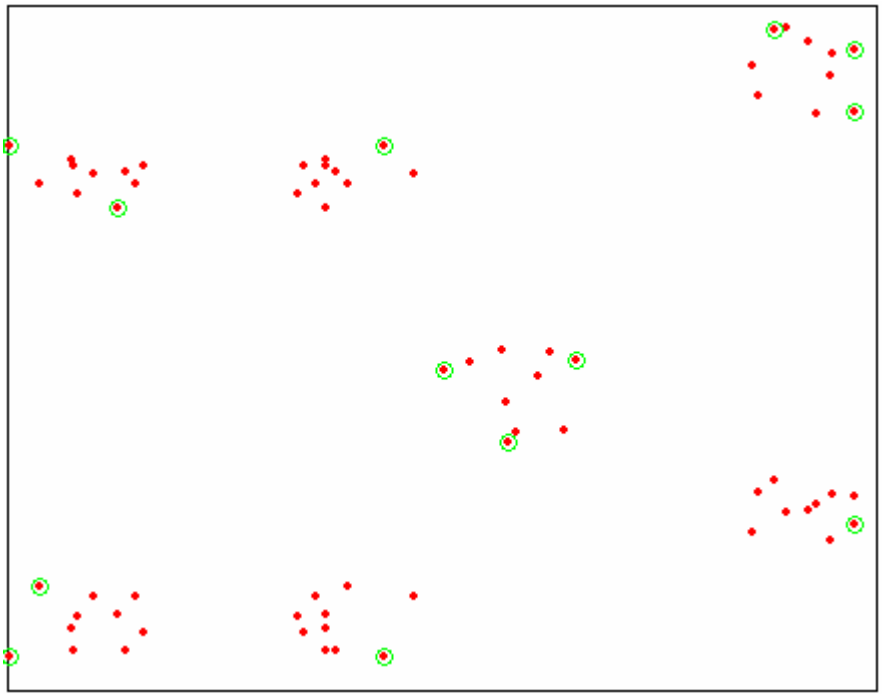
(4.16) ifadesine göre, SV'ler küme sınırı üzerinde, BSV'ler küme dışında ve diğer tüm noktalar küme içerisinde bulunmaktadır. Aşağıdaki şekillerde sırasıyla dağılmış durumda olan veriler, SV'ler ve kümelendirilmiş veriler görülmektedir.

Seçilen kernel fonksiyonuna göre SVM'nin de yapısı değişmektedir. Yapay sinir ağlarında farklı görevler için farklı mimariler seçilirken SVM'lerde ise farklı görevler için farklı kernel fonksiyonları seçilmektedir. Aşağıdaki şekillerde saçılmış veriler ve bunların destek vektörleri ile kümelenmesi gösterilmektedir.

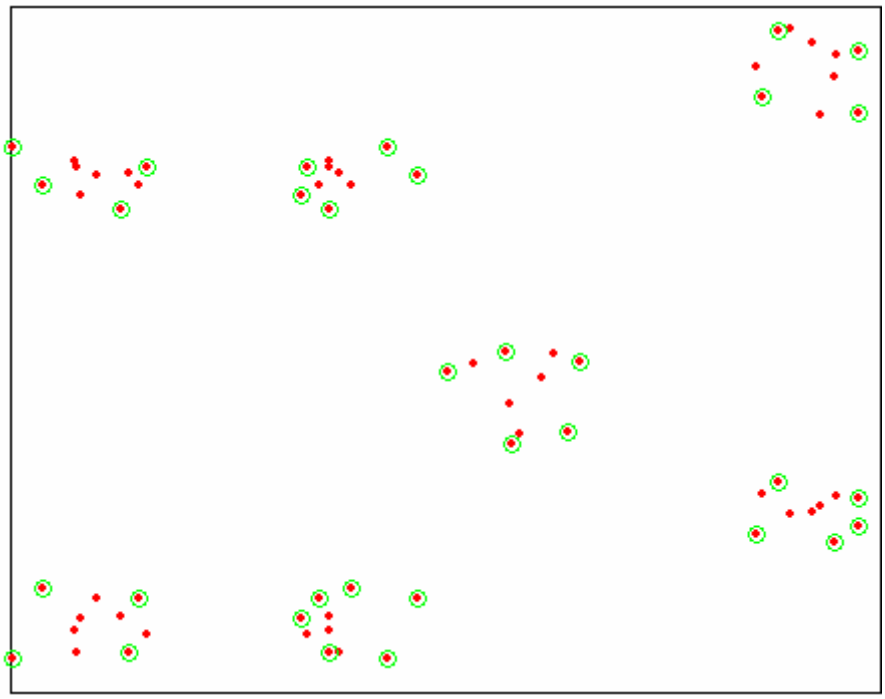
Şekil 4.10'da, $C = 1$ alınmıştır ve kernel fonksiyonunda bulunan q değeri sırasıyla, 0.3 (a) 3 (b) ve 10 (c) olarak alınmıştır. SV'ler daire içinde gösterilmiştir ve q değerini artırdığımız zaman SV sayısı artmakta böylece daha belirgin kümeler ortaya çıkmaktadır. Burada $C = 1$ olması nedeniyle BSV bulunmamaktadır.



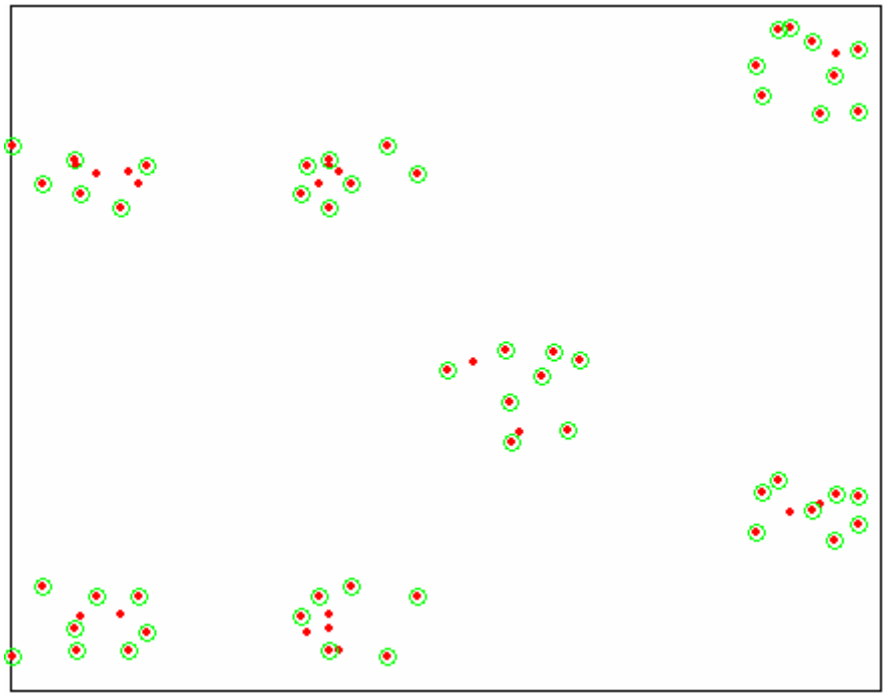
Şekil 4.9 Saçılmış Durumda Bulunan Veriler



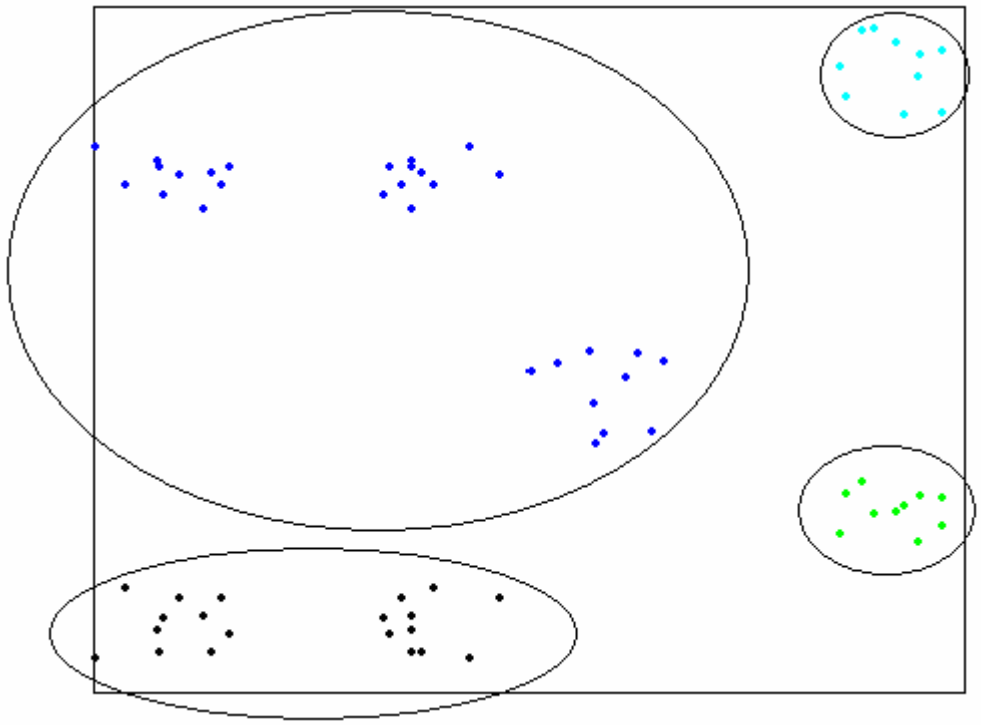
Şekil 4.10 (a) $q = 0.3$ İçin Destek Vektörleri



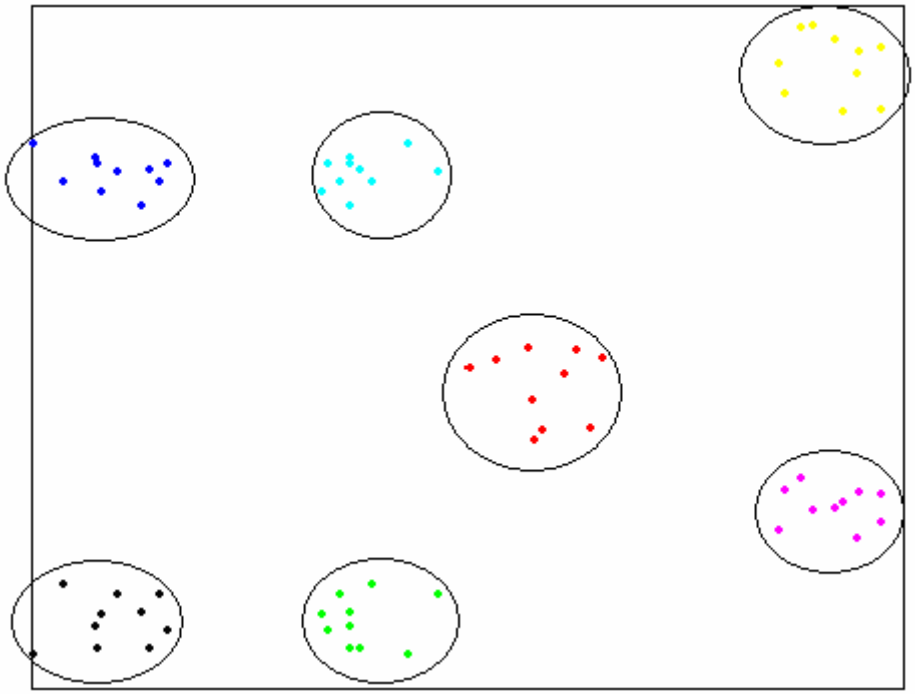
Şekil 4.10 (b) $q = 3$ İçin Destek Vektörleri



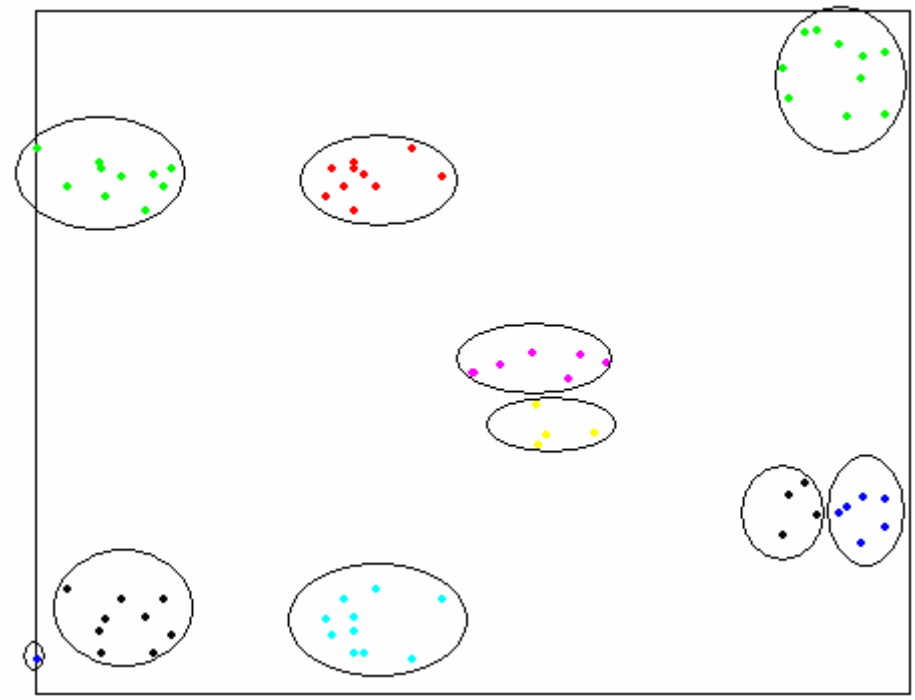
Şekil 4.10 (c) $q = 10$ İçin Destek Vektörleri



Şekil 4.11 (a) $q = 0.3$ Kümelere Ayrılmış Veriler (4 küme)



Şekil 4.11 (b) $q = 3$ Kümelere Ayrılmış Veriler (7 küme)



Şekil 4.11 (c) $q = 10$ Kümelere Ayrılmış Veriler (10 küme)

Kümelendirme algoritması farklı kümeler ait olan noktaları ayırt edememektedir. Bu ayırımı yapabilmek amacıyla $R(\mathbf{x})$ (küme sınırları) ifadesini de kapsayan bir yaklaşım kullanılmıştır. Bu yaklaşıma göre, "*farklı kümelere ait veri çiftlerini birleştiren herhangi bir yol* (path)*, öznitelik uzayında kürenin dışında kalmalıdır*" (Vapnik ve diğerleri, 2001). Bu nedenle, *yol* (path), $R(\mathbf{y}) > R$ gibi $\mathbf{y}$ veri setinin bir bölümünü içermektedir. Bu tanımlamalar ışığında, öznitelik uzayındaki görüntüleri kürenin içinde veya üzerinde bulunan $\mathbf{x}_i$ ve $\mathbf{x}_j$ veri çiftleri arasında $\mathbf{A_{ij}}$ komşuluk matrisi tanımlanmıştır.

$$\mathbf{A_{ij}} = \begin{cases} 1 \; ; y \text{ değerleri } x_i \text{ ve } x_j \text{ çiftlerini birleştiren doğru parçası üzerinde ise} \\ 0 \; ; \text{aksi halde} \end{cases}$$

Bu komşuluk matrisin tanımı ile kümeler birbirine bağlanmış bileşenler olarak tanımlanmıştır. Bu matris tanımı ile BSV'ler sınıflandırılmamış olur çünkü öznitelik uzayında BSV'lerin görüntüsü küme sınırlarının dışında kalmaktadır. Sınıflandırma yapmadan bırakılabileceği gibi en yakın olduğu kümeye de dahil edilebilirler.

Yukarıda tanımlandığı üzere, veri uzayındaki küme sınırlarının şekli, iki parametreye bağlıdır. q, Gaussian kernel fonksiyonun genişlik ifadesi ve C, yumuşatma sabiti. q değerinin arttırılması ile destek vektör sayısı artmaktadır (Horn ve diğerleri, 2000). Tezde yapılan kümeleme işlemlerinde her zaman $C = 1$ seçilmiştir.

BEŞİNCİ BÖLÜM
BENZETİM SONUÇLARI

5. BENZETİM SONUÇLARI

Bu bölümde benzetimlerde kullanılan veri setleri ve farklı algoritmalara göre yapılan kümeleme işlemlerinin sonuçları yer almaktadır.

Benzetimlerde kullandığımız ilk veri seti literatürde İris (Süsen, zambak gibi kılıç şeklinde yaprakları olan ve gösterişli çiçeklerden oluşan bir grup) çiçeği olarak adlandırılan ve 1935 yılında E. Anderson tarafından "*Bulletin of the American Iris Society*" dergisinde yayımlanan "*The irises of the Gaspé peninsula*" adlı makalede sunulmuştur. Üç farklı türde (Setosa, Versicolor, Virginica) iris çiçeğinden 50'şer örnek alınmıştır (Şekil 5.1, Şekil 5.2, Şekil 5.3), bu örnekler cm olarak çiçeklerin çanakyaprak uzunluğu, çanakyaprak genişliği, taçyaprak uzunluğu ve taçyaprak genişliği ölçümlerinden oluşan dört boyutlu bir veri setidir. Bu veri seti daha sonra 1936 yılında R. A. Fisher tarafından, "*Annals of Eugenics*" dergisinde "*The use of multiple measurements in taxonomic problems*" isimli makalesinde sunduğu doğrusal ayırım fonksiyonu tekniğinin başlatılmasında kullanılmıştır. Aşağıdaki resimlerde iris çiçeklerine ait örnekler görülmektedir. Şekil 5.5'de farklı iki özelliğine göre oluşturulmuş noktalar görülmektedir.

Şekil 5.1 İris Setosa

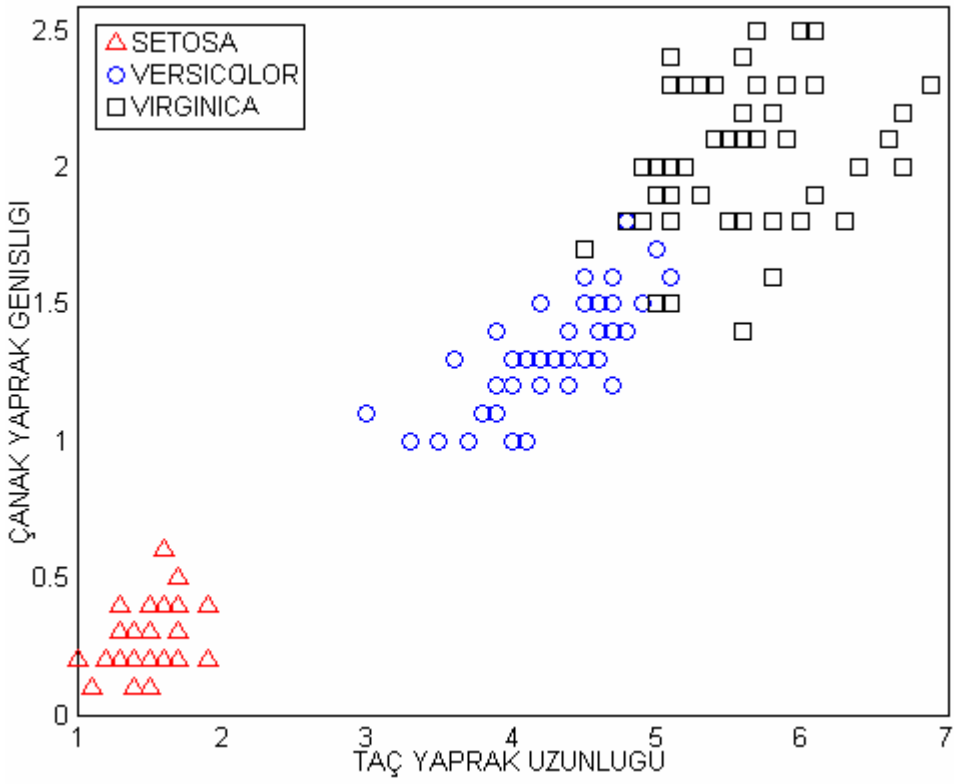
Şekil 5.2 İris Versicolor

Şekil 5.3 İris Virginica

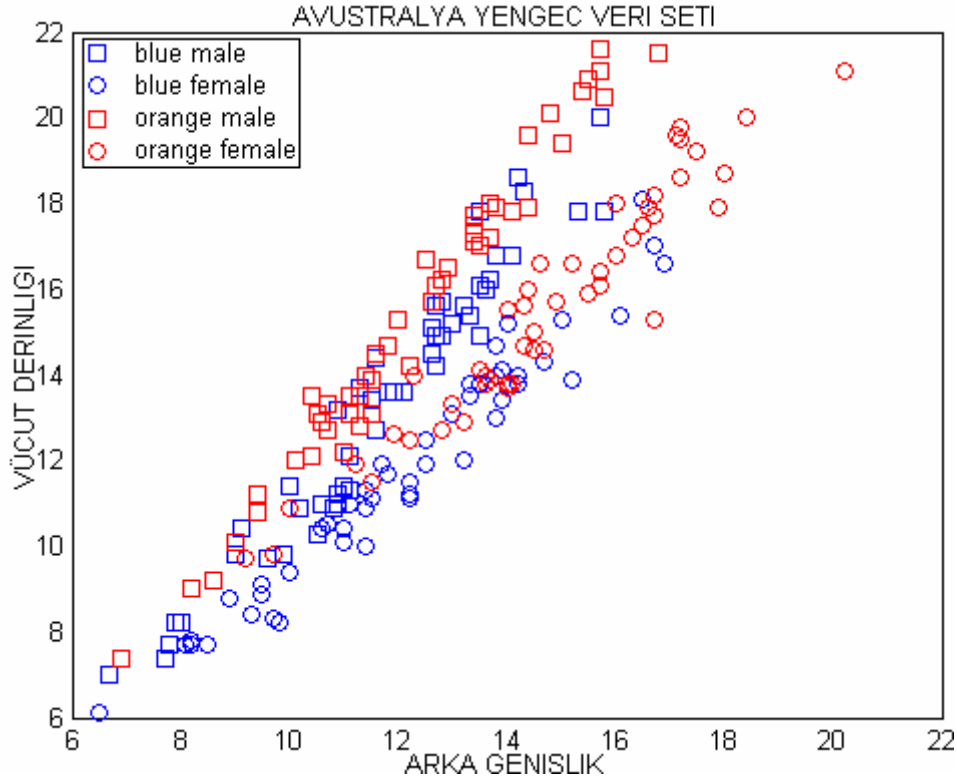
Kümeleme işlemlerinde kullanılan diğer bir veri seti de B.D. Ripley'in 1996'da yayınladığı "*Pattern Recognition and Neural Networks*" isimli eserinde bulunan Avustralya kaya yengeçlerinin (Leptograpsus, Şekil 5.4) farklı özelliklerine göre ölçüm değerlerinin bulunduğu beş boyutlu bir veri setidir.

Şekil 5.4 Avustralya Kaya Yengeci
(http://www.seafriends.org.nz/enviro/crust/grapsida.htm)

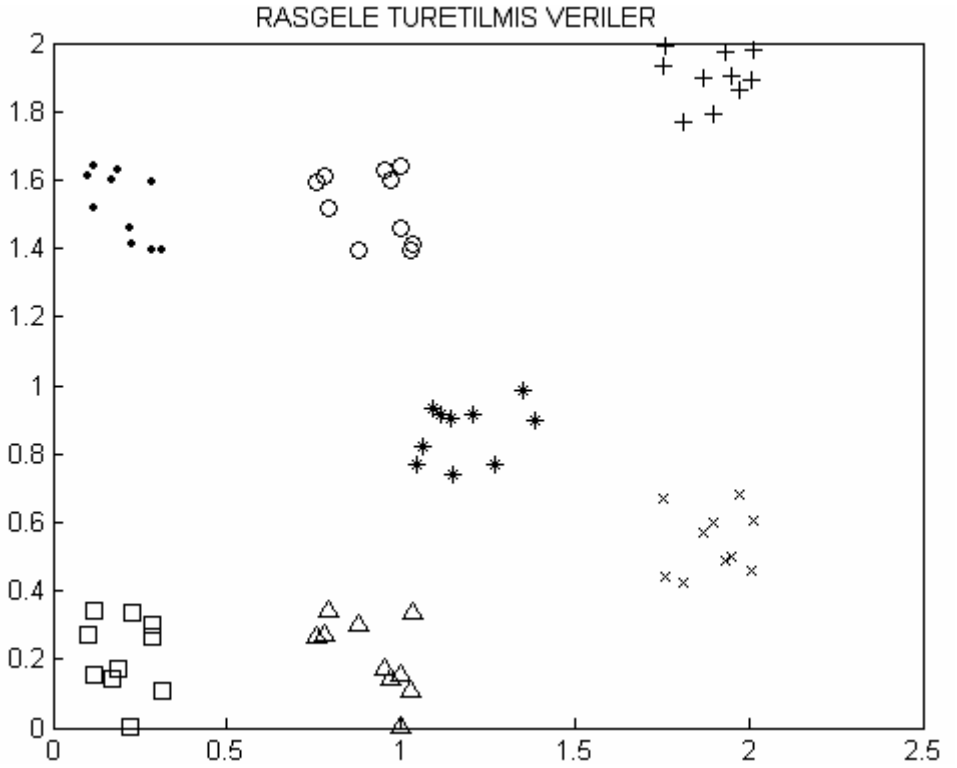
Bu yengeçler mavi ve turuncu olmak üzere iki farklı renkte bulunmaktadır. Ön lob uzunluğu, kabuk uzunluğu, kabuk genişliği, vücut derinliği ve arka genişliklerinin ölçümlerinden oluşan veri setinde mavi-dişi, turuncu-dişi, turuncu-erkek ve mavi-erkek'lerden 50'şer tane örnek alınmıştır. Şekil 5.6'da farklı iki özelliğine göre oluşturulmuş noktalar görülmektedir. Bu iki değerlendirme (benchmark) veri seti dışında MATLAB ortamında rasgele oluşturulmuş üçüncü bir veri seti de (DATA) karşılaştırmalarda kullanılmıştır. Bu veri seti Şekil 5.7'de görülmektedir. Gerçeklenen algoritmalarda kullanılan verieler bu işlemlerden önce normalize edilmiştir.



Şekil 5.5 Iris Veri Seti



Şekil 5.6 Avustralya Kaya Yengeci Veri Seti



Şekil 5.7 Rasgele Oluşturulmuş Veriler

5.2. Algoritma Benzetim Sonuçları

Yedi farklı algoritma, tüm veri setleri için, altı farklı eşik değeri için çalıştırılmış ve aşağıdaki sonuçlar elde edilmiştir. Diğer algoritmalardan farklı olarak, SVC ve MNV algoritmalarında eşik değeri yerine sırasıyla q ve en yakın komşuluk değerleri kullanılmıştır. Eşik değerleri veri setlerine göre oluşturulan yakınlık matrisinden elde edilen en fazla, en az ve ortalama yakınlık değerlerine göre her veri seti için ayrı ayrı belirlenmiştir. İkinci adım olarak, veri setlerine işaret gürültü oranı (SNR) 24Db ve 45Db olan gürültü işaretleri eklenmiş ve bu durum için kümeleme işlemleri tekrarlanmıştır. Iris veri seti 4800 bayt, Crab veri seti 8000 bayt ve diğer rasgele oluşturulmuş veri seti 1120 bayt yer kaplamaktadır.

5.2.1 En Yakın Komşu Algoritması için Sonuçlar

En yakın komşu algoritmasının diğer algoritmalara göre süre bakımından ve flops sayısı bakımından daha hızlı olduğu görülmektedir. Bunun nedeni, noktaların herhangi bir yapıya ihtiyaç duyulmadan kümelere ayrılmasıdır. Seçilen büyük eşik değerleri için tüm noktalar tek bir kümeye dahil edilirken eşik değerinin azaltılmasıyla küme sayısının arttığı görülmektedir. Ayrıca algoritma, veri setlerine eklenen gürültülerden çok az oranda etkilenmektedir. Bu tez çalışmasında gerçeklenen algoritmalar arasında, tüm özellikler göz önünde bulundurulduğunda En Yakın Komşuluk Algoritması en verimli algoritmadır.Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.1'de görülmektedir:

Çizelge 5.1 En Yakın Komşu Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.1	0.735	89867	1	1	1
0.05	0.735	89867	1	1	1
0.02	0.734	89868	2	2	1
0.015	0.733	89869	3	3	3
0.01	0.720	89875	9	9	7
0.008	0.719	89881	15	15	14
CRAB	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.05	1.688	159817	1	1	1
0.02	1.688	159817	1	1	1
0.01	1.1689	159819	3	3	3
0.007	1.69	159821	5	4	6
0.006	1.7	159824	8	8	9
0.005	1.703	159830	14	14	15
DATA	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.5	0.109	19547	1	1	1
0.1	0.109	19548	2	2	2
0.08	0.109	19549	3	3	2
0.065	0.109	19550	4	4	5
0.04	0.109	19553	7	7	7
0.02	0.11	19557	11	11	11

5.2.2 En Küçük Tarama Ağacı (MST) Algoritması için Sonuçlar

Sonuçlardan da görüleceği gibi flops sayısı en yakın komşuluk algoritmasına nazaran daha fazladır bunun sebebi çizge yapılarının önce oluşturulup daha sonra kümeleri oluşturmak amacıyla parçalanmasından kaynaklanmaktadır. En yakın komşuluk algoritmasında ise veriler, herhangi bir yapı oluşturulmadan direk en yakın komşunun bulunduğu kümeye dahil edilmektedir. Küme sayısı, seçilen eşik değerinin azalması ile burada da artmaktadır. Crab ve IRIS veri setleri için küme sayısı, çok küçük eşik değerleri için aşırı derecede artmaktadır, bunun sebebi ise bu veri setlerinde bulunan noktaların özniteliklerinin birbirlerine çok yakın olmasından kaynaklanmaktadır. Veri setlerine eklenen gürültülerden, verilerin kesin sınırlarının olduğu setlerde (DATA) çok fazla etkilenmemekte, iç içe girmiş verilerin olduğu setlerde ise gürültü oranı arttıkça küme sayısı da artmaktadır. Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.2'de görülmektedir:

Çizelge 5.2 En Küçük Tarama Ağacı Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.1	10.969	367879	1	1	1
0.05	5.64	360979	1	1	1
0.02	2.141	355836	3	3	4
0.015	1.437	355570	6	6	7
0.01	0.812	35772	29	25	30
0.008	0.672	362490	49	50	60
CRAB	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.05	20.516	770996	1	1	1
0.02	7.578	754420	1	1	1
0.01	2.484	747614	2	2	3
0.007	1.485	751257	10	9	13
0.006	1.281	751213	24	23	20
0.005	1.141	753814	44	45	57
DATA	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.5	1.562	52673	1	1	1
0.1	0.547	49665	1	1	1
0.08	0.391	49139	1	1	1
0.065	0.281	48490	2	2	2
0.04	0.219	48699	5	4	4
0.02	0.188	48451	7	7	8

5.2.3 Delaunay Üçgen (DT) Algoritması için Sonuçlar

Çok karmaşık bir çizge yapısına sahip olduğundan dolayı flops sayısı çok fazladır, MST algoritmasında olduğu gibi önce tüm çizge oluşturulmakta ve daha sonra eşik değerine göre alt çizgelere parçalanmaktadır. Fakat kümeler oluşturulurken dikkat edilmesi gereken nokta, aynı veri üzerinde birden fazla yol olduğundan dolayı, MST algoritmasında DATA veri setinde 0.02 eşik değeri için 4 küme oluşurken, DT algoritması için 3 küme oluşmaktadır. Boyut arttıkça işlemsel karmaşıklık da artmaktadır. Asıl kullanım alanları biyoloji ve coğrafya gibi bilim dallarıdır. Böyle bir çalışma, 1978 yılında Howe S.E. tarafından "Seçilen bir bölgenin ağaç tipine göre parçalara ayrılması amacıyla göl tortularında bulunan polenlerin gözlemlenmesi" şeklinde bir uygulama gerçekleştirmiştir (Jain ve Dubes, 1988). Literatürde bulunan uygulamaların neredeyse tümü iki boyutlu veriler için yapılmıştır (bu nedenle IRIS ve CRAB veri setlerinin 3. ve 4. öznitelikleri göz önünde bulundurulmuştur). Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.3'de görülmektedir:

Çizelge 5.3 Delaunay Üçgen Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.1	14.89	1660445	1	1	1
0.05	7.875	1128141	1	1	1
0.02	2.734	797918	2	2	2
0.015	2.031	659366	3	3	3
0.01	1.328	557880	6	6	7
0.008	1	513515	8	8	11
CRAB	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.05	25.969	3751299	1	1	1
0.02	11.079	2260337	1	1	1
0.01	5.313	1544031	2	2	2
0.007	3.532	1296177	2	2	2
0.006	2.984	1220427	2	2	2
0.005	2.5	1148111	2	2	2
DATA	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.5	2.094	232125	1	1	1
0.1	0.532	129309	1	1	1
0.08	0.39	111453	1	1	1
0.065	0.281	95896	2	2	2
0.04	0.204	81867	3	3	3
0.02	0.141	77829	5	5	5

5.2.4 Bağıl Komşuluk Değeri (RNG) Algoritması için Sonuçlar

RNG algoritmasına göre oluşan küme sayısının, DT algoritmasına göre daha fazla olmasının sebebi, RNG çizge yapısının DT çizge yapısına göre daha az karmaşık olmasıdır. Bu nedenle flops sayısı ve işlem süresi daha azdır. Aynı veri kümesi için, DT algoritmasına göre oluşturulan bir çizgeden, belli kenarlar silinerek RNG çizge yapıları oluşturulabilmektedir. Bu algoritma ile yapılan uygulamaların çoğu, işlemlerin karmaşıklığından dolayı iki veya üç boyutlu veriler için yapılmıştır. Tüm çizge yapıları (DT, GG, RNG, MST) gürültü eklenmiş veriler için birbirine benzer kümeleme oranlarına sahiptir. Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.4'de görülmektedir:

Çizelge 5.4 Bağıl Komşuluk Değeri Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.1	10.719	1592424	1	1	1
0.05	5.5	1065896	1	1	1
0.02	2.032	639225	3	3	3
0.015	1.313	537739	6	6	7
0.01	0.625	434425	18	18	23
0.008	0.437	408551	35	34	41
CRAB eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.05	23.375	3517510	1	1	1
0.02	8.203	1877270	1	1	1
0.01	2.484	1106424	2	2	2
0.007	1.219	913042	7	7	9
0.006	0.922	868583	10	9	13
0.005	0.687	829939	24	23	20
DATA eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.5	1.593	218424	1	1	1
0.1	0.485	110856	1	1	1
0.08	0.344	92064	1	1	1
0.065	0.218	77299	2	2	2
0.04	0.156	67872	7	7	4
0.02	0.109	59808	7	7	7

5.2.5 Gabriel Çizge (GG) Algoritması için Sonuçlar

E ifadesi çizgelerde bulunan toplam kenar sayısı olduğunu kabul ederek aşağıdaki ifade yazılabilir:

$$E(MST) < E(RNG) < E(GG) < E(DT) \tag{5.1}$$

(5.1) denkleminden de görüldüğü gibi çizge yapıları arasındaki karmaşıklık, DT çizge yapısında en fazladır, çizge yapılarında bulunan karmaşıklık azaldıkça flops sayısı azalmakta ve daha belirgin kümelere daha büyük eşik değerleri için ulaşılabilmektedir. 4. bölümde bulunan Şekil 4.5 (MST), Şekil 4.6 (RNG), Şekil 4.7 (GG) ve Şekil 4.8 (DT)'de çizge yapıları görülmektedir. Boyut arttıkça hem çizge yapısının oluşturulması hem de oluşturulan çizge yapısından uyuşmayan kenarların belirlenerek kaldırılması ve kümelerin oluşturulması zorlaştığından dolayı, bu algoritmalar genellikle iki veya üç boyutlu veriler için uygulanmaktadır. Her ne kadar durum böyle olsa da tüm

algoritmalar veriye bağlıdır. Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.5'de görülmektedir:

Çizelge 5.5 Gabriel Çizge Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
eşik değerleri	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
0.1	10.875	1610816	1	1	1
0.05	5.594	1082920	1	1	1
0.02	2.328	679865	2	2	2
0.015	1.563	575622	4	4	5
0.01	0.875	475158	14	13	15
0.008	0.625	434425	18	18	23
CRAB	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.05	24.75	3602350	1	1	1
0.02	9.375	2020084	1	1	1
0.01	3.5	1254490	2	2	2
0.007	2.016	1037340	2	2	3
0.006	1.609	971593	4	4	3
0.005	1.219	913042	7	7	9
DATA	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
eşik değerleri				**45Db**	**24Db**
0.5	1.594	218424	1	1	1
0.1	0.5	112440	1	1	1
0.08	0.359	94080	1	1	1
0.065	0.234	79315	2	2	2
0.04	0.156	68020	5	5	3
0.02	0.125	61248	7	7	7

5.2.6 Karşılıklı Komşuluk Değeri (MNV) Algoritması için Sonuçlar

Karşılıklı komşuluk değeri algoritması, en yakın komşu algoritmasının biraz değiştirilmiş şeklidir, bu algoritmada eşik değeri yerine en yakın komşuluk değerlerine göre kümeler belirlenmektedir. Her veri için belirlenen değere göre en yakın komşuların bulunması, belirlenen en yakın komşu sayısından küçük olan değerlerin elenmesi ve daha büyük olan değerlerin başka bir değere atanması, seçilen MNV değerine göre kümelerin oluşturulması gibi işlemler bulunduğundan işlem süresi oldukça fazladır. Bu nedenle çok fazla uygulama alanı bulunmamaktadır. Verilerin belli oranlarda bozulmasına rağmen karşılıklı komşuluk değerlerinde çok fazla değişiklik olmamasından dolayı küme sayısı aynı kalmaktadır. Flops sayısı diğer algoritmalara nazaran azdır, mesela CRAB veri setine 24DB gürültü eklendiğinde, MNV değeri 130

seçilirse işlem süresi 10 dakikaya yaklaşmaktadır. En yakın komşu algoritmasına göre daha uzun işlem süresine ve flops sayısına sahiptir. Bu nedenle yapılan uygulamalarda en yakın komşu algoritması tercih edilmektedir. Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.6'de görülmektedir:

Çizelge 5.6 Karşılıklı Komşuluk Değeri Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
en yakın komşu sayısı	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
20	8.797	354038	14	10	10
30	18.64	361436	16	12	11
50	55.422	374634	16	12	11
70	107.219	387930	16	12	12
90	185.203	401199	16	12	12
120	357.875	543064	16	12	12
CRAB	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
en yakın komşu sayısı				**45Db**	**24Db**
10	30.469	855295	7	7	9
20	47.031	870898	8	10	13
40	131.359	901365	10	10	13
60	259.328	933051	10	10	13
90	439.484	979642	10	10	13
130	754.797	1041699	10	10	13
DATA	**Süre (sn.)**	**Flops**	**Küme Sayısı**	**Küme Sayısı**	
en yakın komşu sayısı				**45Db**	**24Db**
5	0.672	62195	4	4	3
10	1.344	65439	6	6	5
15	2.344	68437	6	6	5
20	3.766	71342	6	6	6
30	7.812	77207	7	6	6
50	20.563	88959	7	7	6

5.2.7 Destek Vektörleri (SVC)Algoritması için Sonuçlar

İşlem süresi ve flops sayısı bakımından gerçeklenen diğer tüm algoritmalara oranla en yüksek değerlere sahip algoritmadır. Bunun nedeni yapısında bulunan ikinci dereceden (quadratic) fonksiyonun çözümlenmesi ve verilerin öncelikle daha büyük boyutlu bir uzaya taşınıp kümeleme işlemlerinin burada gerçekleştirilmesinden kaynaklanmaktadır. q değerinin artmasıyla işlem süresi artmakta fakat flops sayısı azalmaktadır. q değerine bağlı olarak oluşan küme sayısı, destek vektör sayısı da artmaktadır (seçilen q değerlerinin bu kadar büyük olmasının sebebi özniteliklerin normalize değerler olmasıdır). Birbirine çok yakın değerlere sahip olan CRAB ve IRIS

gibi veri setleri için iyi sonuçlar ortaya çıkmaktadır. Sıradüzensel bir algoritma olmasına rağmen veri setlerinden dolayı IRIS ve CRAB veri setinde kesin bir sıradüzen oluşturamamaktadır. Çizge tabanlı algoritmalara göre daha belirgin ve sağlam kümeler oluşturmaktadır. Çizge tabanlı algoritmalarda çok küçük eşik değeri için küme sayısı aşırı derecede artarken, SVC algoritmasında oluşturulan kümeler daha kararlı yapıya sahiptir. Bunlara ek olarak çizelgeden de görüldüğü gibi, eklenen gürültü oranlarına rağmen küme sayısı çok az değişmektedir. Gerçeklenen algoritma için elde edilen sonuçlar Çizelge 5.7'de görülmektedir:

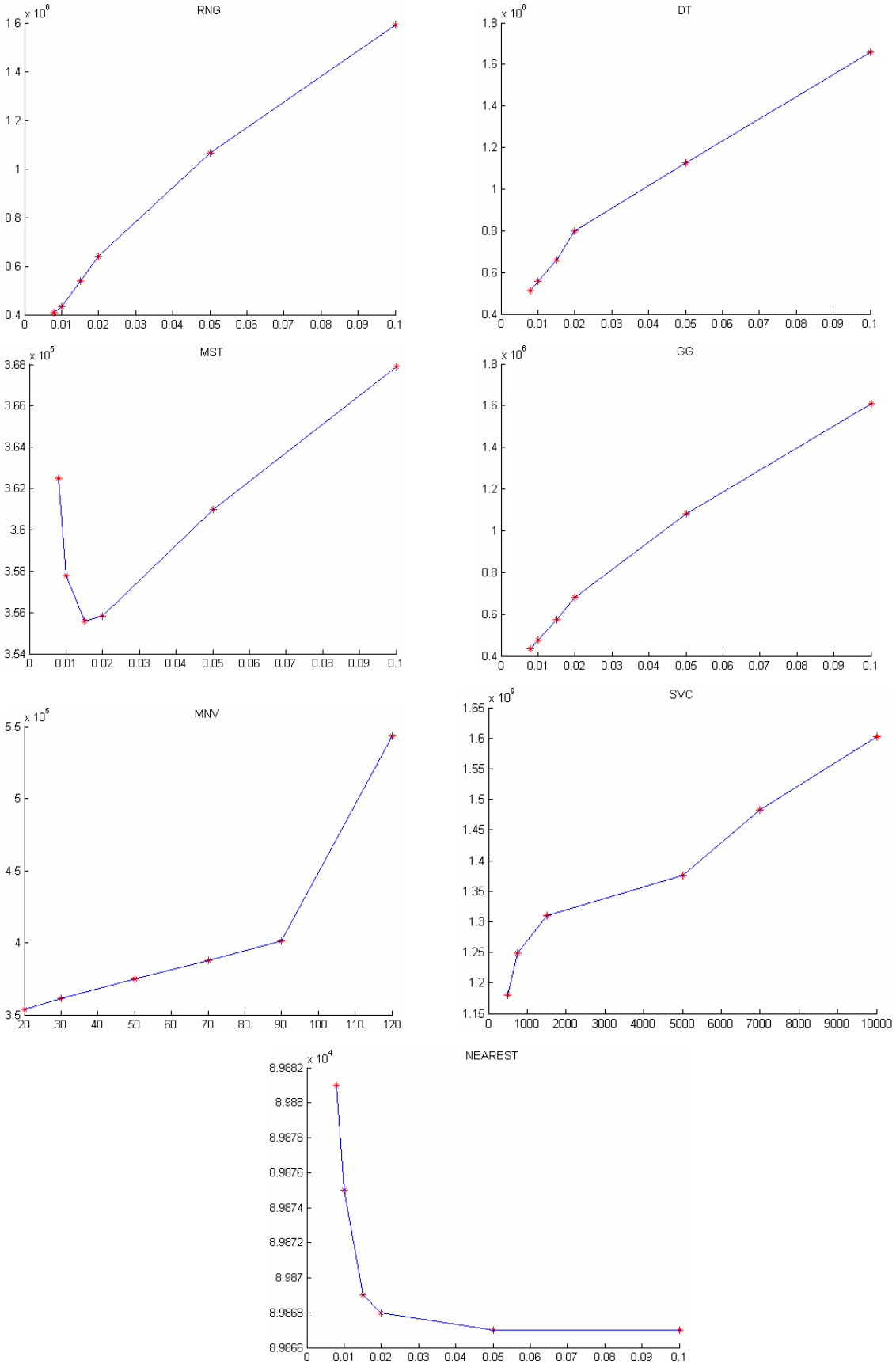
Çizelge 5.7 Destek Vektörleri Algoritması için Sonuçlar

IRIS	**GÜRÜLTÜSÜZ**			**GÜRÜLTÜLÜ**	
q değerleri	**Süre (sn.)**	**Flops (x 10^9)**	**Küme Sayısı**	**Küme Sayısı**	
				45Db	**24Db**
500	191.656	1.1799	1	1	1
750	285.718	1.2488	2	2	2
1500	278.609	1.3095	2	2	2
5000	300.812	1.3755	3	3	2
7000	315	1.483	4	5	6
10000	361.203	1.6023	7	8	8
CRAB	**Süre (sn.)**	**Flops (x 10^9)**	**Küme Sayısı**	**Küme Sayısı**	
q değerleri				**45Db**	**24Db**
750	67.672	3.2894	1	1	1
8000	124.406	3.2411	2	2	2
25000	540.766	2.7744	3	3	2
30000	712.562	2.6208	4	4	4
45000	1.152e+003	2.3451	16	14	15
50000	1.2784e+003	2.348	18	16	18
DATA	**Süre (sn.)**	**Flops (x 10^9)**	**Küme Sayısı**	**Küme Sayısı**	
q değerleri				**45Db**	**24Db**
500	4.047	53658278	1	1	1
750	38.313	75121638	2	2	2
1500	52.016	79146050	6	6	7
5000	52.204	73746657	7	7	7
7000	54.109	73715098	7	7	7
10000	54.422	71681937	7	7	7

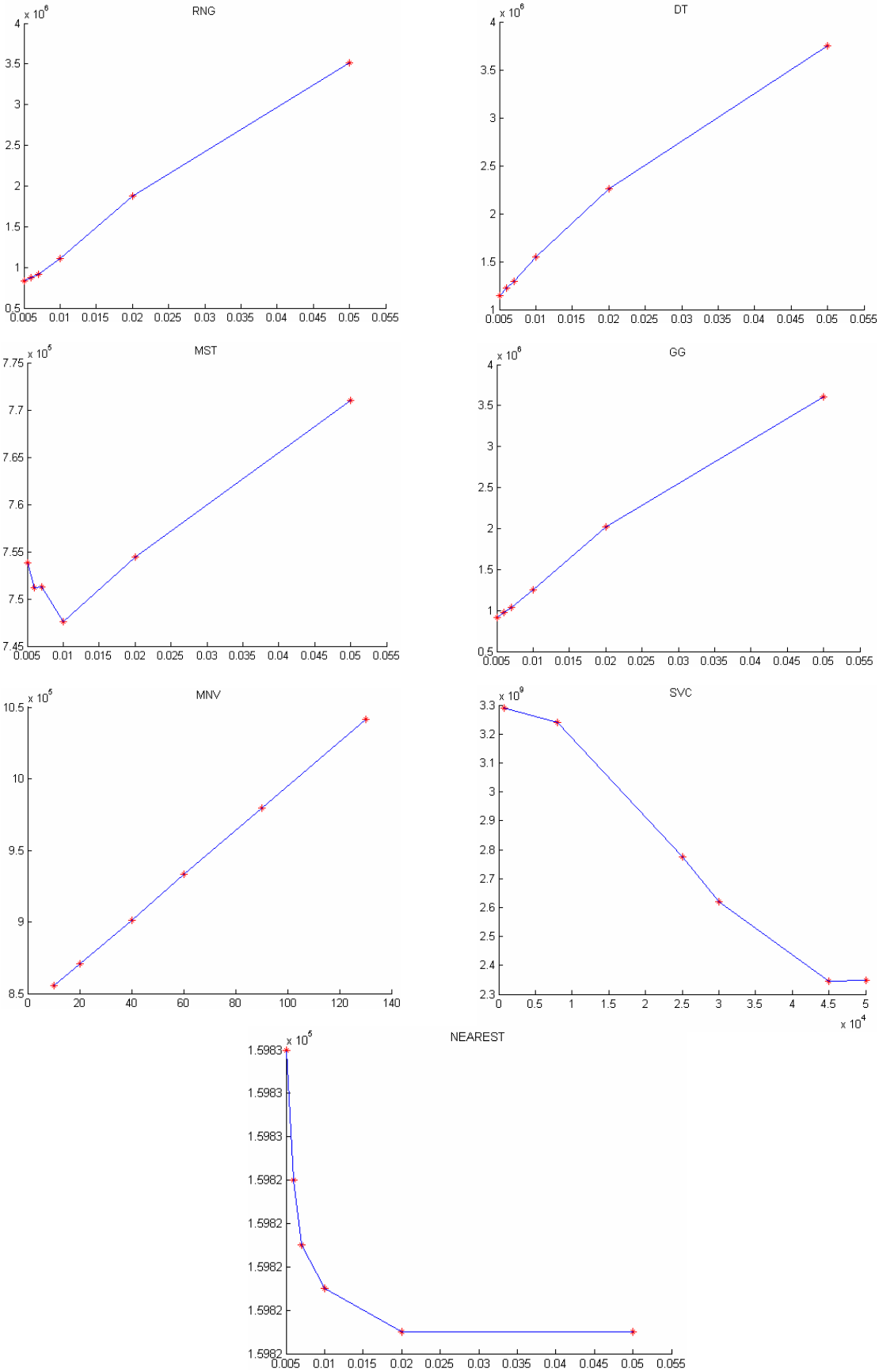
ALTINCI BÖLÜM
SONUÇLAR VE YORUMLAR

6. SONUÇLAR Ve YORUMLAR

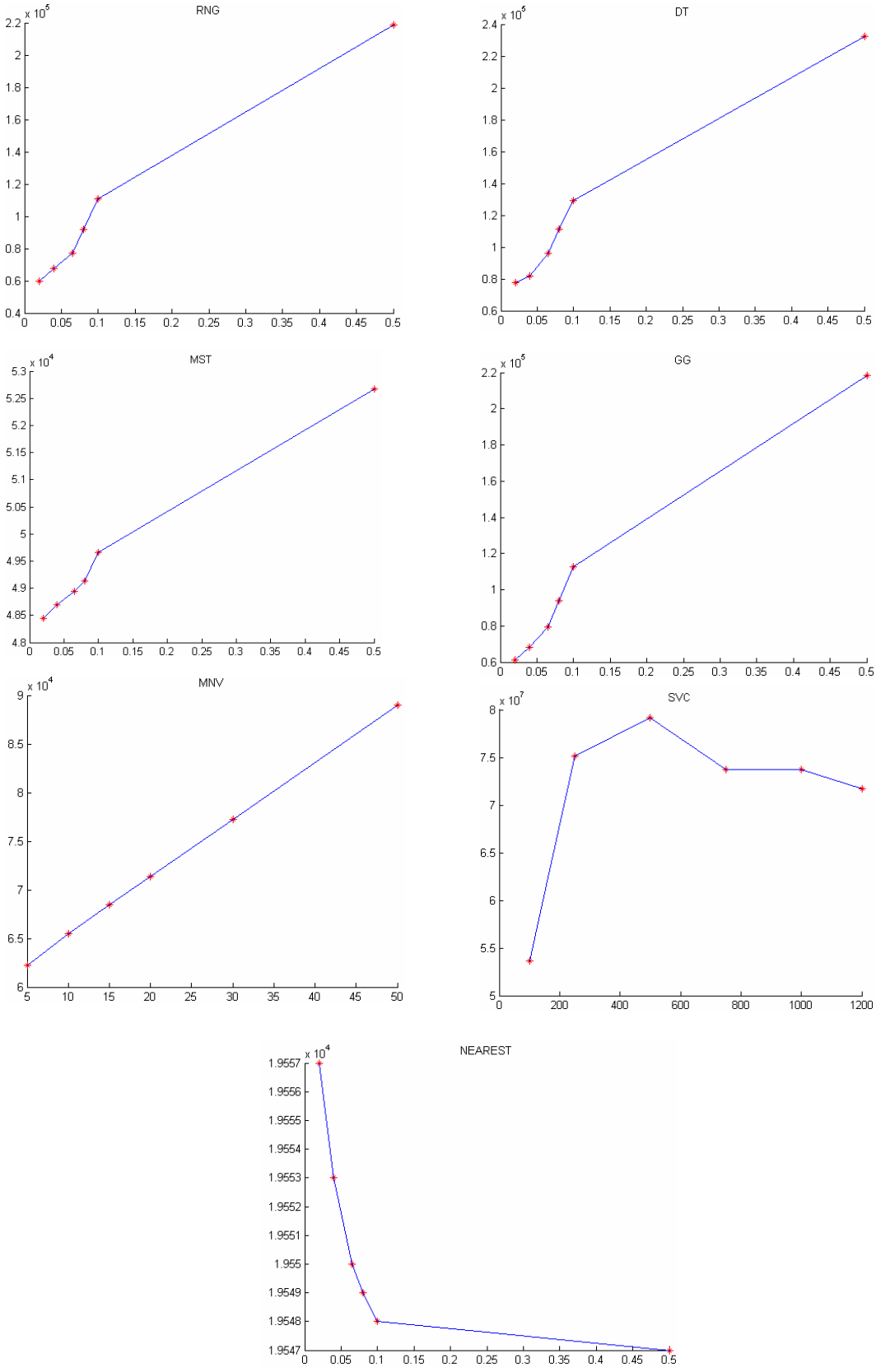
Bu çalışmada öncelikle literatürde bulunan algoritmalardan, küme sayısının daha önceden belli olmadığı, herhangi bir başlangıç kümesinin veya başlangıç noktasının seçilmediği algoritmalar üzerinde çalışılmıştır. Farklı kriter ve farklı veri setlerine göre seçilen yedi algoritmanın performansları MATLAB ortamında karşılaştırılmıştır. Aşağıda sonuçların grafik gösterimleri yer almaktadır. Şekil 6.4, 6.5 ve 6.6 da bulunan grafiklerde siyah kesikli çizgiler (ve kırmızı nokta) gürültüsüz durumu, yeşil noktalı kesikli çizgiler (ve içi boş mavi daire) 45 Db'lik gürültülü eklenmiş durumu ve pembe nokta nokta olan çizgilerde (ve siyah içi boş üçgen) 24 Db'lik gürültü eklenmiş durumu ifade etmektedir.



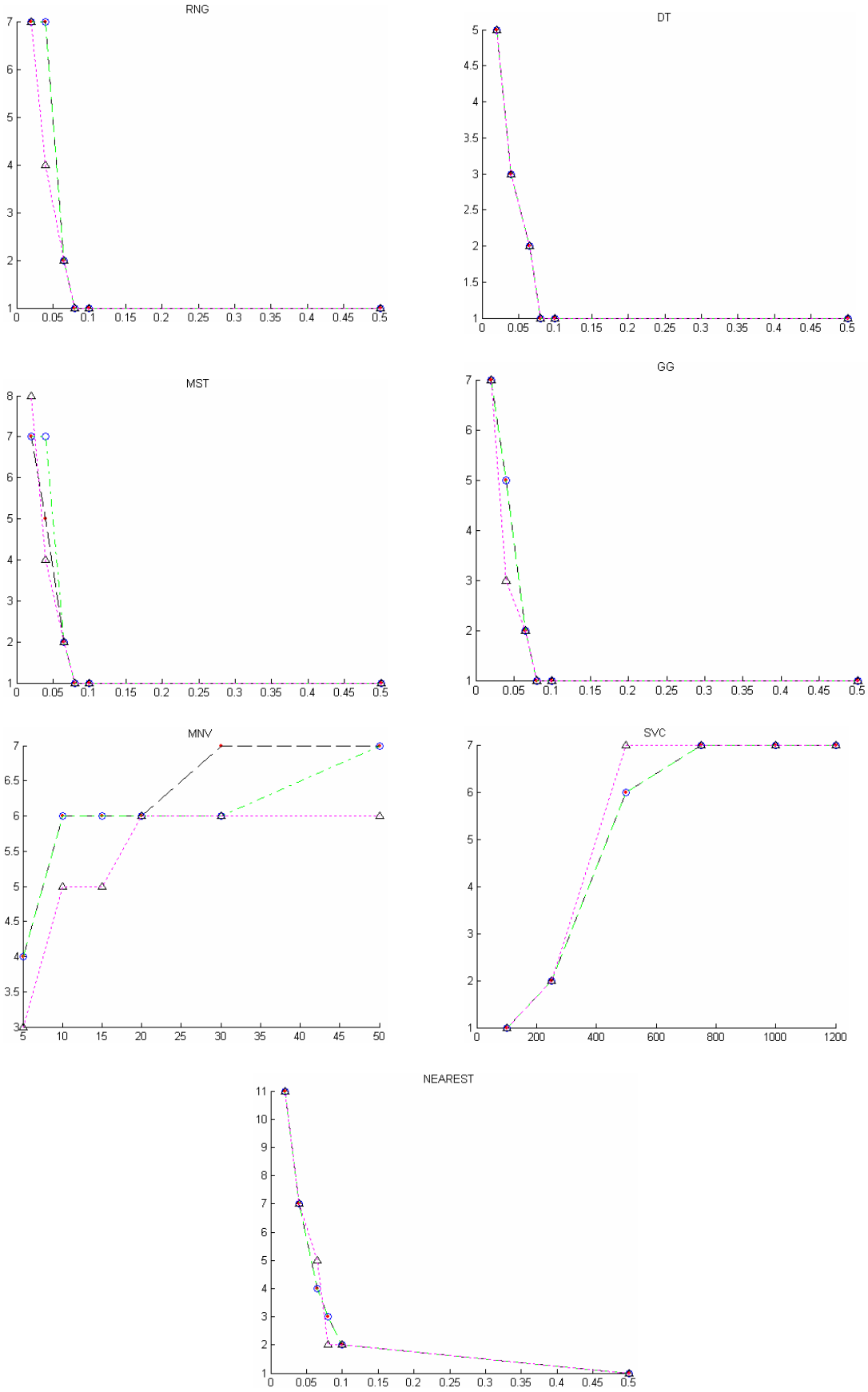
Şekil 6.1 Iris Veri Seti İçin FLOP Sayıları



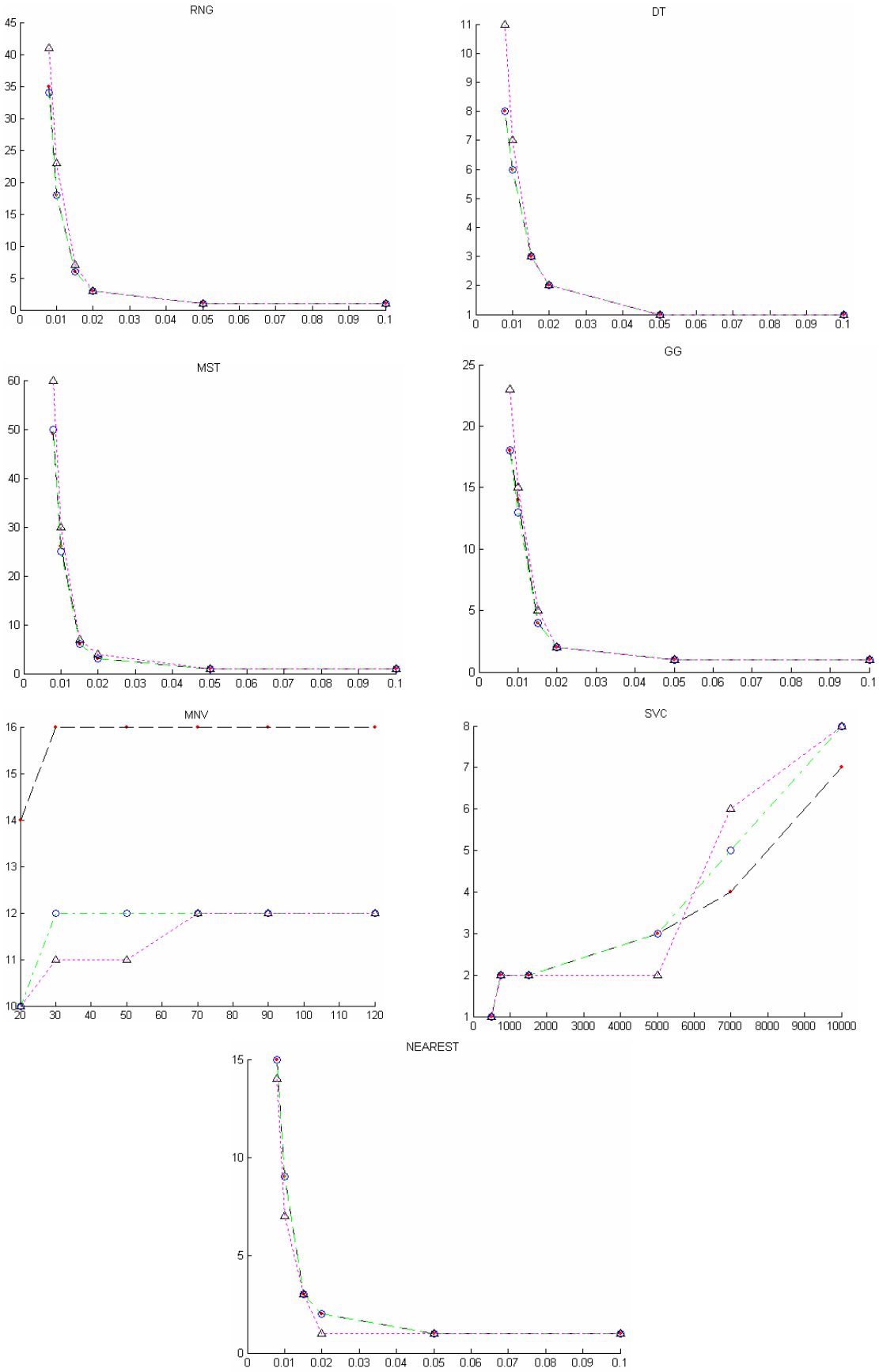
Şekil 6.2 Crab Veri Seti İçin FLOP Sayıları



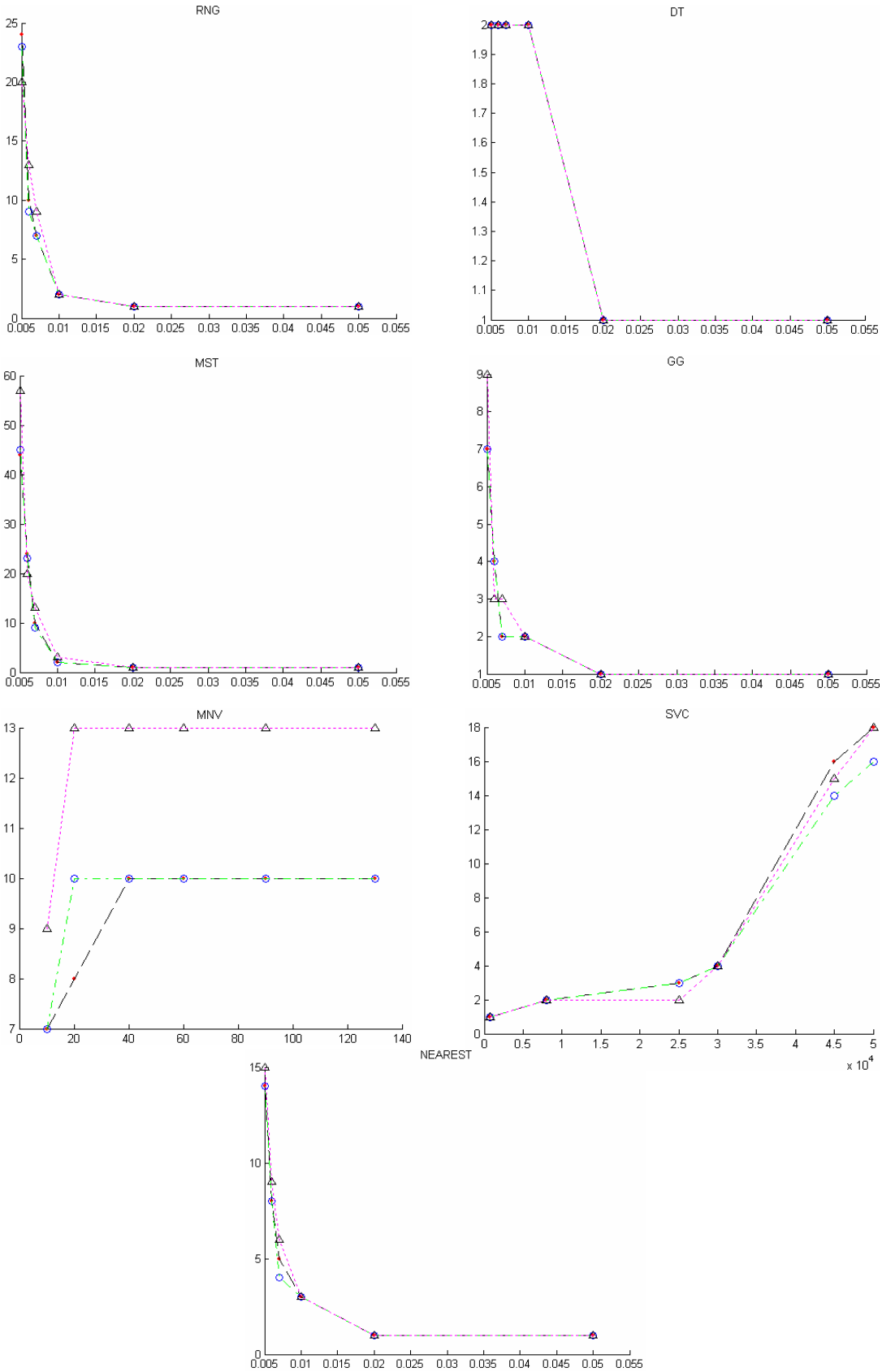
Şekil 6.3 Data Veri Seti İçin FLOP Sayıları



Şekil 6.4 Data Veri Seti İçin Küme Sayılarının Değişimi



Şekil 6.5 Iris Veri Seti İçin Küme Sayılarının Değişimi



Şekil 6.6 Crab Veri Seti İçin Küme Sayılarının Değişimi

Önceden de belirtildiği gibi, kullanılan tüm veri setleri için en iyi kümelemeyi yapabilen bir algoritma bulunmamaktadır. Çünkü tüm kümeleme algoritmalarının performansları verilerin dağılımına bağlıdır, DATA veri seti için başarımı yüksek (düşük flops sayısı ve işlem süresi, daha belirgin kümeler) bir kümeleme yapabilen bir algoritma diğer veri setleri için anlamlı kümeler oluşturamamaktadır. Bu nedenle amacımıza uygun bir kümeleme algoritması önceden belirlenmelidir. Bu belirleme işleminde, uzmanın önemi unutulmamalıdır. Bu çalışmada gerçeklenen algoritmalar arasından tüm özellikler göz önünde bulundurulduğunda, en yakın komşuluk algoritması en iyi algoritma olarak belirlenmiştir fakat bu algoritmanın da dezavantajı eşik değerinin belirlenmesidir.

Algoritmaların seçimi dışında kümeleme işlemlerine önemli oranda etki eden diğer bir önemli husus da uygun eşik değerlerinin belirlenmesidir. Halen üzerinde çalışılan bir konu olmakla birlikte, kümelenecek olan veri setinin yakınlık matrisinin yardımıyla oluşturulan histogram yardımıyla bu eşik değeri seçilebilmektedir (iç içe girmemiş veriler için denenmiştir).

Bundan farklı olarak, yakınlık matrisinde bulunan en büyük, en küçük ve ortalama değerlere göre de eşik değeri belirlenebilmektedir veya bir noktanın diğer noktalara olan uzaklık değerleri arasından ortalama değerin üstünde olan yakınlık değer(ler)i uyuşmayan kenar (inconsistent edge) olarak belirlenip kaldırılmaktadır (Çizge yapılarında sıkça kullanılan bir yöntemdir). İç içe girmiş veriler için bulanık kümeleme kullanılarak daha iyi sonuçlar elde edilebilir.

Günümüzde veri tabanlar terabayt'lar cinsinden ifade edilmektedir. Uydular vasıtası ile alınan bir görüntünün işlenmesi amacıyla kümelere ayırabilmek için hem hızlı, hem de verimli kümeleme algoritmalarına ihtiyaç duyulmaktadır. Daha büyük boyutlu verileri kümeleyebilmek için bu amaca uygun hazırlanmış bilgisayarlar (paket programlar v.b.) ve algoritmalar kullanmak daha elverişlidir.

KAYNAKLAR

Andritsos P., Tsaparas P., Miller R. J., Sevcik K. C., LIMBO: Scalable Clustering of Categorical Data, Hellenic Database Symposium, 2003.

Barbara D., Couto J., Li Y., COOLCAT: An Entropy-Based Algorithm for Categorical Clustering, Eleventh International Conference on Information and Knowledge Management (CIKM'02), 2002.

Ben-Hur A., Horn D., Siegelmann H. T., Vapnik V., Support Vector Clustering, Journal of Machine Learning Research, 125-137, 2001.

Buhmann J. M., Data Clustering and Learning, The Handbook of Brain Theory and Neural Networks, The MIT Pres, 2002.

Cristianini N., Taylor R. S., *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, ISBN 0-521-78019-5, Cambridge, 2000.

Çetin A. E., *Matlab 6.5*, Alfa Basım Yayım Dağıtım Ltd. Şti., ISBN 975-297-364-7, İstanbul, 2003.

Everitt B., *Cluster Analysis*, Heinemann Educational Books, ISBN 0-435-82297-7, London, 1974.

Grizaite G., Innerhofer-Oberperfler R., DBSCAN Clustering Algorithm, Student Reports, Faculty of Computer Science, University of Bozen – Bolzano, 2005.

Gowda K. C., Krishna G., Agglomerative Clustering Using the Concept of Mutual Nearest Neighborhood, Pattern Recognition 10, 105-112, 1978.

Guha S., Rastogi R., Shim K., ROCK: A Robust Clustering for Categorical Attributes, Information Systems, Vol. 25, No. 5, 345-366, 2000.

Guha S., Rastogi, R., Shim K., CURE: An Efficient Clustering Algorithm for Large Databases, In Proceedings of ACM SIGMOD, International Conference on Management of Data, 73-84, New York, 1998.

Hand D. J., Statistics and Data Mining: Intersecting Disciplines, ACM Special Interest Group on Knowledge Discovery and Data Mining, SIGKDD Explorations Journal, Vol. 1, Issue 1, 16, 1999.

Hagan M. T., Demuth H. B., Beale M., *Neural Network Design*, PWS Publishing Company, ISBN: 0-9717321-0-8, Boston, 1996.

Hartigan J. A., *Clustering Algorithms*, John Wiley & Sons Inc., ISBN 0-471-35645-X, New York, 1975.

Horn D., Ben-Hur A., Siegelmann H. T., Vapnik V., A Support Vector Clustering Method, In International Conference on Pattern Recognition, 2000

Hösel V., Walcher S., Clustering Techniques: A Brief Survey, AMS Subject Classification 62H30, 68T10, 2000.

İPLİKÇİ S., Destek Vektörü Makineleri ile Doğrusal Olmayan Sistemlerin Çevrimiçi Modellenmesi, TOK'05, İstanbul, 2005.

Jain A. K., Dubes R. C., *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

Jain A. K., Murty M. N., Flynn P. J., Data Clustering: A Review, ACM Computing Surveys, Vol. 31, No. 3, 1999.

Mangasarian O. L., Data Mining via Support Vector Machines, Data Mining Institute Technical Report, 01-05, 2001.

Mannila H., Data Mining: Machine Learning, Statistics and Databases, Eight International Conference on Scientific and Statistical Database Management, Stockholm, 1996.

Nilsson N. J., *Introduction to Machine Learning*, An Early Draft of a Proposed Textbook, Stanford University, Department of Computer Science, Robotics Laboratory, CA 94305, 1996

Peters M., Zaki M. J., CLICK: Clustering Categorical Data Using K-partite Maximal Cliques, Survey of Clustering Data Mining Techniques, 2002.

Poggio T., Mukherjee S., Rifkin R., Rakhlin A., Verri A., b, Uncertainty in Geometric Computations, Kluwer Academic Publishers, 131-141, 2002.

Pontil M., Verri A., Properties of Support Vector Machines, Neural Computation, Vol. 10, No. 4, 955-974 (20), 1998.

Raymond T., Han J., CLARANS: A Method for Clustering Objects for Spatial Data Mining, IEEE Trans. on Knowledge and Data Engineering, Vol. 14, No. 5, 2002.

Ripley B. D., *Pattern Recognition and Neural Networks*, Cambridge University Press, ISBN 0-521-460-867, Cambridge, 1996.

Stitson M. O., Weston J. A. E., Gammerman A., Vovk V., Vapnik V. N., Theory of Support Vector Machines, Technical Report CSD-TR-96-17, Royal Holloway University of London, England, 1996.

Strauss R. E., Matlab Functions and Script Files, Biological Sciences, Texas Tech University, http://www.biol.ttu.edu/Strauss/Matlab/matlab.htm, 2005.

Tera, M., Techniques in Data Clustering, A report on clustering, http://www.cis.ksu.edu/~mst9696/864paper.pdf

Tryon R. C., Bailey D. E., *Cluster Analysis*, McGraw-Hill Book Company, New York, 1970.

Vapnik V. N., *The Nature of Statistical Learning Theory*, Springer-Verlag Inc., ISBN 0-387-94559-8, New York, 1995.

Vapnik V. N., *Statistical Learning Theory*, John Wiley & Sons Inc., ISBN 0-471-03003-1, New York, 1998.

Vapnik V. N., An Overview of Statistical Learning Theory, IEEE Trans. On Neural Networks, Vol. 10, No. 5, 1999.

Venkataraman P., *Applied Optimization with MATLAB Programming*, John Wiley & Sons Inc., ISBN 0-471-34958-5, U.S.A., 2002.

Zahn C. T., Graph Theoretical Methods for Detecting and Describing Gestalt Clusters, IEEE Trans. on Computers, SLAC-PUB-672, 1970.

Zhang B., Is The Maximal Margin Hyperplane Special In a Feature Space?, HP Laboratories Palo Alto, HPL-2001-89, 2001.

Zhang T., Ramakrishman R., Livny M., BIRCH: An Effective Data Clustering Method for Very Large Databases, ACM SIGMOD Record, Vol. 25, No. 2, pages 103-114, 1996.

http://mathworld.wolfram.com/

http://planetmath.org/

http://jmlr.csail.mit.edu/

http://www.csail.mit.edu/index.php

http://www.lans.ece.utexas.edu/~strehl/soft.html

http://www.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter8/sld023.htm

http://www.unesco.org/webworld/idams/advguide/Chapt7_1_1.htm

http://db.cs.sfu.ca/GeoMiner/survey/html/node9.html

http://www.seafriends.org.nz/enviro/crust/grapsida.htm

EKLER

MATLAB Programı:

Ek 1 :nearest_neighbor.m

```
function [group] = nearest_neighbor(Samples,esik);
tic;
flops(0);
[k Sample_number] = size(Samples);

% figure;
% plot(Samples(1,:),Samples(2,:),'r.');
% title('KUMELENECEK NOKTALAR');

for i = 1:Sample_number;
   adjacent_matrix(i,i) = 0;
   for j = i+1:Sample_number;
      adjacent_matrix(i,j) = sqrt((Samples(1,i) - Samples(1,j))^2 + (Samples(2,i) - Samples(2,j))^2);%
Euclidean Distance.
      adjacent_matrix(j,i) = adjacent_matrix(i,j);
   end
end

for t=1:Sample_number;
   for z=1:Sample_number;
      if adjacent_matrix(t,z)==0;
         adjacent_matrix(t,z)=1000; % Koşegen üzerindeki sıfırların kaldırılması.
      end
   end
end

group=zeros(Sample_number);
group(1,1)=1;                                    % İlk noktanın yerleştirilmesi

for sut=2:Sample_number;
   for sat=1:sut-1;
      d(sat,1)=(adjacent_matrix(sat,sut));
   end

   [dmin nearest]=min(d);
   [clust_of_nearest,j]=find(group==nearest); % En yakın komşunun bulunması.
   max_clust=max(clust_of_nearest);
   [clust_num,k]=find(group~=0);
   max_clust_number=max(clust_num);

   if dmin<=esik;                                % Eşik Değeri ile karşılaştırma
      group(max_clust,sut)=sut;
   else
      max_clust_number=max_clust_number+1;
      group(max_clust_number,sut)=sut;
   end
end

clusters=zeros(max_clust_number);
```

```
for i=1:max_clust_number;
   k=1;
   for j=1:Sample_number;
      if group(i,j)~=0;
         clusters(i,k)=group(i,j);
         k=k+1;
      end
   end
end

for i=1:max_clust_number;
   element_number_in_a_row=find(clusters(i,:)~=0);
   max_element_number_in_a_row(i,:)=max(element_number_in_a_row);
end

for i=1:max_clust_number;
   for j=1:max(max_element_number_in_a_row);
      new_clusters(i,j)=clusters(i,j);
   end
end
toc;

new_clusters=num2str(new_clusters)
max_clust_number
flops
```

Ek 2:mst_son.m

```
function [Xmst,D] = mst_son(Samples,esik_degeri);

tic;

[row,column] = size(Samples);

Sample_number=row;

D = dist(Samples');

Dmax = max(max(triu(D)))*10;

[Dmin,Dwin] = min(D(1,2:row));
Xmst(1,:) = [1 Dwin+1];

for i=2:row-1
   mindist = Dmax;
   Xmstlist = unique(Xmst(:))';
   Xmstnotlist = setdiff(1:row,Xmstlist);   % A'da olup B'de olmayanları ayırdık.
   for j=Xmstlist
      [Dmin,Dwin] = min(D(j,Xmstnotlist));
      if (Dmin < mindist)
         minindex = [j Xmstnotlist(Dwin)];
         mindist = Dmin;
      end
   end
   Xmst(i,:) = minindex;
end

[cluster_assignments] = Assignn(D,Sample_number,esik_degeri,Samples);
```

Assignn.m

```
function [cluster_assignments] = Assignn(D,Sample_number,esik_degeri,Samples);

cluster_assignments = zeros(Sample_number,1);
cluster_index = 0;
dur = 0;

while dur ~= 1;

  index = 1;
  while cluster_assignments(index) ~= 0
    index = index + 1;
    if index > Sample_number;% tüm noktalar kümelendi
      dur = 1;
      break;
    end
  end

  if dur ~= 1;

            % yeni bir küme
    cluster_index = cluster_index + 1;

    % yığın
    stack = zeros(Sample_number,1);
    stack_index = 0;

    % yığının içine koy
    stack_index = stack_index + 1;
           stack(stack_index) = index;

          % yığın boş olmadıkça işleme devam edilir.
       while stack_index ~= 0;

      % yığından bir düğüm seçilir.
      node = stack(stack_index);
      stack_index = stack_index - 1;

      % küme numarası atanır.
      cluster_assignments(node) = cluster_index;

                              % tüm komşular kontrol edilir.
                             for i = 1:Sample_number;

                 % bu düğümün komşu olduğu ve henüz kümelenmediği kontrol edilir.
                  if D(node,i) <= esik_degeri & cluster_assignments(i) == 0 & i ~= node;
          % yığına ekle.
           stack_index = stack_index + 1;
           stack(stack_index) = i;

                  end
               end
             end
          end
       end
```

```
[number_of_clusters,clusters] = gosterr(Samples,cluster_assignments);
```

gosterr.m

```
function [number_of_clusters,clusters] = gosterr(Samples,cluster_assignments);

number_of_clusters = max(cluster_assignments);
color_vector = zeros(1,3);
[Sample_number kk] = size(Samples);
clusters = zeros(number_of_clusters);
z = 1;

for j = 1 : number_of_clusters;
   for i = 1 : Sample_number;
      if j == cluster_assignments (i);
         clusters(j,z) = i;
         z = z+1;
      end
   end
   z = 1;
end

for i=1:number_of_clusters
   element_number_in_a_row=find(clusters(i,:)~=0);
   max_element_number_in_a_row(i,:)=max(element_number_in_a_row);
end

for i=1:number_of_clusters
   for j=1:max(max_element_number_in_a_row)
      new_clusters(i,j)=clusters(i,j);
   end
end
toc;
flops
new_clusters = num2str(new_clusters)
number_of_clusters
```

Ek 3:DT.m

```
function [group]=DT(Samples,esik_degeri)
tic;
flops(0);
[k Sample_number] = size(Samples);

% Noktalar arasindaki uzakliklarin bulunmasi.

adjacent_matrix=dist(Samples);

X=Samples(3,:);
Y=Samples(4,:);

% Grafların cizdirilmesi
% DELAUNAY TRIANGULATION (DT)

figure;
hold;
```

```
TRI = delaunay(X,Y);
trimesh(TRI,X,Y,zeros(size(X)))
plot(X,Y,'r.','LineWidth',2);
title('DELAUNAY TRIANGULATION');

figure;
hold on;
[vx, vy] = voronoi(X,Y,TRI);
plot(X,Y,'r.',vx,vy,'b-');
axis ([0 4 0 4])

figure;
plot(X,Y,'r.','LineWidth',2);
title('UZAK KENARLARIN KALDIRILMASI');
hold on;

%Yakınlık matrisinin oluşturulması ve esik degerine göre uzak noktaların belirlenerek kaldırılması.

for i = 1:Sample_number;
   for j = i+1:Sample_number;
      if adjacent_matrix(i,j) <= esik_degeri;
         plot([Samples(1,i),Samples(1,j)],[Samples(2,i),Samples(2,j)],'g-');
      end
   end
end

[cluster_assignments] = AssignDT(adjacent_matrix,Sample_number,esik_degeri,Samples);
```

Ek 4:RNG.m

```
function [group]=RNG(Samples,esik_degeri)
tic;
flops(0);
[k Sample_number] = size(Samples);

Noktalar arasindaki uzakliklarin bulunmasi.

adjacent_matrix=dist(Samples);

%RELATIVE NEIGHBORHOOD GRAPH (RNG)... lune

figure;
plot(Samples(1,:),Samples(2,:),'r.');
title('RELATIVE NEIGHBORHOOD GRAPH');

hold on;

for i = 1 : Sample_number;
   for j = i+1 : Sample_number;
      flag = 1;
      k = 1;
      while (flag>0 & k<=Sample_number);
         if (k~=i & k~=j);
            if (adjacent_matrix(i,j)>max(adjacent_matrix(i,k),adjacent_matrix(j,k)));
               flag=0;
            end
         end
         k = k+1;
```

```
      end
      if flag > 0
         plot([Samples(1,i),Samples(1,j)],[Samples(2,i),Samples(2,j)],'g-');
      end
   end
end

figure;
plot(X,Y,'r.','LineWidth',2);
title('UZAK KENARLARIN KALDIRILMASI');

hold on;

%Yakınlık matrisinin oluşturulması ve esik degerine göre uzak noktaların belirlenerek kaldırılması.

for i = 1:Sample_number;
   for j = i+1:Sample_number;
      if adjacent_matrix(i,j) <= esik_degeri;
         plot([Samples(1,i),Samples(1,j)],[Samples(2,i),Samples(2,j)],'g-');
      end
   end
end

[cluster_assignments] = AssignRNG(adjacent_matrix,Sample_number,esik_degeri,Samples);
```

Ek 5:GG.m

```
function [group]=GG(Samples,esik_degeri)
tic;
flops(0);
[k Sample_number] = size(Samples);

%Noktalar arasindaki uzakliklarin bulunmasi.

adjacent_matrix=dist(Samples);

%GABRIEL GRAPH (GG)... disk (construction of tree)

figure;
plot(Samples(1,:),Samples(2,:),'r.');
title('GABRIEL GRAPH');
hold on;


for i = 1 : Sample_number;
   for j = i + 1 : Sample_number;
      flag = 1;
      k = 1;
      while (flag > 0 & k <= Sample_number);
         if (k ~= i & k ~= j);
            if (adjacent_matrix(i,j)^2>(adjacent_matrix(i,k)^2+adjacent_matrix(j,k)^2))
               flag=0;
            end
         end
         k=k+1;
      end
      if flag>0
         plot([Samples(1,i),Samples(1,j)],[Samples(2,i),Samples(2,j)],'g-');
```

```
      end
   end
end

figure;
plot(X,Y,'r.','LineWidth',2);
title('UZAK KENARLARIN KALDIRILMASI');

hold on;

%Yakınlık matrisinin oluşturulması ve esik degerine göre uzak noktaların belirlenerek kaldırılması.

for i = 1:Sample_number;
   for j = i+1:Sample_number;
      if adjacent_matrix(i,j) <= esik_degeri;
         plot([Samples(1,i),Samples(1,j)],[Samples(2,i),Samples(2,j)],'g-');
      end
   end
end

[cluster_assignments] = AssignGG(adjacent_matrix,Sample_number,esik_degeri,Samples);
```

Ek 6:MNV.m

```
function [] = MNV(Samples,number_of_near_neighbors)


tic;
flops(0);

[k Sample_number] = size(Samples);

if number_of_near_neighbors >= Sample_number
   disp('!!!WARNING!!!')
   disp('Near Neighbor number is bigger than the sample number')
   disp('if there are n samples; a sample can have n-1 neighborhood')
   disp('So; your near neighbor number must be less then Sample number...')
   break;
end

adjacent_matrix=dist(Samples); % Noktalar arasindaki uzakliklarin bulunmasi.

for t=1:Sample_number;
   for z=1:Sample_number;
      if adjacent_matrix(t,z)==0;
         adjacent_matrix(t,z)=100; % Koşegen üzerindeki sıfırların kaldırılıp yerlerine 100 yazılması.
      end
   end
end

sorted_adjacent_matrix=sort(adjacent_matrix); % Komşuluk matrisindeki değerlerin küçükten büyüğe
(en yakindan en uzağa) yazılması.

for column=1:Sample_number;% en yakın "k" noktanın belirlenmesi
   for row=1:number_of_near_neighbors;
      near_k_neighbors(row,column)=(sorted_adjacent_matrix(row,column));
      [m,n]=find(adjacent_matrix==(near_k_neighbors(row,column)));
      % indislerde hata veriyordu (aynı numaralı indisler;
```

```
      % rasgeldiği zaman kendisinden küçük numaralı indisleri işleme sokuyordu) düzeltildi.
      if column==n(2,1);
         n=n(1,1);
      else
         n=n(2,1);
      end
      near_k_neighbors(row,column)=n; % en yakın "k" noktanın matris olarak gösterilmesi.
   end
end

for column=1:Sample_number; % tüm noktaların MNV değerlerinin hesaplanması.
   for row=1:Sample_number-1;
      all_neighbors(row,column)=(sorted_adjacent_matrix(row,column));
      [m,n]=find(adjacent_matrix==(all_neighbors(row,column)));
      if column==n(2,1);
         n=n(1,1);
      else                % her noktanın en yakın komşularını belirledik.
         n=n(2,1);
      end
      all_neighbors(row,column)=n; % en yakın noktaların matris olarak gösterilmesi.
   end
end

% all_neighbors=num2str(all_neighbors)

for column=1:Sample_number;
  for row=1:Sample_number-1;
    x=all_neighbors(row,column); % tüm noktalar için MNV değerlerinin belirlenmesi.
    [k,l]=find(all_neighbors(:,column)==x);
    if isempty(k)==1;
      p=0.00001;
    else
      p=min(k);
    end
    [u,v]=find(all_neighbors(:,x)==column);
    if isempty(u)==1;
      q=0.00001;
    else
      q=min(u);
    end
    MNV_of_all_points(x,column)=p+q; % Her noktanın MNV değerlerinin bulunması.
  end
end

max_MUNEVA=max(max(MNV_of_all_points));  % maksimum MNV değerinin bulunması.

for column=1:Sample_number;
   for row=1:Sample_number;
      if MNV_of_all_points(row,column) > number_of_near_neighbors
         % Bulunan MNV değerleri, eğer kullanıcının belirlediği sınırdan büyük ise bunları yüksek bir
sayıya atıyoruz.
         MNV_of_all_points(row,column)=100;
      end
   end
end

cluster=zeros(2*number_of_near_neighbors,Sample_number);
```

```
for r=2:2*number_of_near_neighbors;
   [i,j] = find(MNV_of_all_points == r); % sırayla MNV=2,3,...,2k için MNV'leri sıralıyoruz.
   per_cluster=unique([i' j']);
   [t,z]=size(per_cluster);
   for m=1:z
      group(r-1,m)=per_cluster(t,m);
   end
end

[row_group,column_group]=size(group);
clusters=zeros(row_group,column_group);
NN=row_group*column_group;
check=zeros(NN,1);
flag=1;

for i=1:row_group %satırları say
   for j=1:column_group %sutunları say
      kontrol=0;
      for k=1:NN %c dizisine bak
         if group(i,j)==check(k,1) % eğer c dizisinde varsa önceden kullanılmış bir rakamdır
            kontrol=1; % kontrol 1 ise kullanılmıştır, 0 ise hiç kullanılmamıştır
         end
      end

      if kontrol==1
         clusters(i,j)=0; % önceden kullanılmış ise 0 ata
      else if kontrol==0
            clusters(i,j)=group(i,j); % kullanılmamış ise değeri ata
            check(flag,1)=group(i,j); %sonrakileri karşılaştırmak için c dizidisinde sakla
            flag=flag+1;
         end
      end
   end
end

[clust_num,k]=find(clusters~=0);
max_clust_number=max(clust_num);
new_clusters=zeros(max_clust_number);
[num,column_clusters]=size(clusters);

for i=1:max_clust_number;
   k=1;
   for j=1:column_clusters;
      if clusters(i,j)~=0;
         new_clusters(i,k)=clusters(i,j);
         k=k+1;
      end
   end
end

% for z=1:Sample_number;
%    [i,j]=find(new_clusters==z);
%    cluster_assignments(z,1)=i;
% end

% color_vector = zeros(1,3);

% figure;
```

```
% hold on;
%
% for i = 1 : max_clust_number;
%      cur_cluster = find(cluster_assignments == i);% farklı kümelere farklı renklerin verilmesi (toplam 7
renk).
%     color = dec2bin(mod(i-1,7),3);
%     color_vector(1) = str2num(color(1));
%     color_vector(2) = str2num(color(2));
%     color_vector(3) = str2num(color(3));
%
plot([Samples(1,cur_cluster)],[Samples(2,cur_cluster)],'linestyle','none','marker','.','color',color_vector);%
kümelerin gösterimi
%     title('FARKLI KUMELERE AIT NOKTALARIN FARKLI RENKLER ILE GOSTERILMESI');
% end
toc;
flops
new_clusters=num2str(new_clusters)
max_clust_number
```

Not: Çizge tabanlı algoritmalarda küme atamalrı birbirine benzer şekilde yapılmaktadır. Bu nedenle dosyalar uzun olduğu için ekte verilmemiştir. Destek Vektör Makineleri ile kümele yapılmasını sağlayan MATLAB kodları da çok uzun olduğu için burada verilmemiştir.

ÖZGEÇMİŞ

Adı, soyadı: Mustafa Seçkin DURMUŞ

Ana adı: Hatice

Baba adı: Cevdet

Doğum yeri ve tarihi: Antalya, 27.08.1980

Lisans eğitimi ve mezuniyet tarihi: Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü, 2002

Çalıştığı yer: Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü

Bildiği yabancı dil, aldığı belgeler: İngilizce, ÜDS 65

Mesleki etkinlikleri: E.M.O. üyeliği
III. Otomasyon Sempozyum ve Sergisi yürütme kurulu üyeliği