EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

BULANIK *c*-ORTALAMALAR KÜMELEME ANALİZİ VE UYGULAMALARI

Gözde ULUTAGAY

İstatistik Anabilim Dalı

Bilim Dalı Kodu: **406.01.01**

Sunuş Tarihi: **21.12.2004**

Tez Danışmanı: Doç. Dr. Şanslı ŞENOL

Bornova-İZMİR

Sayın **Gözde ULUTAGAY** tarafından YÜKSEK LİSANS TEZİ olarak sunulan **"Bulanık *c*-Ortalamalar Kümeleme Analizi ve Uygulamaları"** adlı bu çalışma, "Lisanüstü Eğitim ve Öğretim Yönetmeliği"nin 12 inci madde (c) ve (d) bentleri ve Enstitü yönergesinin ilgili hükümleri dikkate alınarak tarafımızdan değerlendirilmiş olup yapılan sözlü savunma sınavında aday oy ..birliği..... ile başarılı bulunmuştur. Bu nedenle, Gözde ULUTAGAY'ın sunduğu metnin yüksek lisans tezi olarak kabulüne oy ..birliği... ile karar verilmiştir.

21 Aralık 2004

Jüri Başkanı ; Doç.Dr.....Sadi....SENAL.....imza .....

Üye ; Prof.Dr....Etem.NASIBOV..imza .....

Üye ; Doç.Dr....Ufuk...NURİYEV...imza .....

ÖZET

BULANIK *c*-ORTALAMALAR KÜMELEME ANALİZİ VE UYGULAMALARI

ULUTAGAY, Gözde
Yüksek Lisans tezi, İstatistik Bölümü
Tez Yürütücüsü: Doç. Dr. Şanslı ŞENOL
Aralık 2004, 125 sayfa

Bu çalışmada bulanık kümeleme ile ilgili deneyimler için kolaylık sağlayacak visual-interaktif işlem sistemi oluşturulmuştur. Bazı sık kullanılan bulanık kümeleme, küme sayısı belirleme ve başlangıç kümelerin seçilmesi yöntemlerinin algoritmaları programlanarak bir sistem halinde entegre edilmiştir. Menüler aracılığı ile gerekli yöntemlerin seçilebilmesi, sonuçların görsel olarak ekrana verilebilmesi, veri setinin görsel olarak oluşturulabilmesi, optimal küme sayısının otomatik ayarlanabilmesi gibi işlemlerin yapılabilme imkanı sağlanmaktadır.

Anahtar Sözcükler: Bilgi İşlem Sistemi, Bulanık Kümeleme, Küme Geçerliliği.

ABSTRACT

FUZZY *c*-MEANS CLUSTER ANALYSIS AND ITS APPLICATIONS

ULUTAGAY, Gözde

M.Sc. in Statistics

Supervisor: Assoc. Prof. Şanslı ŞENOL

December 2004, 125 pages

In this study, a visual-interactive processing system providing facility for experiences related to fuzzy clustering is presented. Certain frequent algorithms of fuzzy clustering, cluster validity and selection of initial clusters are programmed as an integrated system. It is possible to select necessary methods, assess the visual outputs in the screen, constitute data set visually, and adjust the optimal cluster number automatically by the help of the menu.

***Keywords*:** Information Processing System, Fuzzy Clustering, Cluster Validity.

TEŞEKKÜR

Tezimi hazırlamam sırasında benden desteklerini esirgemeyen, danışmanım ve bölüm başkanım sayın Doç. Dr. Şanslı ŞENOL'a; kıymetli zamanını ve görüşlerini daima benimle paylaşan, bilim ve insanlık adına çok şey öğrendiğim değerli hocam Prof. Dr. Efendi NASİBOV'a; desteğiyle bana hep cesaret veren hocam Prof. Dr. Onur BASKAN'a; ihtiyacım olduğu her zaman yanımda olan canım annem ve anneanneme; desteğini her an hissettiren, tükenmeyen sabrı ve umuduyla bana güç veren sevgili eşim Murat'a; varlığıyla bana yaşam enerjisi sağlayan melek kızım Zeynep Naz'a ve başta babam olmak üzere beni destekleyen tüm sevdiklerime sonsuz teşekkürler...

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (Devamı)

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ (Devamı)

Sayfa

TABLOLAR DİZİNİ

Sayfa

1. GİRİŞ

Problem çözümlerinde sistemler, gerçekliğin bir kısmının modellemesi olarak kurulurlar. Bütün sistem modelleri karmaşıklık, güvenilirlik, belirsizlik arasındaki ilişki ile değerlendirilir. Gerçek dünya her zaman için idealleştirilmiş dünyadan sapmalar gösterir. Matematik modeller ne kadar ayrıntılı olursa olsun gerçeği yansıtmazlar. Çünkü, gerçek dünya karmaşıktır. Bu karmaşıklık genel olarak, belirsizlik, kesin düşünceden yoksunluk ve karar verilemeyişten kaynaklanır (Baykal ve Beyan, 2004).

Birçok gerçek sistem doğrusal değildir ve bunların klasik yöntemlerle incelenmesinde doğrusallığı kabul etmek için belirli katsayılar hesaplara eklenerek olabilecek belirsizlikler belirgin şekilde göz önünde tutulur. Buna gerek kalmadan hesaplama yapılabilmesi için belirsizlik ilkeleri kullanılmalıdır.

Bilimde belirsizliği istenilmeyen bir durum olarak gören ve mümkün bütün durumlarda kaçınılması gerektiğinde ısrar eden geleneksel anlayıştan, belirsizlikle yaşamayı kabul eden ve bilimde bundan kaçınılmasının mümkün olmadığını iddia eden alternatif bakış açısına doğru bir geçiş vardır. Belirsizlik sadece kaçınılması mümkün olmayan bir durum değil, aynı zamanda büyük bir yarar sağlayan ve üzerinde çalışılması gereken bir alandır. Her tehdit, risk ve belirsizlik aynı zamanda bir fırsat ve seçme şansına dönüşebilir.

Sistem modellemelerinde belirsizliğin artmasına izin vermek karmaşıklığı azaltırken, güvenilirliği arttırmaktadır. En uygun davranış

tarzı, her bir modelleme problemi için optimum düzeyde belirsizliğe izin veren yöntemler geliştirmektir.

Belirsizlik iki kategoride incelenir. Bunlar rasgelelik ve bulanıklıktır. Genel olarak, *rasgelelik* o olayın meydana gelmesindeki belirsizliğin sayısal ölçüsüdür. İstatistik ve olasılık eğitimi almış pek çok kimse belirsizlik ile rasgeleliğin aynı olduğuna inanır. Özellikle, olasılığı, bir sıklık veya diğer sınanabilir bir yığın olarak görmeyip, bilginin öznel bir durumu olarak gören Bayesçi istatistikçiler, bu iddiayı savunmuş ve toplumda tüm belirsizliklerin rasgele olduğu kavramı yaygınlaşmıştır. Rasgeleliğin en önemli özelliği, sonuçların ortaya çıkmasında tamamen şans olayının rol oynaması ve gerekli öngörülerin ve tahminlerin kesin bir doğrulukla önceden yapılamamasıdır.

Ancak bilinen belirsizliklerin hepsi rasgele karakterde değildir. Sözel belirsizlikler *bulanıklık* adını alır. Bulanıklık, belirsiz anlamlılık, değişik yorumlar yapabilmek anlamına gelir. Ne kadar yetersiz veri varsa, bulanıklık o kadar fazla olur. Gerçek dünya sorunları ne kadar yakından incelemeye alınırsa, çözüm daha da bulanık hale gelecektir. Çünkü çok fazla olan bilgi kaynaklarının tümünü insan aynı anda ve etkileşimli olarak kavrayamaz ve bunlardan net sonuçlar çıkaramaz.

Bulanık kelimesi genel olarak puslu, dumanlı, kesinlikle ayırt edilemeyen, kesin olmayan, belirsiz, kafa karıştıran, müphem gibi anlamlara gelir. Bulanıklık, incelenen konunun inceleyen kişi tarafından tam kesinlikle bilinmemesi durumunda sahip olunan eksik ve belirsiz bilgilerin tümüdür. Böylece klasik analitik yöntemler ile dinamik ve korunum ilkelerin elde ettiği denklemler, veri ve bilgilerde bulanıklık tür

belirsizlik bulunduğu durum için kullanılamaz. Bu gibi durumlarda ilgili sözel ve oldukça belirsiz bilgiler de göz önünde tutularak modellenebilir.

Rasgelelik, olayın oluşundaki kesin olmayışlığı ifade eder. Bulanıklık ise, olayın olup olmadığını değil, hangi dereceye kadar olduğunu ölçer. Bir olayın olup olmadığı rasgeleliktir. Hangi dereceye kadar olduğu ise bulanıklıktır. Bulanıklık, genel olarak gerekirci olmasına rağmen, rasgelelik öngörüye dayanır.

Bulanık küme kavramı, 1960'ların ortasında Zadeh'in, klasik sistem kuramının matematiksel yöntemlerinin gerçek dünyadaki, özellikle insanları içeren kısmen karmaşık sistemlerle uğraşırken yetersiz kalmasından dolayı hoşnut kalmayışından doğmuştur. Matematiksel olarak bulanıklık, çok-değerlilik demektir. Azerbaycan asıllı Lütfü Askerzade (1965), niteliklerin ikili üyelik fonksiyonuyla ifade edildiği klasik kümeler yerine, dereceli üyelik fonksiyonuyla ifade edildiği bulanık kümeler tanımlamasını teknik terimlere dahil etmiş ve bilim dünyasında bir dönüm noktası yaşanmıştır.

Kompleks büyük dereceli sistemlerin analizine olan geleneksel yaklaşımlar faydasız olduğundan, temel düşünce, kesin olmamayı basit matematik terimleri ile göstermekti. Bulanık mantığın temelini oluşturan bulanık kümeler teorisi, belirsiz kavramların matematiksel olarak ifade edilmesidir. Bulanık kümeler teorisi ortaya atılıncaya kadar belirsizlikle ilgili matematiksel işlemler yalnızca olasılık teorisi ile modellenmiştir. Olasılık teorisindeki belirsizlik, olayın belli bir dağılıma bağlı olarak gerçekleşme ihtimali ile ilgilenir. Bulanık kümeler teorisindeki belirsizlik ise, bir kümenin sınırlarının kesin olarak tanımlanamaması ile ilgilidir.

Bulanıklık ile rastlantısallık farklı kavramlardır ve belirsizliğin farklı yönleri ile ilgilenir.

Bulanık mantığın klasik matematiksel dayanağı, araştırmacılar arasında büyük ilgi yaratmış ve bu mantık, biyoloji, işletme, ekonomi, mühendislik ve tıbbi teşhisi içeren birçok alanda uygulanmaya başlanmıştır (Sato et al, 1997).

Bir çeşit çok değerli küme kuramı olan bulanık kümeler kuramı, belirsizliğin bir çeşit formülleştirmesidir ve temel amacı, belirsizlik ifade eden, tanımlaması güç veya anlamı zor kavramlara üyelik derecesi atayarak onlara belirlilik getirmektir (Türkşen, 1985, Baykal ve Beyan, 2004). Kümedeki her bir birey, klasik çift değerli küme kuramlarında olduğu gibi "üye" ya da "üye değil" olarak değil, "bir dereceye kadar üye" olarak görülür.

Bulanık küme kavramı, duyarlılığın arttırılması açısından, klasik kümelerinkine göre daha uygun olan yeni bir araç sağlıyor olarak görülebilir. Getirdiği yaklaşım, klasik küme kuramlarında kullanılan üyelik kavramını bir kenara bırakıp yerine tamamen yenisini koymak değil, iki-değerli üyeliği çok-değerliğe taşıyarak genelleştirmektir.

Bir bulanık küme öğesi aynı değişken özelliğine sahip olmak üzere başka bir kümenin de öğesi olabilir. Sonuç olarak bulanık küme için "klasik kümelere göre belirsiz, bulanık sınırı olan kümedir", denebilir.

Bulanık üyelik kavramı ile sözel terimler tanımlanabilir. Göz önünde tutulan bir bulanık kelime veya ifadenin temsil ettiği sayısal aralık o ifade hakkında bilgi sahibi kişiler tarafından belirlenebilir. Bu aralıkların sınırlarında Aristoteles mantığına göre katı kararlar

alınmalıdır. Bu aralıkların arasındaki geçiş kısımları birbirinin devamı olmayabilir ve örtüşme söz konusu olabilir.

Bulanık mantık yaklaşımının en fazla kullanıldığı alanlardan birisi, bulanık kural tabanlı sistemlerdir. Büyük ölçülü veri tabanlarını tarayarak bilgi oluşturma ve karar verme teknolojilerinden olan veri madenciliği (data mining) modern bir teknoloji olarak son zamanlarda yaygın olarak kullanılmaktadır (Agrawal, et al.,1993; Fu et al.,1998; Kandel, et al., 2001). Kümeleme, belirleme, sınıflandırma gibi yöntemler bu teknolojide sık kullanılan önemli yöntemlerdendir (Dumitrescu et al., 2000; Wolkenhauer, O., 2001; Nasibov, 2000, 2002(a), 2002(b)). Bir nesneler kümesinin denetlenemeyen sınıflandırmasının (kümelemesinin) esas amacı, nesne çiftleri arasında benzerliğe veya farklılığa dayalı "doğal alt gruplar" oluşturmaktır. Doğal alt grupların tanımına veya anlamına bağlı olarak, günümüzde çeşitli kümeleme yöntemleri bulunmaktadır (Dunn, 1973; Bezdek, 1980(a),1981; Gordon, 1981; Bezdek et al., 1981, 1987; Bobrowski ve Bezdek, 1991; Chepoi ve Dumitrescu, 1992; Hathaway ve Bezdek, 1993; Zahid et al., 2001; Belace et al., 2002; Runkler ve Bezdek, 2003). Veri setine bağlı olarak bu yöntemlerden herhangi biri daha etkili performans sağlayabilir. Mevcut veri setinde hangi kümeleme yönteminin kullanılmasının daha doğru olacağının belirlenmesi önemli sorunlardan biridir. Kümeleme algoritmalarının birçoğunda küme sayısı önceden belirlenmelidir. Bunun için de farklı yöntemler oluşturulmuştur (Yager ve Filev, 1994; Pal ve Bezdek,1995; Velthuizen et al., 1997; Sugar ve James, 2003).

Bu tez çalışmasında, çeşitli veri setleri için literatürde sıklıkla kullanılan yöntemlerin sonucunu karşılaştırmak ve görsel olarak değerlendirebilmek imkanını sağlayan program sistemi oluşturulmuştur (Nasibov vd., 2004). Sistemin esas özelliği bir veya iki ölçülü veri setini, kümeleme sonuçlarını, başlangıç ve sonuç küme merkezlerini görsel olarak ekranda görüntüleme olanağı vermesidir. Bunun yanı sıra, isteğe bağlı veri setleri türeterek benzer veri setini görsel olarak ekranda oluşturma imkanı da sağlanmaktadır.

Sunulan program sisteminin özellikleri aşağıdaki gibi sıralanabilir:

- çeşitli kümeleme yöntemlerinin seçilebilmesi;
- bulanıklığı yansıtan üs (m) parametresinin seçilebilmesi;
- başlangıç kümeleri oluşturma yöntemlerinin seçilebilmesi;
- başlangıç küme sayısının direkt verilebilmesi;
- çeşitli küme geçerliliği endekslerine dayalı olarak küme sayısının otomatik olarak belirlenebilmesi;
- bir ve iki ölçülü kümeleme durumunda sonuçlarının görsel olarak ekrana verilebilmesi;
- verilere bağlı olarak ekranın ölçeklerinin ayarlanabilmesi.

Çalışmanın 2. Bölümünde, literatürde yer alan kümeleme yöntemlerine değinilmiş ve bulanık c-ortalamalar (Fuzzy c–means - FCM) kümeleme yöntemi ele alınmıştır. Küme merkezleri ve üyelik derecelerinin elde edilişine yer verilerek, bulanık c-ortalamalar kümeleme algoritması anlatılmıştır. Ayrıca, bulanık c-ortalamalar

kümeleme algoritmasının genelleştirilmiş haline (GFCM) değinilmiş ve hem FCM, hem de GFCM algoritmalarının sonsuz ailesi incelenmiştir.

3. Bölümde, optimal küme sayısının belirlenmesi için literatürde yer alan küme sayısı belirleme ve küme geçerliliği yöntemlerine değinilmiştir. Ayrıca, literatürde sıklıkla karşılaşılan ve program sistemi içinde kullanılan küme geçerliliği yöntemleri incelenmiştir.

4. Bölümde, kümeleme algoritmalarının önemli adımlardan birisi olan başlangıç kümelerin belirlenmesi problemi ele alınarak, programda kullanılan başlangıç kümelerin belirlenmesi yöntemleri anlatılmıştır.

5. Bölümde, oluşturulan program sisteminin çalışma prensipleri ele alınmıştır. Program sisteminde kullanılan formlar ve menüler incelenerek, formlardaki buttonların kullanım amaçları ayrıntılı olarak anlatılmıştır.

6. Bölümde, tezde geliştirilmiş olan program sisteminde bulunan "Visual Input" teknolojisi kullanılarak türetilen 2 ile 7 arasında değişen kümelere sahip veri setleri ile Anderson (1935) ve Fisher'in (1936) biyometrik IRIS veri setine çeşitli küme geçerliliği yöntemleri ve başlangıç küme merkezi belirleme yöntemleri ile kümeleme uygulanmış ve sonuçlar tablolar halinde sunulmuştur.

2. BULANIK KÜMELEME YÖNTEMLERİ

Çoğu gerçek dünya sınıfı kesin olmaktan öte bulanıktır. Yani, sınıflandırma problemleri ile ilgilenirken bulanık küme teorisini düşünmek doğaldır.

Bulanık kümelerin kümelemede kullanımını ilk kez Bellman, Kalaba ve Zadeh önermişlerdir (1966). Ruspini, veri setinin küme yapısını anlatmak için özel kriter fonksiyonları ile bulanık bölünme (bölümleme) kavramı tanıtmış ve optimum bulanık bölünmeyi hesaplamak için bir algoritma önermiştir (1969, 1970, 1973). Negoita, küme-tabanlı bilgi düzeltme sistemini anlatmak amacıyla bulanık kümeler için bir ayırma teoremi kullanmıştır (1973). Dunn, minimum-varyans kümeleme yöntemini bulanık ISODATA kümeleme yöntemine genellemiştir (1973, 1974). Bezdek, Bulanık c-Ortalamalar algoritmaları olarak bilinen algoritmaların sonsuz ailelerini elde etmek için Dunn'ın yaklaşımını genellemiştir (1973, 1974). Dunn ve Bezdek (1975), kümelerin şekillerindeki farklılıkları dikkate alarak, ilgili algoritmalar önermişlerdir.

Bulanık kümeleme yöntemleri uzaklığa dayalı (distance-based) ve ilişkiye dayalı (relation-based) olmak üzere iki çeşittir. Uzaklığa dayalı kümeleme yöntemleri birimlerin her bir kümeye aitlik dereceleri ile bu kümenin ağırlık merkezine olan uzaklıklarının çarpımlar toplamının minimum yapılmasına dayanır. Bazı durumlarda kümeleme sırasında kullanılmak üzere, birimler ve kümelerin ağırlık merkezleri arasındaki uzaklıkları bulmak mümkün olmayabilir. Bu gibi durumlarda amaçlara uygun olarak, birimler arasındaki ilişkilerden yararlanılabilir.

Kümeleme aslında, veri kümesinin uygun benzerlik ölçülerine göre homojen sınıflara ayrılması ile ilgilenir. Herhangi bir kümeye ait olan birimler benzer, farklı kümelerdeki birimler ise mümkün olduğunca farklıdır. Klasik kümeleme analizinde, farklı kümelerin sınırları keskindir ve bir birim yalnızca bir kümeye ait olabilir. Pratikte, veriler genelde iyi bir dağılıma sahip olmazlar, yani sınırlar kesin olarak tanımlanamaz. O zaman, bir birim farklı üyelik dereceleri ile birden fazla kümeye ait olabilir (Kendall ve Stuart, 1976; Jang et al., 1997). Bu durum klasik kümeleme analizine benzer olarak bulanık kümelemede de farklı yöntemler ile ifade edilebilir. Bu çalışmada değinilecek olan yöntem, Dunn tarafından önerilip Bezdek tarafından geliştirilen Bulanık c-Ortalamalar yöntemidir (Fuzzy c-Means – FCM) (Dunn, 1973; Bezdek, 1974).

Bu bölümde bulanık sınıfın küme temelini belirlemek için bazı algoritmalar ele alınacaktır. Algoritmalar, kriter fonksiyonunun minimizasyonuna dayanmaktadır. Kriter fonksiyonlar, bulanık sınıfa göre yerel uzaklıklar kullanılarak elde edilmektedir (Dumitrescu, 1986, 1987, 1988).

Bulanık sınıfın küme temelini arama, aşağıda verildiği gibi farklı durumlarda gerekli olmaktadır:

i) Veri bilgisi daha arındırılmış bir analiz gerektirdiğinde, bir-seviyeli veri sınıflandırmanın yetersizliği,

ii) Hiyerarşik olarak düzenlenmiş verinin yapısını elde etme gereksinimi,

iii) Bir öğrenme seti kullanılarak bulanık hiyerarşik sınıflandırıcının tasarlanması,

iv) Hiyerarşik kümeleme yönteminin tasarımı. Böyle bir yöntem, özellikle küme sayısının bilinmediği durumlarda ilgi alanına girer.

Bulanık sınıfın alt küme yapısını belirlemek için Bulanık c-Ortalamalar algoritmasının genelleştirilmiş türüne (GFCM) değinilmiştir.

Kriterin biraz değiştirilmesiyle, GFCM amaç fonksiyonlarının sonsuz ailesi elde edilecek ve sınıf prototipleri olarak kümeleri kullanan bulanık kümeleme yöntemi incelenecektir. Kümelerin geometrik şekilleri ile ilgili uygun bilgi olmadığında veya nokta prototip temsilcisi çok kabaca verildiyse temsilci uygundur. Euclidean kümelemeye bir alternatif olarak L_p metriği incelenecektir.

Parametrelerden bir tanesini, örneğin üyelik derecelerini, elemek için amaç fonksiyonları tekrar formüle edilebilir. Böylece, daha basit amaç fonksiyonları elde edilir. Amaç fonksiyonlarının yeniden formüle edilmiş şekli genel optimizasyon yöntemleri kullanılarak düzeltilebilir.

2.1. Bulanık Sınıfın Küme Temeli ve Nokta Prototipleri

$X = \{x_1, x_2, ..., x_n\}$, $x_j \in R^s$, veri seti olsun. X'teki kümelerin optimal sayısının bilindiği varsayılsın. X'teki noktaların kümeleri, bulanık kümeler ile tarif edilebilirse, bu bulanık kümelerin ayrık olması gerekmektedir. Bu yüzden, X'in küme yapısı, birleşimi X olan, yani X'in bulanık bölünmeleri olan, ayrık kümeler ile tanımlanabilir.

C, X'ten gelen nesneler sınıfına uyan bulanık küme olsun. C'nin küme temeline sahip olması da mümkündür. Bu küme temeli, C'nin bulanık bölünmesi olarak görülebilir.

C'nin c-atomlu bulanık bölünmelerinin ailesi $F_c(C)$ ile gösterilsin. C'deki alt kümelerin sayısı c ise, C'nin küme temeli, $F_c(C)$'den $P=\{u_1,u_2,...,u_c\}$ ile ifade edilebilir. Her u_i bulanık sınıfı, belli bir temsil uzayından v_i prototipi ile ifade edilebilir. Bunu takiben, temsil uzayının s-boyutlu R^s Euclidean uzayı olduğunu varsayılsın. Eğer, kümeler yaklaşık küresel veya elipsoid şeklinde ise, prototipler R^s'de noktalardır. Örneğin, v_i, u_i bulanık sınıfının ağırlık merkezi olabilir. Eğer kümeler doğrusal şekildeyse, prototipler doğrular veya diğer doğrusal türlerdir. Çeşitli geometrik prototiplerin konveks kombinasyonları da düşünülebilir.

M, simetrik pozitif tanımlı matris ve T transpozu göstermek üzere, iç-çarpım etkili $\|.\|$ normu aşağıdaki gibi tanımlansın:

$$\|.\| = (x,x) = x^T M x$$

ve d, norm-etkili uzaklık olsun:

$$\begin{aligned} d^2(x,y) &= \|x-y\|^2 \\ &= (x-y)^T M(x-y) \end{aligned}$$

d_i, u_i bulanık sınıfına göre uzaklık ise:

$$d_i(x_j,x_k) = \min(u_{ij},u_{ik})\ d(x_j,x_k) \quad \forall x_J, x_k \in X$$

d_i'nin, R^s'ye genişlemesi, d_i^*'dır.

m, X'ten gelen vektörlerin ortalama vektörü olsun. X'ten gelen noktaların uzaklıklar kareleri toplamının minimum olduğu durum için, m'nin R^s'de tek nokta olduğu bilinmektedir. Bulanık sınıftaki ortalama vektörü için de benzer özellik olması istenmektedir.

v'nin R^s'de bir nokta olduğunu ve X'ten olmadığı varsayılsın. u_i bulanık sınıfında, v ve X'teki noktalar arasındaki uzaklıkların kareleri toplamı:

$$F_i(\mathrm{v}) = \sum_{j=1}^{n} d_i^{*2}(x_j, v)$$
$$= \sum_{j=1}^{n} u_{ij}^2 \; d_i^2(x_j, v)$$

olur. F_i fonksiyonunun minimum noktası, u_i bulanık sınıfının ortalama noktası olarak yorumlanabilir.

TEOREM 1: $F_i(\mathrm{v}) = \sum_{j=1}^{n} u_{ij}^2 \left\| v - x_j \right\|^2$

şeklinde olan $F_i : (R^s \setminus X) \rightarrow R$ fonksiyonunun tek minimum noktası aşağıdaki gibi belirlenen m_i'dir:

$$m_i = \frac{\sum_{j=1}^{n} u_{ij}^2 \; x_j}{\sum_{j=1}^{n} u_{ij}^2}$$

2.1.1. Kümeleme için Kriter Fonksiyonu

Kümeleme, sınıf-içi benzerlik (veya kohezyon) ölçüsü veya benzemezlik ölçüsü kullanılarak türetilen bir kriter fonksiyonu seçilerek uygulanabilir. Önce, benzemezlik kavramına dayalı bir kriter fonksiyonu ele alınsın. Benzemezlik, bulanık sınıfa göre karesi alınmış uzaklık olarak tanımlanabilir.

C bulanık sınıfının küme temelinin, C'nin bulanık bölünmesi $P=\{u_1,u_2,...,u_c\}$ ile ifade edildiği varsıldığında, u_{ij}, j-nci elemanın, i-nci kümeye ait olma derecesini göstermek üzere;

$$\sum_{j=1}^{n} u_{ij}^2 = C_j \quad , \quad j=1,2,...,n$$

koşulu sağlanır. $v_i \in R^s$, u_i bulanık sınıfının prototipi olsun. Eğer, v_i, X veri setinden bir nokta ise, prototipin u_i bulanık sınıfına en yüksek üyeliğe sahip olduğu söylenebilir. Yani,

$$u_i(v_i) = \max_j \; u_{ij}$$

olur. Bu durumda,

$$\begin{aligned} d_i^*(x_j,v_i) &= \min(u_{ij},u_i(v_i))d(x_j,v_i) \\ &= u_{ij}\; d(x_j,v_i) \end{aligned}$$

olur. Eğer, v_i, X'ten değilse, o zaman x_j ve v_i arasındaki $d_i^*(x_j,v_i)$ uzaklığı da benzer ifadeye sahip olur:

$$d_i^*(x_j,v_i) = u_{ij}\, d(x_j,v_i)$$

Bir veri noktası ile v_i prototipi arasındaki "benzemezlik", u_i'ye göre açıklanacaktır. D_i, karesi alınmış d_i^* olarak seçilecektir.

$$D_i = (x_j, v_i) = d_i^{*2}(x_j, v_i)$$
$$= u_{ij}^2 \, d^2 \, (x_j, v_i)$$

u_i bulanık sınıfı ve onun prototipi arasındaki "yetersizlik",

$$I(u_i, v_i) = \sum_{j=1}^{n} \mathrm{D_i}(u_i, v_i)$$

olarak yazılabilir. $v = (v_1, v_2, ..., v_c)$'nin, P bulanık bölünmesinin temsili olduğu varsayılsın. P bölünmesi ile onun temsili v arasındaki yetersizlik;

$$J(P, v) = \sum_{i=1}^{c} I(u_i, v_i)$$

olur. O zaman,

$$J(P, v) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \, d^2 \, (x_j, v_i)$$
$$= \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \, \| x_j - v_i \|^2$$

olarak yazılan $J : F_c(C) \; x \; R^{sn} \rightarrow R$ kriter fonksiyonu elde edilir.

2.1.2. Değişen Optimizasyon Yöntemi

Önceki kriter fonksiyonu, "En Küçük Karesi Alınmış Hata" (Least Squared Error (LSE)) tipindeydi. Optimal bulanık bölünme ve onun temsilcisi;

$$\begin{cases} \text{minimize } J(P,v) \\ P \in F_c(C) \\ v \in R^s \end{cases} \tag{2.1}$$

minimizasyon problemine yerel çözüm olarak elde edilebilir. Bu problemin kesin çözümü işlemsel olarak pratik değildir. Probleme yaklaşık bir çözüm elde etmek amacıyla, $J(P,\bullet)$ ve $J(\bullet,v)$ fonksiyonlarının minimizasyonuna dayalı iteratif bir yöntem kullanılabilir. Bu yönteme "Değişen Optimizasyon Tekniği" denir. (2.1) problemi aşağıdaki iki problem ile değiştirilebilir.

$$\begin{cases} \text{minimize } J(P,v) \\ P \in F_c(C) \\ (v \text{ sabit}) \end{cases} \tag{2.2}$$

$$\begin{cases} \text{minimize } J(P,v) \\ v \in R^{sc} \\ (P \text{ sabit}) \end{cases} \tag{2.3}$$

İlk probleme çözüm için:

$$I_j = \{ i \mid 1 \le i \le c \,,\ \ d(x_j, v_i) = 0 \}$$

ve

$$\bar{I}_j = \{1,2,...,c\} - I_j$$

olarak gösterilebilir.

TEOREM 2: *J(•,v)* fonksiyonunun kümeleme yapısı P'ye göre kritik noktası aşağıdaki gibi belirlenir:

$$i \in \bar{I}_j \Rightarrow u_{ij} = \frac{\sum_{j=1}^{n} u_{ij}^2}{\sum_{k=1}^{c} \frac{d^2(x_j, v_i)}{d^2(x_j, v_k)}}, \quad 1 \le j \le n \qquad (2.4)$$

ve

$$i \in I_j \Rightarrow u_{ij} = 0, \quad 1 \le j \le n. \qquad (2.5)$$

TEOREM 3: *J(P, •)* fonksiyonunun $v \in R^{sn}$ değişkenine göre yerel minimumu,

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^2 \, x_j}{\sum_{j=1}^{n} u_{ij}^2} \qquad (2.6)$$

vektörüdür.

2.2. Genelleştirilmiş Bulanık c-Ortalamalar Algoritması

Bu kesimde, bulanık sınıfın küme temelini belirlemek için temel genelleştirilmiş bulanık c-ortalamalar (GFCM) algoritması incelenecektir.

$X = \{x_1, x_2, ..., x_n\}$, $x_j \in R^s$ veri seti ve C, X'te bir bulanık küme olsun. Aşağıdaki varsayımların olduğu varsayılsın:

i) C, X 'ten bir noktalar kümesini temsil eder.

ii) C'nin, $P=\{u_1,u_2,...,u_c\}$ bulanık bölünmesi ile tanımlanan küme temeli vardır.

iii) C'nin alt kümeleri sayısı c bilinmektedir.

Temel GFCM algoritması, bulanık bölünme için keyfi bir çözümle başlar. (2.6) eşitliği kullanılarak $v_1,v_2,...,v_c$ prototipleri; (2.4) ve (2.5) eşitlikleri kullanılarak C'nin yeni bulanık bölünmesi hesaplanır. (2.4), (2.5) ve (2.6) eşitlikleri iteratif olarak kullanılır ve iterasyon, iki başarılı bulanık bölünme yeteri kadar yakın olduğunda durur.

İki bulanık bölünme arasındaki uzaklığı ölçmek için, her P bölünmesi ile bir $c\times n$ boyutlu Q matrisi ilişkilendirilebilir.

$$Q_{ij}=u_{ij} \quad , \quad i=1,2,...,c \; ; \quad j=1,2,...,n$$

Eğer Q_i, P_i 'nin matris temsilcisi ise, P_1 ve P_2 arasındaki uzaklık:

$$d(P_1,P_2)=\|Q_1-Q_2\|$$

olur. P_r, r.nci iterasyondaki bulanık bölünme ise, ε kabul edilebilir hata olmak üzere; m.nci iterasyonda

$$d(P_m,P_{m-1})<\varepsilon$$

olduğunda süreç durur.

4 adımda gerçekleştirilen GFCM algoritması aşağıda verilmiştir:

GENELLEŞTİRİLMİŞ BULANIK c-ORTALAMALAR (GFCM) ALGORİTMASI

ADIM 1:

- C 'deki alt kümelerin sayısı c seçilir.
- Kabul edilebilir hata ε (örn: $\varepsilon = 10^{-5}$) seçilir.
- R^s üzerinde skaler çarpım sabitlenir. (M matrisi seçilir.)
- P_1 ile ilgili matris olarak Q_1 olsun.

ADIM 2 :

- (2.6) eşitliği kullanılarak, Q_1 matrisine dayalı v_i, $(i = 1,...,c)$ prototipleri hesaplanır.

ADIM 3:

- (2.4) ve (2.5)'te verilen kurallar kullanılarak C'nin yeni bulanık bölünmesi P_2 hesaplanır.

ADIM 4 :

- P_2 ile ilgili matris olarak Q_2 olsun. $\| Q_1 - Q_2 \| < \varepsilon$ ise DUR!

  Aksi halde, $P_1 = P_2$ ve $Q_1 = Q_2$ alınır ve Adım 2'ye geçilir.

2.3. Küme Prototipleri ile Bulanık Kümeleme

Şu ana kadar düşünülen bulanık kümeleme yöntemleri metrikti, yani uygun bir uzaklık ve geometrik sınıf prototipi kullanılarak bir amaç fonksiyonu tanımlanıyordu. Bazı problemlerde, veri setinin her (x_j, y) çifti için tanımlanan benzemezlik ölçüsü kullanılarak elde edilen kriter fonksiyonu kullanılır, yani metrik olmayandır. Bu durumda, geometrik prototip tanımlanamaz.

Bu durumla baş etmek için, Roubens (1978, 1982) bulanık sınıfı bir bulanık olmayan küme ile temsil etmeyi önermiştir. X veri setinin küme temelini tanımlayan bir bulanık bölünme, X'in (veya X'in bir alt kümesinin) bulanık olmayan bölünmesi ile temsil edilebilir. Bu kesimde Ruspini'nin yaklaşımına benzer bir yaklaşım ele alınacaktır.

d, $d : Y \times Y \rightarrow R$ fonksiyonu olmak üzere, bir pseudo-metrik uzay aşağıdaki aksiyomları sağlayan bir (Y, d) çiftidir:

i) $d(x_j, y) = d(y, x_j)$, $\forall x_j, y \in Y$

ii) $d(x_j, y) \geq 0$, $\forall x_j, y \in Y$

iii) $d(x_j, x_j) = 0$, $\forall x_j \in X$

$X = \{x_1, x_2, ..., x_n\}$, pseudo-metrik uzaydan bir veri seti olsun. $P = \{u_1, u_2, ..., u_n\}$, X'in bulanık bölünmesi olsun. Her u_i bulanık sınıfının bir $v_i \subseteq X$ alt kümesi ile temsil edildiği varsayılsın. v_i, u_i'nin küme temsilcisi veya çekirdeğidir (kernel). İki durum

düşünülebilir: $v=(v_1,v_2,...,v_c)$, X'in bulanık olmayan bölünmesi olabilir ya da v, X'in bir alt kümesi olan Y'nin bulanık olmayan bölünmesidir. İkinci durumda, tüm prototip özelliği gösteren durumlar düşünülmemiştir. Her iki durumda da v, P bulanık bölünmesinin temsilcisidir.

$u_i(x)$; x elemanının, $u_i(y)$; y elemanının i-nci bulanık sınıfa ait olma derecelerini göstermek üzere, u_i bulanık sınıfına göre, d ile etkilenen pseudo-metriğin aşağıdaki şekilde olduğu varsayılır:

$$d_i(x,y)=u_i(x)u_i(y)\ d(x,y)$$

x_j ve v_i prototipi arasındaki $D_i(x_j,v_i)$ benzemezliği:

$$D_i(x_j,v_i)=\frac{1}{card\ v_i}\sum_{y\in v_i} d_i^2(x_j,y)$$

şeklinde tanımlanırsa, P bulanık bölünmesi ile temsilcisi v arasındaki yetersizlik:

$$J(P,v)=\sum_{i=1}^{c}\sum_{j=1}^{n} D_i(x_j,v_i)$$

$$J(P,v)=\sum_{i=1}^{c}\sum_{j=1}^{n}\frac{1}{card\ v_i}u_i^2(x_j)u_i^2(y)\ d^2(x_j,y)$$

olarak ifade edilebilir.

$$D_{ij}=\frac{1}{card\ v_i}\sum_{y\in v_i} u_i^2(y)\ d^2(x_j,y) \qquad (2.7)$$

olarak alınırsa, D_{ij}, x_j'nin u_i 'ye üyeliğini içermez ve

$$D_i(x_j, v_i) = u_{ij}^2 \, D_{ij}$$

şeklinde yazılabilir. D_{ij}'ye, x_j ve küme temsilcisi v_i arasındaki mutlak (veya ağırlıklandırılmamış) uzaklık denilebilir. O zaman kriter fonksiyonu aşağıdaki gibi elde edilir:

$$J(P, v) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 D_{ij}$$

Bu kriter fonksiyonunu minimize etmek için, iteratif bir yöntem kullanılır. Bu yöntemde, D_{ij}'yi hesaplamak için önceki iterasyondaki üyelikler kullanılır.

Elde edilen kriter fonksiyonuna göre optimal bulanık bölünme aşağıdaki üyelik dereceleri ile belirlenir:

$$u_{ij} = \frac{1}{\sum_{k=1}^{n} \frac{D_{ij}}{D_{ik}}} \quad , \quad \forall i, j \tag{2.8}$$

6 adımda gerçekleştirilen küme prototipleri ile bulanık kümeleme algoritması aşağıda verilmiştir:

KÜME PROTOTİPLERİ İLE BULANIK KÜMELEME ALGORİTMASI

ADIM 1:

- $l = 1$ belirlenir.
- X 'in başlangıç bölünmesi $P_l = P_0 = \{u_1, u_2, ..., u_c\}$ seçilir.
- X 'in başlangıç bulanık olmayan bölünmesi $v_0 = v_1 = \{v_1, ..., v_c\}$ seçilir.

ADIM 2 :

- $D_{ij} = \frac{1}{card\, v_i} \sum_{y \in v_i} u_i^2(y)\, d^2(x_j, y)$, $1 \le i \le c$; $1 \le j \le n$ hesaplanır.

ADIM 3 :

- $u_{ij} = \frac{1}{\sum_{k=1}^{n} \frac{D_{ij}}{D_{ik}}}$, $1 \le i \le c$; $1 \le j \le n$

kullanılarak X 'in yeni bulanık bölünmesi P_{l+1} hesaplanır:

ADIM 4 :

- $D'_{ij} = \frac{1}{card\, v_i} \sum_{y \in L_i} u_i^2(y)\, d^2(x_j, y)$

kullanılarak D_{ij} güncellenir.

ADIM 5 :

- x_j , v_i'ye atanır $\Leftrightarrow D'_{ij} = \min_k D'_{ik}$

kuralına göre, bulanık olmayan bölünme güncellenir. Yeni bulanık olmayan bölünme v_{l+1} olur.

ADIM 6 :

- $v_{l+1} = v_l$ ise DUR! Aksi halde $l=l+1$, $D_{ij} = D'_{ij}$ kurulur ve Adım 3'e geçilir.

C bir bulanık sınıf ve v_C onun küme prototipi olsun. C'nin küme temelini elde etmek için, C'nin bulanık bölünmesi aranmalıdır. Bu bulanık bölünme, v_C'nin bulanık olmayan bölünmesi ile temsil edilir. Bu amaçla, önceki algoritmanın basit bir genellemesi kullanılabilir.

2.4. Bulanık *c*-Ortalamalar Kriter Fonksiyonları Ailesi

Bu kesimde, FCM algoritmalarının sonsuz aileleri ve limit özellikleri ele alınacaktır.

2.4.1. FCM Algoritmalarının Sonsuz Ailesi

$P = \{u_1, u_2, ..., u_c\}$, X veri setinin bir bulanık bölünmesi ve $v_i \in R^s$, u_i bulanık sınıfının prototipi olsun. Bulanık kümeleme için kriter fonksiyonu genellikle, $m > 1$ ağırlık üssü olmak üzere, aşağıdaki gibi verilir:

$$J_m(P, v) = \sum_{i=1}^{c} \sum_{j=1}^{n} (u_{ij})^m \, d^2(x_j, v_i)$$

m parametresi, sonuçlanan bölünmenin bulanıklık derecesini kontrol eder. İlk terim bulanık bölünmeyi gösterirken ve v sabit iken

indirgenmiş amaç fonksiyonu $J_m(\bullet,v)$ ile gösterildiğinde, $J_m(\bullet,v) \rightarrow$ min optimizasyon probleminin çözümü, aşağıdaki üyelik dereceleri ifadesinin elde edilmesini sağlar:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left[\frac{d^2(x_j,v_i)}{d^2(x_j,v_k)}\right]^{2/(m-1)}} \quad , \; 1 \le i \le c \; ; \; 1 \le j \le n$$

$J(P,\bullet) \rightarrow$ min optimizasyon probleminin çözümü ile elde edilen prototipler aşağıdaki gibidir:

$$v_i = \frac{\sum_{j=1}^{n}(u_{ij})^m x_j}{\sum_{j=1}^{n}(u_{ij})^m} \quad , \quad 1 \le i \le c$$

Elde edilen u_{ij} ve v_i ifadelerini kullanan değişen optimizasyon, FCM algoritmasının standart halidir. m parametresi, sonsuz değerleri tahmin edebildiği için, bulanık kümeleme algoritmalarının sonsuz ailesi elde edilir.

2.4.2. FCM Ailesinin Limit Özellikleri

$m \rightarrow 1$ ise, bulanık c-ortalamalar algoritması, bulanık olmayan c-ortalamalar çözümüne; üyelik dereceleri de 0 ve 1'e yakınsar (Bezdek, 1981).

$m \rightarrow \infty$ ise, $u_{ij} \rightarrow 1/n$, $1 \le i \le c$; $1 \le j \le n$, olur ve algoritma ile türetilen tüm prototipler, $x_1,...,x_n$ vektörlerinin ağırlık merkezine yakınsar. Yani, $m(X)$,

$$m(X) = \frac{\sum_{j=1}^{n} x_j}{n}$$

X'in ağırlık vektörü olmak üzere $v_i \to m(X)$ olur. $u_{ij} \to 1/n$ durumu, maksimum bulanıklık derecesidir.

Genel olarak, m ne kadar büyükse, üyelik dereceleri ile ilgili bulanık bölünmeler o kadar bulanıktır. m, 1'e yakınsadığında, elde edilen bulanık bölünmeler bulanık olmayan bölünmelere yaklaşır.

Genellikle, m=2 değeri seçilir. Bu da, kriter fonksiyonunun bulanık sınıflara göre karesi alınmış uzaklıkların toplamları olarak yazılan çıkarımla uyumludur.

2.5. GFCM Algoritmalarının Sonsuz Ailesi

Elde edilen kümeleme algoritmalarının sonsuz ailesi, C bulanık sınıfının alt küme yapısını belirlemek için genellenebilir.

C, X'te bir bulanık küme olsun. C'nin c-tane alt küme içerdiği ve bu kümelerin C'nin bölünmeleri olan $u_1, u_2, ..., u_c$ ile gösterildiği varsayılsın. J_m kriter fonksiyonunun minimizasyonu aşağıdaki üyelik derecelerini verir:

$$u_{ij} = \frac{C_j}{\sum_{k=1}^{c} \left[\frac{d^2(x_j, v_i)}{d^2(x_j, v_k)} \right]^{1/(m-1)}} \quad , \quad \begin{matrix} i = 1,..., \mathrm{c} \\ \mathrm{j} = 1,..., n \end{matrix}$$

v_i, $i = 1,...,c$ prototipleri, ilgili sınıfların ortalama vektörleridir. Elde edilen u_{ij} ve v_i değerleri ile değişen optimizasyon Genelleştirilmiş Bulanık c-Ortalamalar algoritmalarının sonsuz ailesine ulaşır.

Hathaway ve Bezdek, (Hathaway ve Bezdek, 1995) kümeleme amaç fonksiyonlarının yeniden formüle edilmeleri üzerinde çalışmışlardır. Amaçları, bulanık kümeleme ile ilgili optimizasyon problemlerini yeniden formüle ederek, genel optimizasyon yöntemlerinin uygulanmasını sağlamaktır. Orijinal ve yeniden formüle edilmiş amaç fonksiyonları tamamen denktir. Yeniden formüle edilmiş türler, P'nin sağladığı optimum (gerekli) koşulları P bulanık bölünmesinin yerine koyarak elde edilmektedir.

$$J_m(P,v) = \sum_{i=1}^{c}\sum_{j=1}^{n}(u_{ij})^m \; d^2(x_j,v_i)$$

kriter fonksiyonunda,

$$v_m(P) = \sum_{i=1}^{c}\sum_{j=1}^{n}(u_{ij})^m \; d^2(x_j,v_i)$$

fonksiyonunun minimize edilmesiyle aşağıdaki üyelik dereceleri bulunur:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left[\frac{d^2(x_j,v_i)}{d^2(x_j,v_k)}\right]^{1/(m-1)}}, \qquad \begin{matrix} i = 1,...,\mathrm{c} \\ \mathrm{j} = 1,...,n \end{matrix}$$

$$u_{ij}^{m-1} = \frac{1}{d^2(x_j,v_i)}\sum_{k=1}^{c}\left[\left(\frac{1}{d^2(x_j,v_k)}\right)^{1/(m-1)}\right]^{-(m-1)}$$

alınırsa ve u_{ij}^{m-1} yerine yazılırsa:

$$J_m(P,v) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}\, u_{ij}^{m-1}\, d^2(x_j, v_i)$$

olur ve

$$J_m(P,v) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij} \frac{1}{\left[\sum_{i=1}^{c}\left(\frac{1}{d^2(x_j, v_k)}\right)^{1/(m-1)}\right]^{(m-1)}}$$

elde edilir. Ayrıca,

$$J_m(P,v) = \sum_{j=1}^{n} \frac{\sum_{i=1}^{c} u_{ij}}{\left[\sum_{i=1}^{c}\left(\frac{1}{d^2(x_j, v_k)}\right)^{1/(m-1)}\right]^{(m-1)}}$$

yazılırsa, $\sum_{i=1}^{c} u_{ij} = 1 \quad , \quad \forall j = 1,...,n$ olduğundan;

$$J_m(P,v) = \sum_{j=1}^{n} \frac{1}{\left[\sum_{i=1}^{c}\left(\frac{1}{d^2(x_j, v_k)}\right)^{1/(m-1)}\right]^{(m-1)}}$$

bulunur. Elde edilen ifade $R_m(v)$ ile gösterilecektir ve bu da FCM kriterinin yeniden formüle edilmiş halidir:

$$R_m(v) = \sum_{j=1}^{n}\left[\sum_{k=1}^{c}\left(\frac{1}{d^2(x_j, v_k)}\right)^{1/(m-1)}\right]^{1-m}$$

k indisi i ile değiştirilir ve $q = 1/(1-m)$ ile ifade edilirse;

$$R_m(L) = \sum_{j=1}^{n} \left[\sum_{i=1}^{c} d^2(x_j, v_i)^q \right]^{1/q}$$

olur. Yani, amaç fonksiyonu, genelleştirilmiş harmonik ortalamaların toplamı şeklinde yeniden formüle edilebilir. J_2 kriter fonksiyonunun yeniden formüle edilmiş şekli aşağıdaki gibi olur:

$$R_2(v) = \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{c} \frac{1}{d^2(x_j, v_i)}}$$

Değişen optimizasyonu kullanan her bulanık kümeleme algoritması bu şekilde yeniden formüle edilebilir.

GFCM kriter fonksiyonunu yeniden formüle etmek için optimuma uyan üyelik dereceleri $J_m(P,v)$'de yer değiştirilir (Krishnapuram ve Keller, 1993). u_{ij}^{m-1} için;

$$u_{ij}^{m-1} = \frac{C_j^{m-1}}{d^2(x_j, v_i)} \left[\sum_{k=1}^{n} \left(\frac{1}{d^2(x_j, v_k)} \right)^{1/(m-1)} \right]^{1-m}$$

yazılabilir. $J_m(P,v)$'de u_{ij}^{m-1} yerine yazılırsa;

$$J_m(P,v) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} \frac{C_j^{m-1}}{\left[\sum_{k=1}^{c} \left(\frac{1}{d^2(x_j, v_k)} \right)^{1/(m-1)} \right]^{m-1}}$$

olur. $\sum_{i=1}^{c} u_{ij} = C_j$ olduğundan;

$$J_m(P,v) = \sum_{j=1}^{n} \frac{C_j^m}{\left[\sum_{k=1}^{c} \left(\frac{1}{d^2(x_j, v_k)} \right)^{1/(m-1)} \right]^{m-1}}$$

olur. $R_m(v) = J_m(P,v)$ ile gösterilirse GFCM kriter fonksiyonunun yeniden formüle edilmiş şekli;

$$R_m(v) = \sum_{j=1}^{n} C_j^m \left[\sum_{i=1}^{c} d^2(x_j, v_i)^{1/(m-1)} \right]^{1-m}$$

olur. Yeniden formüle edilmiş kriter fonksiyonu, genelleştirilmiş harmonik ortalamaların ağırlıklı toplamıdır.

$m = 2$ için GFCM kriter fonksiyonunun yeniden formüle edilmiş şekli aşağıdaki gibidir:

$$R_2(L) = \sum_{j=1}^{n} C_j^2 \frac{1}{\sum_{i=1}^{c} d^2(x_j, v_i)}$$

3. OPTİMAL KÜME SAYISININ BELİRLENMESİ

Kümeleme analizinde sağlıklı ve anlamlı sonuçlara ulaşabilmek için önemli koşullardan biri küme sayısının belirlenmesidir. Küme sayısının belirlenmesinde yapılacak küçük bir hata, küme yapısı keskin olsa bile bu yapının doğru olarak belirlenmesini imkansız yapar. Yani, küme sayısının aranması, kümeleme analizinin temelini oluşturur ve tüm kümeleme yöntemlerinin en önemli sorunudur. Birçok biyolojik uygulamada, ortak bir yaklaşım, *n* örneklem biriminin aynı grup içindeki bireylerinin mümkün olduğunca benzer, farklı gruplardakilerin ise olabildiğince farklı olacağı grup sayısını aramaktır. Bölünmede bulunan grupların çalışılan kitledeki doğal gruplara uyması ve bölünmedeki grupların özelliklerinin bu doğal gruplarla ilgili çıkarsamalara öncülük etmesi umut edilmektedir. Böyle bir yaklaşımla, tıp, psikoloji, ziraat, ekoloji ve taksonomide sıklıkla karşılaşılmaktadır (Krzanowski ve Lai, 1988).

Literatürde küme sayısının belirlenmesi için pek çok yöntem vardır. 1985'te Milligan ve Cooper, 1996'da ise Hardy bu yöntemlerin ayrıntılı listesini vermiş ve yine 1996'da Gordon yöntemlerin performanslarını karşılaştırmıştır. İstatistik literatürü, Edwards ve Cavalli-Sforza (1965), Scott ve Symons (1971), Marriott (1971), Calinski ve Harasabasz (1974), Hartigan (1975), Krzanowski ve Lai (1988), Kaufman ve Rousseeuw (1990), Kass ve Raftery (1995), Frayley ve Raftery (1998), Tibshirani, Walther ve Hastie (2001), Sugar ve James (2003) tarafından küme sayısını bulmak için önerilen yöntemlerle zenginleşmiştir.

3.1. Küme Geçerliliği Yöntemlerinin Genel Değerlendirilmesi

Kümeleme analizinin temel problemlerinden biri algoritmanın kümeleme sonuçlarının değerlendirilmesidir (Yen ve Langari, 1999). Bu problem literatürde küme geçerliliği olarak bilinmektedir. Daha net olarak, küme geçerliliği problemi, kümeleme algoritmaları ile bulunan bölünmelerin ne kadar iyi olduğunu belirlemek için bir amaç kriteri bulmak olarak tanımlanabilir. Böyle bir kriter, aşağıdaki amaçlara ulaşılmasını sağlaması bakımından önemlidir:

a) Veri seti için en iyi küme sayısının belirlenmesi,

b) Veri seti için alternatif kümeleme algoritmalarının sonuçlarının karşılaştırılması,

c) Veri setinin herhangi bir doğal gruplama yapısının olup olmadığının belirlenmesi.

Bunlardan en önemlisi küme sayısının belirlenmesidir. Küme sayısını belirlerken yapılacak küçük bir hata bile, keskin olan küme yapısının dahi doğru olarak bulunmasını imkansızlaştırır. Küme geçerliliği endeksleri çoğu zaman temel amacı olan küme sayısının belirlenmesi olarak tanımlanır.

Metotları farklı olsa bile klasik ve bulanık kümeleme yöntemleri küme geçerliliği konusuna değinmeye ihtiyaç duyarlar. Literatürde pek çok küme geçerliliği yöntemi bulunmaktadır. Bezdek (1974,1975) bir bulanık bölünmenin kalitesi veya bulanık olmama ölçüsü olarak tanımlanan "bölünme katsayısı" (partition coefficient) ve bir bulanık bölünmenin ne kadar bulanık olduğunu hesaplayan sınıflandırma veya bölünme entropisi (classification entropy / partition entropy) denilen

geçerlilik fonksiyonları önermiştir. Windham (1981,1982), bölünmenin bulanık kümesine maksimum üyeliği kullanılan oran üssü (proportion exponent) ve bu yöntemin değiştirilmiş şekli olan ve bir bulanık bölünmeyi veri setinin küme yapısı olmayan benzer bir bulanık bölünmesi ile algoritma parametrelerine duyarlı olmayan bir şekilde karşılaştırmaya dayanan uniform veri fonksiyoneli (uniform data functional) küme geçerliliği yöntemlerini öne sürmüştür. Fukuyamo ve Sugeno (1989), küme merkezlerinin ortalama vektörden sapmalarını birleştiren bir geçerlilik yöntemi olan Fukuyamo-Sugeno küme geçerliliği ölçüsü tanımlamışlardır. Gath ve Geva (1989), ortalama bölünme yoğunluğu (average partition density) denilen, küme hacmine ve bölünme yoğunluğuna dayalı bir geçerlilik fonksiyoneli önermişlerdir. Xie ve Beni (1991), bir bulanık c-bölünmesi üzerinden ortalama kompaktlığını ve ayrılmasını ölçen ve $J_m(P,v)$ yetersizlik ölçüsü ile merkezler arası uzaklıkları kullanan bir küme geçerliliği endeksi önermişlerdir. Araki, Nomura ve Wakami (1993), daha önceden Davies ve Bouldin'in (1979) önerdiği klasik küme dağılımına ve küme ayrılmasına dayanan küme geçerliliği ölçüsünü temel alarak bulanık kümeler için Xie ve Beni'nin (1991) önerdiği ölçüye benzer bir geçerlilik endeksi önermişlerdir. Rezaee, Lelieveldt ve Reiber (1998), bir c-bulanık bölünmesinin kompaktlık ve ayrılmasını göz önüne alarak grup-içi ve gruplar-arası dağılımın oluşturulması (compose within and between scattering) için bir geçerlilik fonksiyoneli önermişlerdir. Zahid, Limouri ve Essaid (1999), bulanık kompaktlık ve ayrılma bilgilerini üyelik dereceleri ve verinin yapısal özellikleriyle birleştirerek bulanık ayrılma/bulanık kompaktlık oranı tanımlamışlardır. Dumitrescu,

Lazzerini ve Jain (2000), iki bulanık küme arasındaki uzaklık tanımını kullanarak, Dunn'ın (1974) klasik bölünme için tanımladığı ayrılma endeksi (separation index) yaklaşımını bulanık bölünmeler için genelleştirmişlerdir. Dumitrescu et al., (2000) küme sayısı ile artmayan bir geçerlilik ölçüsü olan ve noktaların sınıf merkezlerine ne kadar yakın olduklarını ölçen sınıf-içi inertia (within class inertia) geçerlilik yöntemi önermişlerdir. Hathaway ve Bezdek (2003), herhangi bir kümeleme algoritması ile üretilen kümeleri bulmaktan ziyade, bulunmuş olan kümeleri gösterme aracı olan ve çeşitli veriler arası uzaklıkları göstermek için dijital görüntüleri kullanan bir yöntem olan visual küme geçerliliği (visual cluster validity-*VCV*) yöntemini önermişlerdir.

3.2. Bölünme Katsayısı

$X = \{x_1, x_2, ..., x_n\}$ veri seti ve $P = \{u_1, u_2, ..., u_c\}$, X'in bir bulanık bölünmesi olsun. Bu bulanık bölünmenin klasik bölünmeye ne kadar yakın olduğunu ölçmek için, Bezdek (1974) "bölünme katsayısı" (partition coefficient) denilen bir geçerlilik fonksiyonu önermiştir.Bu fonksiyon, x_j'nin ne kadar iyi sınıflandırıldığının bir ölçüsü olarak,

$$s_j = \sum_{i=1}^{c} u_{ij}^2 \tag{3.1}$$

miktarını kullanır. s_j'nin maksimum değeri 1, minimum değeri $1/c$'dir. Bu nedenle, s_j uygun bir göstergedir. Çünkü x_j veri noktası u_i-nci küme merkezine yaklaştıkça, u_{ij}'nin değeri 1'e yaklaşır. Sonuç olarak, s_j, 1'e yaklaştıkça, maksimum değerine de yaklaşır. Aksine, x_j tüm

küme merkezlerinden ne kadar uzak olursa, s_j'nin değeri, olası en küçük değeri olan $1/c$'ye yaklaşır.

Bölünme katsayısı, X veri seti üzerinden s'_j'lerin ortalamasıdır ve V_{PC} sayısı olarak aşağıdaki gibi ifade edilebilir:

$$V_{PC} = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2 . \qquad (3.2)$$

Bölünme katsayısı V_{PC}'nin özellikleri aşağıdaki gibi sayılabilir:

(i) $\frac{1}{c} \le V_{PC} \le 1, \quad \forall P \in F_c(X);$

(ii) $V_{PC} = \frac{1}{c} \Leftrightarrow u_{ij}^2 = \frac{1}{c}, \quad \forall i, j;$

(iii) $V_{PC} = 1 \Leftrightarrow P, X$'in klasik bölünmesidir.

Bölünme katsayısı bir bulanık bölünmenin kalitesi veya bulanık olmama ölçüsü olarak tanımlanabilir. V_{PC} bire ne kadar yakın olursa, P bölünmesi o kadar iyidir.

Bulanık kümeleme algoritmasının sonuçları küme sayısı c'nin farklı değerlerine göre, bölünme katsayısı kullanılarak karşılaştırılabilir. En iyi bölünme, bölünme katsayısı en yüksek olandır. Ancak, bölünme katsayısının değerleri, verideki herhangi bir yapıdan bağımsız olarak uygulamada, c'nin değerlerine karşı duyarlıdır. Yani, en iyi bulanık bölünme, her zaman en yüksek bölünme katsayısı değeri ile seçilemeyebilir. Bunun için, "sınıflandırma entropisi" kullanılabilir.

3.3. Sınıflandırma Entropisi

Bezdek'in (1975) önerdiği sınıflandırma veya bölünme entropisi (classification entropy / partition entropy) bulanık bölünme ile ilgili belirsizliği ölçmedir (Dumitrescu et al., 2000). Yani, bir bulanık bölünmenin ne kadar bulanık olduğunu hesaplamaya yardımcı olur.

Sınıflandırma entropisi, $X = \{x_1, x_2, ..., x_n\}$ veri seti üzerinden, $a > 1$ olmak üzere,

$$h_j = -\sum_{i=1}^{c} u_{ij} \log_a u_{ij} \tag{3.3}$$

miktarının ortalamasını hesaplar. h_j, x_j'nin üyeliklerinin $u_1, u_2, ..., u_c$ bulanık sınıflarına nasıl yayıldığını ölçer. $P = \{u_1, u_2, ..., u_c\}$, X'in bir bulanık bölünmesi olmak üzere, P'nin sınıflandırma entropisi aşağıdaki gibi tanımlanan V_{CE} sayısıdır.

$$V_{CE} = -\frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} \log_a u_{ij} \,. \tag{3.4}$$

İyi bir küme, sıfıra yakın bir entropi değeri ile belirlenir. Öte yandan, entropinin ortalama değeri küme sayısı ile artmaktadır. Bunun anlamı da, entropinin verideki yapıdan bağımsız olarak küme sayısı ile artacağıdır. Bu durumda, entropi ile optimum olarak belirlenen küme sayısı, trendde bir süreksizliğin gözlendiği değerdir (Windham, 1981).

$P \in F_c(X)$, $2 \le i \le c$ olmak üzere sınıflandırma entropisi V_{CE}'nin özellikleri aşağıdaki gibi sayılabilir:

(i) $0 \le V_{CE} \le \log c$;

(ii) $V_{CE} = \log c \Leftrightarrow u_{ij} = \frac{1}{c}, \quad \forall i, j;$

(iii) $V_{CE} = O \Leftrightarrow P$, klasik bölünmedir.

3.4. Oran Üssü

Oran üssü (proportion exponent), küme sayısına veya diğer algoritma parametrelerine duyarlı olmayan bir küme geçerliliği fonksiyonudur (Windham, 1981) ve $m_j = x_j$'nin, bölünmenin bulanık kümesine maksimum üyeliği kullanılarak tanımlanır. m_j ne kadar büyükse, x_j noktası o kadar iyi sınıflandırılmıştır. m_j, kalitenin göstergesi yerine daha çok, veri için üyeliklerin rasgele seçilmesi ile daha iyi yapılma olasılığının göstergesi olarak kullanılır.

$P = \{u_1, u_2, ..., u_c\}$ bulanık bölünmesi ile ilgili $c \times n$'lik bir bölünme matrisi ise, her $j = 1,2,...,n$ için

$$m_j = \max_i u_{ij}$$

tanımlanabilir. Eğer, $m_j = u_{ij}$ ise, i, x_j noktasının en iyi tanımlandığı kümeyi gösterir. O zaman, x_j, i-nci *klasik* kümeye atanabilir.

$I_j = \left[\frac{1}{m_j}\right]$, $\frac{1}{m_j}$'de en büyük tamsayı olmak üzere, P'nin oran üssü, aşağıdaki gibi tanımlanan V_{PE} sayısıdır:

$$V_{PE} = -\log_2 \left[\prod_{j=1}^{n} \left(\sum_{k=1}^{I_j} (-1)^{k+1} \binom{n}{k} (1 - km_j)^{n-1} \right) \right] \qquad (3.5)$$

V_{PE}'nin, yalnızca her veri noktasının maksimum üyeliklerini içerdiği varsayılsın. Bu değerler ne kadar büyükse, kümeler ile veri noktalarının belirlenmesi o kadar açıktır. Böylece, m_j'ler küme geçerliliğinin bir ölçüsü olarak tanımlanabilir. Ancak, m_j'lerin değerleri, küme sayısına bağlıdır. Bu zorluğu aşmak için, PE'nin hesaplanmasında sonraki adım uygulanır.

P matrisinin her sütunu c-boyutlu üyelik vektörü olarak ele alınırsa, tüm c-boyutlu üyelik vektörlerinin kümesi

$$MF_c = \left\{ u \in R^n \mid u_i \geq 0, \sum_{i=1}^{c} u_i = 1 \right\}, \tag{3.6}$$

olur. Bu küme, $(c-1)$-boyutlu bir hiperdüzlem ile R^c'nin ilk ortantının kesişimidir. MF_c'nin alanı (3.7)'de verildiği gibidir:

$$\text{Alan}(MF_c) = \frac{\sqrt{c}}{(c-1)!}. \tag{3.7}$$

Eğer, U matrisi (veya P bölünmesi), bulanık olmayan kümeleri gösteriyorsa, her sütunun oranı küçük olacaktır. Eğer c çok büyükse, benzer olarak $f(U)$ da çok küçük olacaktır. Çünkü, son hesaplama adımı negatif algoritma alınarak yapılmaktadır. Bu yüzden, oran üssü

$$V_{PE}(P) \equiv V_{PE}(U) = -\log_2 f(P), \tag{3.8}$$

olur. Bu son hesaplama, V_{PE}'nin değerlerini, özellikle sıfıra yakın oranlar için, daha büyük bir aralığa yaymaktadır. Ayrıca, V_{PE}'nin büyük

değerlerinin, P'nin iyi bir küme yapısı gösterdiğini de anlatmaktadır. Bu yüzden, V_{PE} bir bulanık bölünmenin kalite ölçüsüdür.

Eğer P, bir x_j noktası için

$$u_{ij} = 1, \quad u_{kj} = 0, \quad k \neq i$$

ise, maksimumları biri aşan vektörlerin oranı sıfır olduğu için, V_{PE} tanımsızdır. Bulanık kümeleme algoritmaları nadiren böyle bir durum yaratırlar, yani V_{PE} fonksiyonunun kullanımı için önemli bir engel değildir.

V_{PE}, klasik bölünmeler için tanımlanmayan tek geçerlilik fonksiyonudur. V_{PE}, diğer geçerlilik fonksiyonları gibi, kümeleme algoritmalarında, diğer parametreler arasında seçim yapmak için kullanılabilir. Örneğin, farklı başlangıç bölünmelerin göreli değerlerini değerlendirmek için geçerlilik fonksiyonları kullanılabilir. Aynı zamanda, farklı kümeleme algoritmaları kullanılarak elde edilen bulanık bölünmeleri karşılaştırmak için de kullanılabilir.

Oran üssü geçerlilik ölçüsü değerlerinin, c'nin ne değer aldığına bakılmaksızın küme yapısı olmayan bir veri setinin tüm bulanık bölünmeleri için aynı olması beklenir. Gerçekte, oran üssü,

$$u_i(x_j) = \frac{1}{c}, \quad 2 \leq c < i$$

gibi uniform dağılan üyelikler için , c'ye duyarlı değildir. Küme yapısı olmayan bir veri setinde, üyelikler değil, noktalar rasgele veya uniform dağılır. Bu durumda, oran üssü yalnızca küme sayısına bağlı olur. Bu nedenle, etkili bir geçerlilik fonksiyonu sağlamada başarısızdır.

3.5. Uniform Veri Fonksiyoneli

Oran üssü (proportion exponent), etkili bir geçerlilik fonksiyoneli sağlamak için başarısız olmasına rağmen, yöntemin uygun bir modifikasyonu, en azından teoride, FCM algoritmasında kullanılan parametrelere duyarlı olmayan bir fonksiyonel bulabilir. Bir fonksiyonelin parametrelere duyarlı olmaması şöyle açıklanabilir: Eğer FCM algoritması, veri yapısı olmayan bir kümeye uygulanır ve daha sonra sonuç için bir geçerlilik fonksiyonelinin değeri hesaplanırsa, bu değer, c ve m 'nin hangi değeri kullanılırsa kullanılsın, aynıdır. Eğer bu fonksiyonel, sonra herhangi başka bir veri setine uygulanırsa, bu sabit değerden elde edilen herhangi bir sapma, verideki küme yapısı ihtiyacından kaynaklanabilir. Hatta, sapmanın miktarı, kümelemenin kalitesini gösterir. Böyle bir fonksiyonel oluşturmaya girişmeden önce, test edilecek bir veri setinin olup olmadığını sormak gerekir. Başka bir deyişle, küme yapısı olmayan bir veri seti olup olmadığını sormak gerekir. Bu veri setinin belirlenmesi, uniform veri fonksiyoneli (uniform data functional- *UDF*) oluşturulmasının ilk adımıdır.

Bir uniform veri fonksiyoneli, bir bulanık bölünmenin kalite ölçüsüdür. *UDF* ler, özellikle küme sayısı gibi algoritma parametrelerine duyarlı olmayan kalite ölçüsü elde etmek için Windham tarafından öne sürülmüştür ve yalnızca kümelemenin kalitesindeki farka yanıt verir (Windham, 1982).

Bir *UDF*, etkinliğin tam anlamıyla küme yapısıyla tanımlanmasını ölçer. Uniform dağılmış noktaların bir kümesi Y olsun. Bir u kalite fonksiyonu eğer aşağıdaki koşulları sağlıyorsa bir *UDF* 'dir.

(i) Y'nin herhangi bir bulanık bölünmesi için u'nun değeri v_0 gibi bir sabittir.

(ii) Uniform olmayan herhangi diğer bir veri seti için, u'nun v_0'dan herhangi bir sapması, verideki küme yapısına bağlı ihtiyaçtan olabilir.

(iii) u'nun değerleri küme sayısına veya diğer algoritma parametrelerine duyarlı değildir.

Uniform veri fonksiyoneli, P bulanık bölünmesini veri setinin küme yapısı olmayan benzer bir bulanık bölünmesi ile algoritma parametrelerine duyarlı olmayan bir şekilde karşılaştırmaya dayanır. UDF'nin değerindeki değişimler, bulanık kümelemenin kalitesindeki değişimlere bağlıdır.

Küme altyapısı olmayan bir veri seti, uniform dağılımlı noktaların tam olarak bir kümesini içerir. Hatta, FCM algoritması yalnızca küre şeklinde kümeleri belirleme eğiliminde olduğundan, veri seti küresel olmalıdır. O zaman, seçilecek doğal küme, $Y = B^d = \{x \in R^d : |x| \leq 1\}$ unit ball'dur. Tabii ki, bu set sonsuzdur ve FCM algoritması uygulanamaz. B^d'yi bırakmak yerine, FCM'in sonsuz setlerde uygulanabilir bir genişlemesi verilebilir:

v_i, D_i bulanık sınıfının prototipi, yani merkezi olsun. $v = \{v_1,...,v_c\}$, Q'nun temsilidir. $y \in Y$ noktası ve v_i arasındaki benzemezlik,

$$d_i^2(y,v_i) = D_i^2(y)\|y - v_i\|^2 \tag{3.9}$$

olur. D_i ve v_i arasındaki $I(D_i,v_i)$ yetersizliği ise;

$$I(D_i, v_i) = \int_Y d_i^2(y, v_i)\, dy = \int_Y D_i^2(y) \|y - v_i\|^2\, dy \tag{3.10}$$

olarak yazılabilir. O zaman, Q ile v arasındaki $J(Q,v)$ yetersizliği;

$$J(Q,v) = \sum_{i=1}^{c} I(D_i, v_i) \tag{3.11}$$

şeklinde elde edilir. Yani, M tanımlı, pozitif, simetrik $s \times s$ matris olmak üzere, amaç fonksiyonu;

$$J(Q,v) = \sum_{i=1}^{c} \int_Y D_i^2(y) \|y - v_i\|^2\, dy = \int_Y \sum_{i=1}^{c} D_i^2(y)(y - v_i)^T M (y - v_i)\, dy \tag{3.12}$$

olarak yazılabilir.

$$\frac{\partial J}{\partial v_i} = 0, \quad i = 1,...,c$$

koşullarından; amaç fonksiyonunun minimuma ulaşması için gerekli koşullar aşağıdaki gibi elde edilir:

$$v_i = \frac{\int_Y D_i^2(y) y\, dy}{\int_Y D_i^2(y)\, dy}, \quad i = 1,...,c \tag{3.13}$$

elde edilir. O zaman, v_i'nin k bileşeni,

$$v_i^k = \frac{\int_Y D_i^2(y) y_k \, dy}{\int_Y D_i^2(y) \, dy}, \quad k = 1,...,s \tag{3.14}$$

olur. Lagrange çarpanları yöntemi kullanılarak, $D_i(y)$ için aşağıdaki ifade elde edilir:

$$D_i(y) = \frac{1}{\sum_{k=1}^{c} \frac{\|y - v_i\|^2}{\|y - v_k\|^2}}, \; i = 1,...,c. \tag{3.15}$$

v_i ve $D_i(y)$ için elde edilen formüller iteratif yöntemler kullanılarak, en azından teoride, sonsuz kümenin bir bulanık c-bölünmesi hesaplanabilir. Bu yönteme "Genişletilmiş FCM Algoritması" adı verilir (Windham, 1982).

Eğer böyle bir kümeleme, c, m ve d'nin herhangi bir seçimi için elde edilebiliyorsa, o zaman bu parametrelere duyarlı olmayan bir geçerlilik fonksiyoneli oluşturulabilir. İlk önce, sınıflandırılmanın kalitesindeki artış ile azalan bir kalite ölçüsü seçilir. Örneğin, $\sum_i (u_i(x))^2$ yerine $1 - \sum_i (u_i(x))^2$ yazılabilir. $\rho(x) = x$ noktasının sınıflandırma kalitesinin ölçüsü olmak üzere $\rho : B^d \to R$ tanımlanır. Daha sonra, $S_r \subset B^d$ olmak üzere, ρ'nun aralığındaki her r için $S(r) = \{x \in B^d \mid \rho(x) \le r\}$ ve son olarak $\phi : R \to [0,1]$ ile

$$\phi = \frac{\text{volume of } S(r)}{\text{volume of } B^d} \tag{3.16}$$

tanımlanır. ϕ fonksiyonu, $\phi(\rho(x))$, x'in kendisinden daha iyi sınıflandırıldığı B^d veri setinde oran olmak üzere, bir noktanın sınıflandırma derecesinin ölçüsünü sağlar. Bu fonksiyonun aynı zamanda,

$$\frac{\int_{B^d} \phi(\rho(x))dx}{\text{volume of } B^d} = \frac{1}{2} \tag{3.17}$$

özelliği de vardır. Yani, elde edilen kümelemeden veya kullanılan ρ fonksiyonundan bağımsız olarak B^d üzerinden $\phi.\rho$'nun ortalama değeri ½'dir.

Sonuçta, ρ, c, m ve d'nin her seçimi için ϕ fonksiyonlar ailesi elde edilmiş olur. Bu fonksiyonlar kullanılarak, *UDF* şöyle tanımlanabilir. Eğer, $X = \{x_1, x_2, ..., x_n\}$ d-boyutlu vektörler seti ve U, c ve m parametreleri kullanılıp FCM algoritması uygulanarak elde edilmiş $c \times n$'lik üyelikler matrisi ise, o zaman, ρ sınıflandırma ölçüsü olmak üzere,

$$V_{UDF} = \frac{1}{n} \sum_{j=1}^{n} \phi(\rho(x_j)), \tag{3.18}$$

oluşturularak, söz konusu parametrelere duyarlı olmayan ve yalnızca algoritma ile üretilen kümelemenin kalitesindeki farka yanıt veren bir ölçü elde edilmiş olur. Bu, ρ üzerinden c, m ve d'nin aynı değerleri için elde edilen amaç fonksiyonudur. Bu sadece, $\phi.\rho$ veri seti üzerinden ortalamadır ve sonuçta eğer, V_{UDF}'nin değeri ½'ye yakınsa, FCM'in veri setinde bir küme yapısı bulmadığı söylenebilir. Ayrıca,

UDF, noktaların eğer unit ball'dan seçilmiş olsalar, daha iyi sınıflandırılacakları olasılıklarının ortalaması olarak yorumlanabilir. Sonuç olarak, V_{UDF} değeri sıfıra ne kadar yakınsa, kümeleme o kadar iyidir.

3.6. Ayrılma Endeksi

Dunn (1974), klasik bölünme için bir ayrılma endeksi (separation index) tanımlamış ve Dumitrescu, Lazzerini ve Jain (2000), iki bulanık küme arasındaki uzaklık tanımını kullanarak Dunn'ın yaklaşımını bulanık bölünmeler için genelleştirmişlerdir.

SI ayrılma endeksi küme sayısına ya da diğer algoritmik parametrelere duyarlı olmayan bir geçerlilik fonksiyonelidir ve küresel veya elipsoid şekil gösteren bulanık bölünmeleri karşılaştırmak için kullanılır (Dumitrescu et al., 2000). $\delta(u_i)$, u_i bulanık sınıfının çapı olmak üzere,

$$\delta(u_i) = \max_{j,k}[\min(u_{ij}, u_{ik}) d(x_j, x_k)], \tag{3.19}$$

bir P bulanık bölünmesinin V_{SI} ayrılma endeksi,

$$V_{SI} = \frac{\max_i \min_{k \neq i} d(u_i, u_k)}{\min_i \delta(u_i)} \tag{3.20}$$

olarak tanımlanır. u_i ve u_k, X sonlu kümesinde tanımlı bulanık kümeler olduğundan , aralarındaki uzaklık

$$d(u_i,u_k)=\int_0^1 D(u_i^t,u_k^t)dt \tag{3.21}$$

formülüne dayanarak hesaplanabilir. Burada $D(u_i^t,u_k^t)$, t seviye kümeleri olmak üzere, klasik u_i^t ve u_k^t kümeleri arasındaki uzaklıktır.

$V_{SI}>1$ olduğunda, kompakt ve ayrılmış sınıflar bulunmuş olur.

3.7. Fukuyamo-Sugeno Küme Geçerliliği Ölçüsü

Fukuyamo-Sugeno küme geçerliliği ölçüsü, $m_X=M(X)$, X veri setinin ortalama vektörü olmak üzere, amaç fonksiyonu ile v_i, $i=1,...,c$ küme merkezlerinin ortalama vektörden sapmalarını birleştirir (Fukuyamo ve Sugeno, 1989).

Fukuyamo-Sugeno geçerlilik fonksiyonu aşağıdaki gibi tanımlanabilir:

$$\begin{aligned} V_{FS_m}(P,v) &= \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m [d^2(x_j,v_i)-d^2(m_X,v_i)] \\ &= J_m(P,v)-\sum_{i=1}^{c} d^2(m_X,v_i)\sum_{j=1}^{n} u_{ij}^m \end{aligned} \tag{3.22}$$

Eğer, ikinci ifade

$$K_m(P,v)=\sum_{i=1}^{c} d^2(m_X,v_i)\sum_{j=1}^{n} u_{ij}^m \tag{3.23}$$

olarak gösterilirse,

$$V_{FS_m}(P,v) = J_m(P,v) - K_m(P,v) \tag{3.24}$$

yazılabilir. $K_m(P,v)$, u_i'ye olan üyelikleri, u_i'nin v_i prototipleri ve verinin ortalama vektörü arasındaki uzaklık ile birleştirir. $K_m(P,v)$, v temsilinin geometrik kompaktlığı olarak düşünülebilir.

3.8. Xie-Beni Küme Geçerliliği Ölçüsü

Xie ve Beni (1991), $J_m(P,v)$ yetersizlik ölçüsü ile merkezler arası uzaklıkları kullanan bir V_{XB} küme geçerliliği endeksi önermişlerdir. Yani, bu küme geçerliliği ölçüsünde bulanık veri kümelemesinin kalitesini ifade etmek için yalnızca üyelikler kullanılmaz. Bu geçerlilik fonksiyoneli, bir bulanık c-bölünmesi üzerinden veri setinin ortalama kompaktlığını ve ayrılmasını ölçer.

π ile gösterilen oran, tüm veri setinin büyüklüğünün toplam varyasyonunu ölçer. Yani,

$$\sigma(P,v) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2 d^2(x_j, v_i) \tag{3.25}$$

olmak üzere, $\pi = (\sigma / n)$, bir veri setinin bulanık c-bölünmesinin kompaktlığı olarak adlandırılır.

π değeri, her sınıfın ne kadar kompakt olduğunu ölçer. Sınıflar ne kadar kompakt ise, π o kadar küçük olur. π, veri setinin karakteristikler dağılımının ve daha da önemlisi, verilerin kümelere nasıl dağıtıldığının bir fonksiyonudur. Ancak, veri noktalarının sayısından bağımsızdır. Bir

veri seti için, küçük bir π değeri, daha kompakt kümeler ile bir bölünmeye ulaşıldığının, yani daha iyi bir bölünmenin göstergesidir.

Benzer olarak, $\pi_i = (\sigma_i / n_i)$ miktarına da i sınıfının kompaktlığı adı verilir. n_i, i sınıfındaki vektörlerin sayısı olduğundan, (σ_i / n_i), i sınıfındaki ortalama varyasyonu gösterir.

Bir bulanık c-bölünmesinin ayrılması ise aşağıdaki gibi tanımlanır:

$$sep(v) = \min_{i \neq k} d^2(v_i, v_k) \tag{3.26}$$

$sep(v)$'nin büyük değeri, tüm kümelerin iyi ayrılmış olduğunu gösterir.

O zaman, kompaktlık ve ayrılma geçerlilik fonksiyoneli V_{XB}, kompaktlık ölçüsü π'nin ayrılma ölçüsü $sep(v)$'ye oranı olarak tanımlanabilir. Yani, $v_i \in R^s$, u_i bulanık sınıfının prototipi olmak üzere, P ve $v = \{v_1, ..., v_c\}$ ile tanımlanan kümelemenin V_{XB} ölçüsü aşağıdaki gibi yazılabilir:

$$V_{XB} = \frac{1}{n} \frac{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 d^2(x_j, v_i)}{\min_{i \neq k} d^2(v_i, v_k)} \tag{3.27}$$

En iyi bulanık bölünme, tüm sınıfların kompakt ve ayrılmış olduğunun göstergesi olan V_{XB} geçerlilik fonksiyonelini minimum yapan değer olarak elde edilir.

V_{XB}'nin tanımı, üyelik derecelerini bulmak için kullanılan algoritmadan bağımsızdır. Yani, FCM algoritması için $m = 2$ olduğunda, V_{XB} aşağıdaki gibi yazılabilmektedir:

$$V_{XB} = \frac{1}{n} \frac{J(P,v)}{\min_{i \neq k} d^2(v_i, v_k)} \tag{3.28}$$

V_{XB}'nin minimize edilmesi, FCM algoritmasının amacı olan, $J(P,v)$'nin minimize edilmesi ile ilgilidir.

4. BAŞLANGIÇ KÜMELERİN OLUŞTURULMASI

Kümeleme algoritmalarında diğer önemli adımlardan biri, başlangıç kümelerin oluşturulmasıdır. Buna bağlı olarak kümeleme algoritmasının sonuca ulaşma hızı değişebilir ve aynı zamanda oluşturulmuş sonuç kümelerin şekilleri farklı olabilir. Bundan dolayı, kümeleme algoritmasında kullanılan başlangıç kümeleri oluşturma yöntemleri büyük önem taşır. Tez içinde sunulan program sisteminde diğer çalışmalarda daha sık kullanılan başlangıç kümelerin oluşturulması yöntemleri yer almaktadır.

4.1. Mountain Yöntemi

Yager ve Filev (1994(a, b)), küme merkezlerinin tahmin edilmesi için Mountain yöntemi olarak adlandırdıkları bir algoritma önermişlerdir. Bu algoritma, FCM algoritmasında kullanılacak küme merkezlerinin başlangıç tahminlerini elde etmek için bir araç olmasının yanında, küme merkezlerinin yalnızca yaklaşık değerlerinin istendiği durumlarda, başlı başına bir kümeleme algoritması olarak da kullanılabilmektedir. Mountain yönteminin özü, insanoğlunun görsel olarak küme oluşturmak için yaptıklarına dayanmaktadır (Jang et al., 1997).

Mountain yönteminde ilk adım, R^s'de bir düğüm (grid) oluşturarak, R^s nesneler uzayını kesiklendirmektir. Düğüm noktalarında oluşan düğüm doğruların kesişimi, istenilen kesiklendirmeyi sağlar. Böyle noktaları içeren R^s'nin sonlu alt kümesi $G_{i_1,\ldots,i_s}$, aday küme merkezlerini oluşturur. Yani, sonuç merkezlerin yaklaşıklık derecesinin, düğümlemenin kalitesine duyarlı olduğu görülür. Düğümleme ne kadar

iyi olursa, yaklaşıklık azalır; ancak hesaplamalar artar. Düğümleme R^s'de uniform olmak zorunda değildir ve uzayın farklı bölümlerinde düğümlemenin farklı yoğunlukları olabilir.

İkinci adım, verilerin girilmesi ve V'de tanımlı mountain fonksiyonunun (M) oluşturulmasıdır. M'nin oluşturulması için, her x_k verisi için her G noktasındaki M'ye bir "miktar" eklenir. Bu eklenen miktar, x_k'nın G'den uzaklığına bağlıdır. Yani nokta merkeze ne kadar yakınsa, eklenecek miktar artmaktadır. Böylece, tüm noktalar ele alındıktan sonra, R^s'de verilerin dağılımını yansıtan sıradağa benzer bir fonksiyon elde edilmiş olur.

x özellik değerlerinin olasılık yoğunluk fonksiyonunun $G_{i_1,\ldots,i_s}$ düğüm noktalar dizisinde i_j indisleri ile düzgün, dikdörtgensel düğüm noktalar; d, uzaklık ölçüsü; s veri setinin boyutu, n veri sayısı ve α sezgisel olarak belirlenen parametre olmak üzere, Mountain fonksiyonu aşağıdaki gibidir:

$$M(G_{i_1,\ldots,i_s}) = \sum_{k=1}^{n} \exp[-\alpha\, d(x_k, G_{i_1,\ldots,i_s})]. \qquad (4.1)$$

Sonraki adım, tepelerin yıkılarak, küme merkezlerinin seçilmesidir. İlk küme merkezi, $G_{i_1,\ldots,i_s}$'de, bu sıradağın zirvesi olan ve maksimum M değerinin olduğu G_1 düğüm noktasında bulunur. $G_{i_1,\ldots,i_s}$'deki G_1'den olan uzaklıklarına bağlı olarak $M(G_1)$ değerinden bir "miktar" çıkarılır. Bunun amacı, tepeleri küçültmektir. Daha sonra yeni zirve aranır. Bu yeni zirve, ikinci küme merkezi, G_2, olur. G_2 ve değeri yine

mountain fonksiyonunu düşürür. Mountain fonksiyonu gözle görülür şekilde bozulana kadar bu işleme devam edilir.

$$M^2(G_{i_1,\ldots,i_s}) = M^1(G_{i_1,\ldots,i_s}) - M_1^* \exp[-\beta d(G_{j_1,\ldots,j_s}, G_{i_1,\ldots,i_s})]. \qquad (4.2)$$

Burada, β sezgisel olarak belirlenen parametre M^2 ilk küme elendikten sonra elde edilen Mountain fonksiyonu, M^1 orijinal Mountain fonksiyonu ve M_1^* , M mountain fonksiyonunun $j_1,\ldots, j_s$ düğüm noktasında ulaştığı maksimum değeridir. Sonra yeniden M^2 fonksiyonuna en büyük değer veren düğüm noktası bulunur ve bu nokta ikinci küme merkezi olarak ele alınır. Bu döngü $\frac{M_k^*}{M_1^*} < \delta$ olana kadar devam eder. Burada k, sonuncu bulunan kümenin numarası ve δ, herhangi belirli küçük bir sayıdır.

Mountain yönteminin algoritması aşağıdaki gibi verilebilir:

Mountain Yöntemi Algoritması

ADIM 1 : α , β ve δ parametrelerinin değeri ve G düğümü belirlenir. k=1 alınır.

ADIM 2 : (4.1) eşitliği ile $M^1 = M$ Mountain fonksiyonu hesaplanır.

ADIM 3 : M fonksiyonunun değerinin maksimum olduğu G düğüm noktası seçilir.

ADIM 4 : (4.2) eşitliğine dayanarak yeni M^k fonksiyonu hesaplanır.

ADIM 5 : $\frac{M_k^*}{M_1^*} < \delta$ ise DUR! Aksi halde k=k+1 alınarak Adım 3'e dönülür.

Düğümlemenin yapılması ile sürekli optimizasyon probleminin sonlu bir probleme yaklaştırılması, mountain fonksiyonunu bozulması ile yeni merkezler aranırken, bulunmuş merkezlerin etkisinin yok edilmesi sağlanır. Mountain yöntemi algoritmasının performansı, α ve β parametrelerinin seçimi, düğümlemenin tanelendirilmesine (granularity) ve δ 'nın seçimine bağlıdır.

4.2. Geliştirilmiş Mountain Yöntemi

Yager ve Filev'in (1994(a,b)) önerip, Chiu'nun (1994) sadeleştirdiği Mountain yöntemini büyük veya çok boyutlu veri setlerine uygulamak zor olduğundan, düğüm noktaların sayısını indirgemek için Velthuizen et al. (1997) Geliştirilmiş Mountain yöntemini (Modified Mountain Method) önermişlerdir. Geliştirilmiş Mountain yönteminde sözü geçen indirgeme, çok-boyutlu histogramdan aday küme merkezlerinin seçilmesi ile sağlanır. Histogramın hesaplanması çok hızlı yapılabilir. Yani, histogram için düğüm noktaların sayısı, hesaplama zamanı ile sınırlı değildir. Aday küme merkezleri, Khotanzad ve Bouarfa'nın (1990) önerdiği graf teorisi bazında yaklaşıma dayanan bir yöntem kullanılarak belirlenir. Histogramdaki her hücre için, çevresindeki komşu hücreler içinde eleman sayısı en çok olana bir ok yönlendirilir. Daha sonra aday küme merkezleri, belli bir eşiğin üzerinde ok sahibi olan hücreler içinde aranır.

Mountain yönteminde (4.1) eşitliği ile verilen α çekirdek genişliği (kernel width), Geliştirilmiş Mountain yönteminde aşağıdaki gibi hesaplanır:

$$\alpha = \frac{1}{[\sqrt{trace(\Sigma)}/n]r(N/c)} \tag{4.3}$$

Burada, Σ, verilerin dağılışının kovaryans matrisi, N, örneklem sayısı, s, verilerin boyutu ve $\Gamma(.)$, Gamma fonksiyonu olmak üzere, r aşağıdaki gibi hesaplanır:

$$r = \left\lceil \frac{2^{s+2}\Gamma[(s+2)/2]}{(s+2)^{(s/2)+1}} \right\rceil \tag{4.4}$$

Orijinal Mountain yönteminde, küme merkezleri, mountain fonksiyonundaki en yüksek tepe olarak belirleniyordu ve kümenin etkisi, (4.2) eşitliği ile eleniyordu. Üst üste gelen dağılımlı kümeler için, kümelerin başarılı bir şekilde elenmesi çok önemlidir. Orijinal Mountain yöntemi, sabit bir çekirdek kullandığı için, eleme parametresi β'nın seçimine karşı da duyarlıdır. Yani, mountain fonksiyonunda yerel komşuluklara bağlı olarak kümenin dağılımının gerçek şeklinin tahmini kullanılır. Bu yaklaşım, sabit bir çekirdek yerine, dağılımın gerçek şekli kullanıldığı için, yöntemin β parametresine olan duyarlılığını yok eder.

Mountain fonksiyonunun gerçek en yüksek tepe değerine sahip aday küme merkezi belirlendikten sonra, en yakın aday küme merkezlerine olan uzaklıklar kullanılarak, tepenin yerel komşulukları belirlenir. Daha sonra mountain fonksiyonu, seçilen aday küme merkezi

etrafında düğüm noktaların sınırlı sayısı ile "fine düğüm" üzerinde değerlendirilir.

Yoğunluk fonksiyonlarının şekli için açık bir seçim, çok değişkenli normal dağılımdır. Bu dağılım, birçok domain için optimal seçim olmayabilir, ama genellikle doğru dağılım bilgisinin eksikliğinde uygun bir seçim olacaktır.

Geliştirilmiş Mountain yönteminin algoritması aşağıdaki gibi verilebilir:

Geliştirilmiş Mountain Yöntemi Algoritması

ADIM 1: Histogramdaki tepelerden global G_g düğümü belirlenir.

ADIM 2 : (4.3) eşitliği kullanılarak α hesaplanır.

ADIM 3 : (4.1) eşitliği kullanılarak global M_g Mountain fonksiyonu hesaplanır.

ADIM 4 : M_g fonksiyonunun değerinin maksimum olduğu G düğüm noktası seçilir.

ADIM 5 : Maksimum M_g'lerin komşulukları içinde yerel düğüm G_1 belirlenir.

ADIM 6 : M_1'de vadileri aranır ve G_1'de kümeye ait olan noktalar seçilir.

ADIM 7 : Seçilen noktaların çok değişkenli normal dağılıma uyumu sağlanır.

ADIM 8 : Bulunan normal dağılım M_g'den çıkarılır.

ADIM 9 : c sayıda tüm kümeler bulunana kadar Adım 4 ile devam edilir.

4.3. K-En Yakın Komşuluk Kuralı

Bulanık c-ortalamalar kümeleme algoritması (FCM), her zaman küme merkezlerinin başlangıç tahmininden başlayarak J_m'nin kümeleme endeksinin bir yerel optimasına yakınsar. Bu küme merkezlerinin başlangıç yerlerinin nasıl seçildiği, sonuca varılabilmesi açısından önemli değildir. Fakat, küme merkezlerinin farklı başlangıç seçimleri, farklı yerel optimalara götürür. K-En Yakın Komşuluk (K-Nearest-Neighbours) (K-NN) karar kuralı, başlangıç küme merkezlerinin bir denetlenmemiş izlemesini yapmaya dayanır (Zahid et al., 2001). Bu kural, model tanıma problemleri için kullanışlı bir parametrik olmayan yöntem sağlar. Bu, kümelenmeyen bir y nesnesini, K, y'nin en yakın nesnelerinin sayısı olmak üzere, K-en yakın komşuları arasında en iyi temsil edilen kümeye atar.

Klasik K-en yakın komşuluk algoritması aşağıdaki gibidir:

Klasik *K*-En Yakın Komşuluk Algoritması

ADIM 1:

- Etiketlenmemiş veri seti $X = \{x_1,...,x_n\}$ verilsin.
- Herhangi bir kümelenmemiş y belirlenir.
- y 'nin en yakın komşularının sayısı K belirlenir.

ADIM 2 :

- i=1 alınır.

Do Until (y 'ye en yakın K-NN bulunana kadar)

- y ile x_i arasındaki $d(y,x_i)$ uzaklığı hesaplanır.

If $(i \leq K)$ **then**

- x_i, K-NN'in E kümesine eklenir.

Else If (x_i,önceki herhangi bir en yakın komşudan y'ye daha yakın ise) **then**

- E kümesindeki en uzak birim silinir, x_i eklenir.

EndIf

$i = i + 1$

End Do Until

ADIM 3 :

- E kümesinde temsil edilen çoğunluk küme belirlenir.
- y, çoğunluk kümesine eklenir.

end.

X'in başlangıç küme merkezlerini yerleştirme algoritmasının ilk aşaması aşağıdaki gibidir:

$X = \{x_1,...,x_n\}$, n tane etiketlenmemiş nesnenin sonlu kümesi ve c, verilen küme sayısı olsun. İlk aşama, tüm uzay X'in c hücreye bölünmesini içerir. Aynı hücrede nesneler Euclid anlamda birbirine benzerdir. Bu gruplama K-NN algoritmasının ilk iki adımı kullanılarak bulunur. Başka bir deyişle, her hücrenin aranması şu şekilde yapılır: her $1 \le i \le c$ için, her hücre $E_i : (y_i, y_i$'nin K-en yakın komşuları, $G_i)$, olarak formüle edilen bir K-en yakın komşuluk kümesi ile gösterilir. Bunun anlamı, her y_i birimi için K-en yakın komşuluğu ile ilgili bir E_i kümesinin bulunacağıdır. E_i kümesinin G_i ortalama merkezi (4.5)'te verildiği gibi tanımlanır:

$$G_i = \frac{\sum_{x_k \in E_i} x_k}{K+1}. \tag{4.5}$$

y_i aşağıdaki gibi belirlenir:

$i = 1$ ise, y_1 aşağıdaki koşulları sağlar:

- $y_1 \in E_1$
- y_1, tüm $x_k \in X$ bireylerinin G_0 global ortalamasından en uzak olan bireydir.

$$G_0 = \frac{\sum_{x_k \in X} x_k}{n}. \tag{4.6}$$

$2 \le i \le c$ ise, E_i'yi oluşturan y_i bireyi aşağıdaki koşulları sağlar:

- $y_i \in E_i$,
- $y_i \in E_\alpha$, $1 \le \alpha \le i-1$,
- y_i , y_α 'nın K-en yakın komşuluğunda değildir,
- y_i, G_{i-1} 'den en uzak olan bireydir.

Pratikte y_i'nin K-en yakın komşularının sayısı, 1 ve (n-1) arasında değişebilir. Öte yandan, E_i , $(1 \le i \le c)$, kümelerinin sayısı c, K'ya bağlıdır. $K \to (n-1) \Rightarrow c \to 1$ olur ve eğer $K \to 1 \Rightarrow c \to n/2$ olur. Böylece, K, c'nin bir fonksiyonu gibi tanımlanır:

$$K = Int\left(\frac{n}{c} - 1\right). \tag{4.7}$$

Küme sayısı c, kullanıcı tarafından veya küme geçerliliği yöntemlerinden biri kullanılarak tahmin edilebilir.

Bazı özel durumlarda, sınıflandırılmamış x bireyleri $(x \notin (E_1,...,E_c))$ en yakın merkeze $(G_1, G_2,..., G_c)$ atanır ve sonra tüm bu merkezler güncellenir. Yani, bu adımın sonunda tüm bireyler, elde edilen $(E_1, E_2,..., E_c)$ kümelerine atanır. Diğer bir deyişle, bu yöntem ile X veri seti c homojen gruba bölünür, başlangıç küme merkezleri tanımlanır ve en iyi bölünmeye yaklaşılır.

Elde edilen bölünmeyi bulanıklaştırmak için, ikinci aşama FCM algoritmasının bir iterasyonunu kullanılır. Bulanık üyelik derecelerini ve yeni bulanık merkezleri hesaplamak için 2. Bölümde verilen (2.3) ve (2.4) no'lu eşitlikler kullanılır.

5. PROGRAM SİSTEMİNİN ÇALIŞMA PRENSİPLERİ

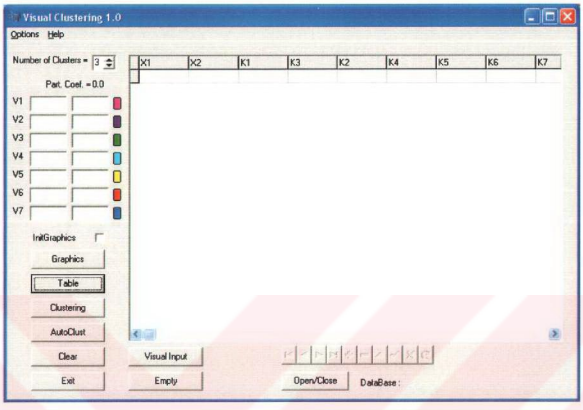
Bu bölümde, 2, 3 ve 4. Bölümlerde incelenen yöntemler ve algoritmalar için oluşturulmuş program sisteminin anlatımı ve çalışma prensipleri ele alınmıştır (Nasibov vd., 2004). Söz konusu program sistemi, Borland C++ Builder 6.0 programlama ortamında yazılmış ve Intel Pentium IV, 1.7 GHz, 256 MB RAM teknik özellikleri olan bilgisayarda uygulanmıştır.

5.1. Program Sisteminin Temel Modulları

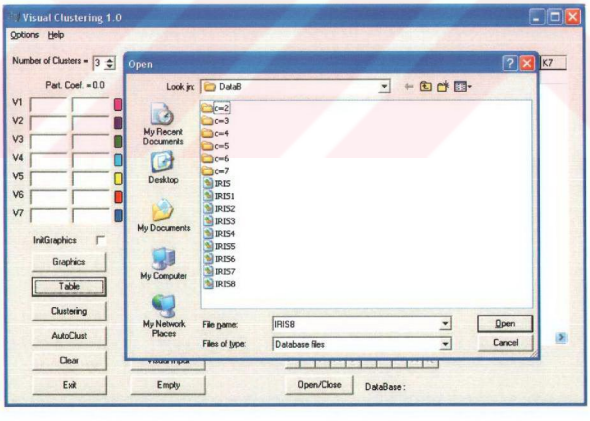
Bu kısımda, program sisteminde oluşturulan formlar, fonksiyonel buttonlar, bilgisel komponentler ve fonksiyonel modullar hakkında bilgi verilecektir.

5.1.1. Formlar

Program çalıştırıldığında ekrana ilk önce Şekil 5.1'deki pencere gelir. Kümelemenin yapılabilmesi için ilk önce çalışılacak tablonun seçilmesi gerekmektedir. Bunun için, "Table" buttonuna basılarak açılan pencereden (Şekil 5.2), üzerinde işlem yapılacak veri seti seçilir.

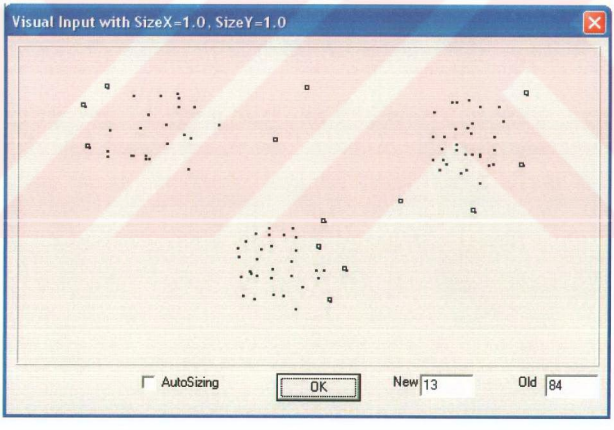


Şekil 5.1 Programın açılış penceresi.



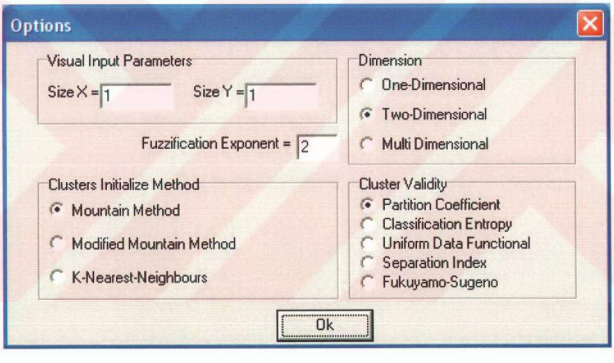
Şekil 5.2. İşlem yapılacak veri setinin seçileceği pencere.

İki boyutlu kümelemede "Visual Input" buttonuna basıldığında işlem yapılacak tabloda bulunan veriler Şekil 5.3'teki gibi gözükür ve yeni verilerin visual olarak girilmesine olanak sağlanır. Pencerenin sağ alt köşesinde, yeni girilen verilerin sayısı ve onun yanında ise mevcut veri sayısı sırasıyla "New" ve "Old" kısmında gösterilmektedir. "AutoSizing" parametresi seçildiğinde, verilerin minimum ve maksimum değerleri dikkate alınarak formun ölçeğinin X ve Y koordinatları boyunca otomatik olarak ayarlanması sağlanır. Veri kümesi değiştirildikten sonra, kümelemenin her defa yeniden yapılması gerekmektedir.



Şekil 5.3. İşlem yapılacak veri setinin visual gösterimi ve yeni verilerin eklenmesi.

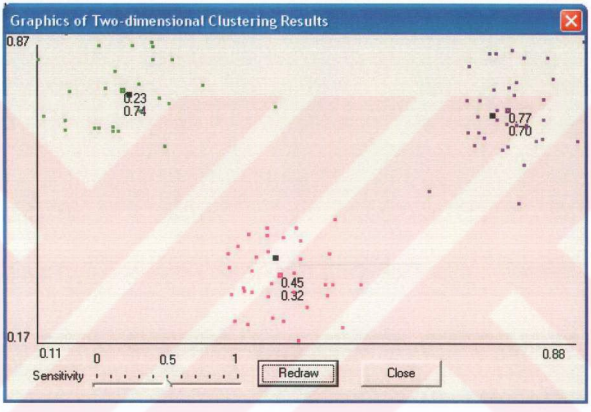
Daha sonra, ana menüde "Options" seçeneği seçilerek, Şekil 5.4'te görünen pencerede "Visual Input Parameters" kısmından girilecek verilerin X ve Y koordinatları boyunca ölçeği, "Dimension" kısmından veri uzayının boyutu, "Cluster Validity" kısmından küme sayısını belirlemek için kullanılan yöntem ve "Cluster Initialize Method" kısmından ise başlangıç küme merkezlerinin belirleneceği yöntem seçilir ve "Fuzzification Exponent" kısmından bulanıklaştırma parametresinin değeri belirlenir.



Şekil 5.4. Seçenek penceresi.

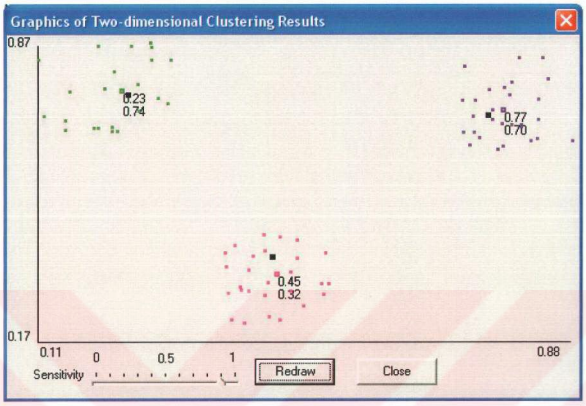
Kümeleme işlemini, küme sayısını geçerlilik ölçülerine bağlı olarak belirleyecek şekilde gerçekleştirmek için açılış penceresindeki "AutoClustering" düğmesine basılması gerekmektedir. Buna göre bulunan kümelerin optimal sayısı, seçilmiş küme geçerliliği yöntemine uygun değer ve küme merkezlerinin koordinatları ekranın sol üst

Başlangıç küme merkezlerini, yeni bulunan küme merkezleri ile birlikte görüntülemek için"InitGraphics" seçeneği seçilir. Böylece, renkli kareler ile verilen sonuç küme merkezlerle beraber, başlangıç küme merkezleri de siyah kareler olarak Şekil 5.8'deki gibi görülebilir.

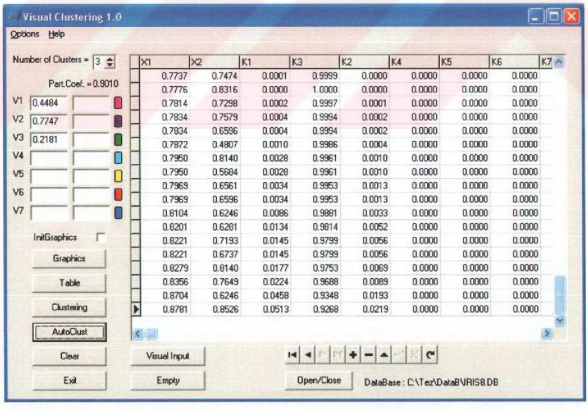


Şekil 5.8. İki boyutlu kümeleme sonuçları, başlangıç ve sonuç küme merkezleri.

Grafik penceresinin sol alt kısmında bulunan "Sensitivity" kısmında ok hareket ettirilerek, elemanların kümelere minimum ait olma dereceleri belirlenir ve "Redraw" buttonuna basıldığında yalnızca belirlenen üyelik derecesinden yüksek derece ile kümelere ait olan elemanlar grafikte görünür (Şekil 5.9).

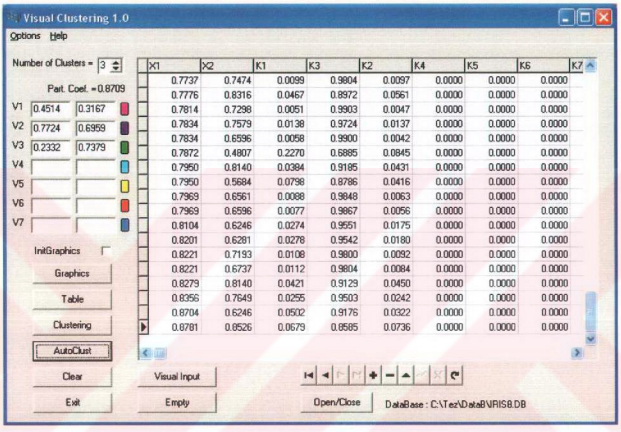


Şekil 5.9. Kümelere belli bir derecenin üzerinde üyeliği olan elemanların grafiği.



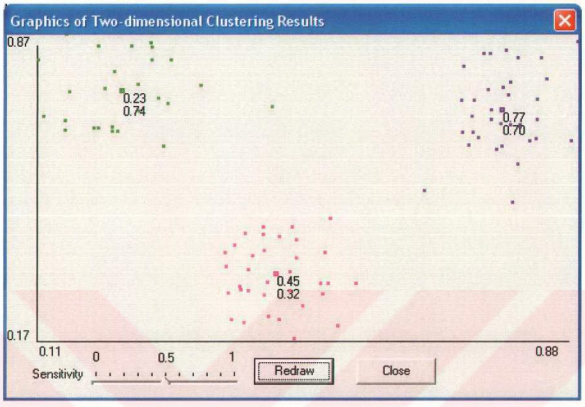
Şekil 5.10. Tek boyutlu kümelemenin uygulanması.

bölümünde görüntülenir (Şekil 5.5). Ekranın sağ alt kısmında ise, çalışılan veri tabanının ismi görülmektedir.

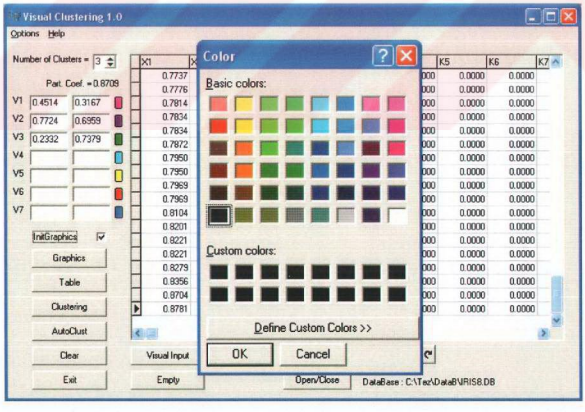


Şekil 5.5. Kümelemenin uygulanması.

Kümelemenin sonuçlarını ekrana visual olarak vermek için "Graphics" düğmesi kullanılır. Şekil 5.6'da iki boyutlu kümelemenin sonuçları ve küme merkezleri görülmektedir. Elemanların farklı kümelere ait olmasını ayırt etmek için her kümenin elemanları farklı renklerde görünür ve kümelerin renkleri kullanıcının isteğine bağlı olarak merkezlerin yanındaki renk kutularının üzerine tıklayarak değiştirilebilir (Şekil 5.7).



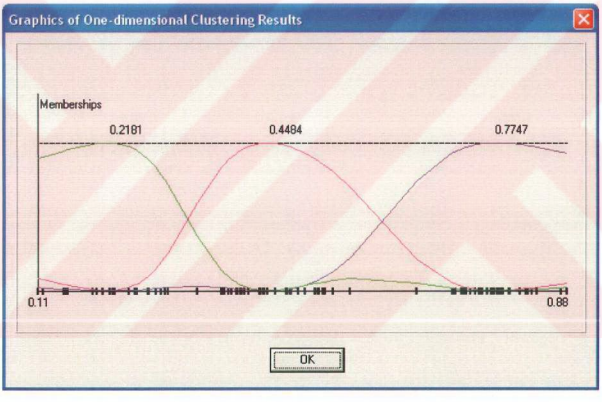
Şekil 5.6. İki boyutlu kümeleme sonuçları ve sonuç küme merkezleri.



Şekil 5.7. Başlangıç küme merkezlerinin görüntülenmesinin seçilmesi.

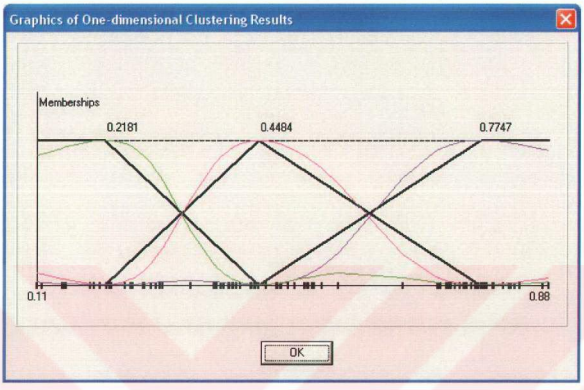
"Options" seçeneğinde "Dimension" kısmında veri uzayının boyutu tek boyutlu olarak seçilip "AutoClustering"ten kümeleme işlemi yapıldığında, optimal küme sayısı, seçilmiş küme geçerliliği yöntemine uygun değer ve küme merkezleri ekranın sol bölümünde Şekil 5.10'daki gibi görüntülenir.

Tek boyutlu kümeleme sonuçları ve küme merkezleri Şekil 5.11'deki gibi görüntülenmektedir.

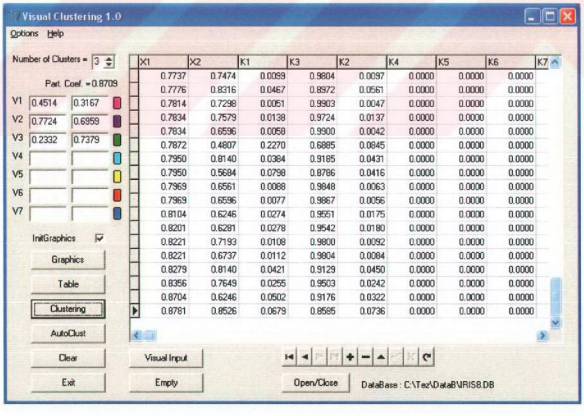


Şekil 5.11. Tek boyutlu kümeleme sonuçları ve küme merkezleri.

Grafiği çizdirmeden önce "InitGraphics" parametresi işaretlediğinde, iki boyutlu kümelemede olduğu gibi, tek boyutluda da başlangıç kümeler (koyu siyah hatlar), sonuç kümelerle (renkli eğriler) beraber görüntülenmektedir (Şekil 5.12).



Şekil 5.12. Tek boyutlu kümeleme sonuçları, küme merkezleri ve başlangıç kümeler.



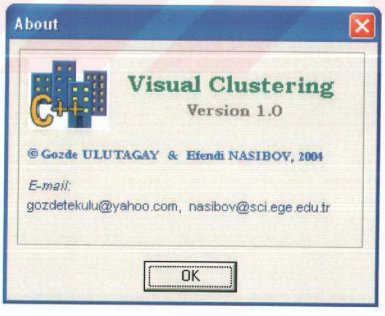
Şekil 5.13. Küme sayısını vererek kümelemenin uygulanması.

Kümeleme işlemini küme sayısını önceden belirleyerek gerçekleştirmek için, açılış penceresindeki "Number of Clusters" parametresine değer vererek "Clustering" düğmesine basılması gerekmektedir. Kümeleme işlemi bittikten sonra, seçilmiş küme geçerliliği yöntemine uygun değer de ekranın sol üst bölümünde görüntülenir (Şekil 5.13).

5.1.2. Menüler

Programın açılış penceresinde "Options" menüsünden "Parameters" seçildiğinde, ekrana önceki bölümde Şekil 5.4'te gösterilen form gelir. Bu formdaki yöntemlerin seçimine ilişkin bilgiler ilgili bölümde anlatılmıştır.

Yine aynı pencerede, "Help" menüsündeki "About" seçeneği seçildiğinde ise, ekrana Şekil 5.14'te verilen pencere gelmektedir.



Şekil 5.14. Program sistemi hakkında bilgiler penceresi.

5.1.3. Fonksiyonel Buttonlar

Programın açılış penceresinde bulunan buttonlar ve kullanım amaçları aşağıda verilmiştir:

Graphics - Kümeleme sonuçlarını ekrana visual olarak vermek için kullanılır. Bu buttonun çalışması Şekil 5.6'da gösterilmiştir.

Table - Kümeleme yapabilmek için çalışılacak veri setinin seçilmesi amacıyla kullanılır. Bu buttona basıldığında ekrana gelen pencere Şekil 5.2'de verilmiştir.

Clustering - Kümeleme işlemini küme sayısını önceden belirleyerek gerçekleştirmek için, açılış penceresindeki "Number of Clusters" parametresine değer verilerek bu buttona basılır.

AutoClust - Kümeleme işlemini "Options" menüsünden seçilmiş olan küme geçerliliği yöntemlerinden biri ile gerçekleştirerek küme sayısını otomatik olarak belirlemek için kullanılır.

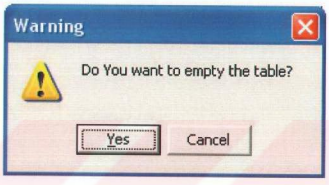
Clear - Kümeleme işlemini yeniden gerçekleştirmek için, kümelere ait eski üyelik derecelerini temizler.

Exit - Programdan çıkışı sağlar.

Visual Input - Bu buttona basıldığında işlem yapılacak tabloda bulunan veriler görüntülenir. Ekrana gelen pencere Şekil 5.3'te ve pencerede yapılacak işlemlerin açıklaması ilgili bölümde verilmiştir.

Open/Close - Üzerinde çalışılan veri seti tablosunu aktif/pasif duruma geçirir.

Empty - Yeni veri girişi yapabilmek için çalışılacak veri setini temizler. Veri setini yanlışlıkla silmemek için ekrana önce Şekil 5.15'te gösterilen uyarıcı pencere çıkarılır ve onay istenir.



Şekil 5.15. Tablo silmek için uyarı penceresi.

5.1.4. Bilgisel Komponentler

Number of Clusters - İşlem yapılacak veri setindeki küme sayısı önceden biliniyorsa, "Clustering" buttonuna basmadan önce bu komponentin penceresine değer verilmesi gerekmektedir. Eğer kümeleme işlemi "AutoClust" ile yapılmışsa, yine bu komponentin penceresinde bulunan kümelerin optimal sayısı görüntülenir.

Part.Coef. - Seçilen küme geçerliliği yöntemine ilişkin (Partition Coefficient, Classification Entropy, Separation Index, UDF veya Fukuyamo-Sugeno) değerini gösterir.

V1...V7- Kümeleme işlemi yapıldıktan sonra optimal küme sayısına göre (1-7 arası) bulunan küme merkezlerinin koordinatlarını gösterir.

InitGraphics - Yeni bulunan kümelerle beraber başlangıç kümeler de görüntülenmek istendiğinde, bu parametrenin işaretlenmesi gerekmektedir. Ayrıntılı bilgi Şekil 5.7, 5.8 ve 5.12'de verilmiştir.

DataBase - Çalışılan veri setinin adını gösterir.

5.1.5. Fonksiyonel Modullar

Bu kesimde, formlardaki fonksiyonlar ve kullanım amaçlarına değinilmiştir.

- *TForm1* formuna bağlı fonksiyonlar:

void _fastcall TForm1::About2Click(TObject *Sender)

Bu fonksiyon, program sistemi hakkında bilgiler içeren formu ekrana çıkarır.

void _fastcall TForm1::Button1Click (TObject *Sender)

Bu fonksiyon, kümeleme işlemini gerçekleştirir.

void _fastcall TForm1::Button2Click (TObject *Sender)

Bu fonksiyon, çalışılan tabloyu aktif/pasif duruma geçirir.

void _fastcall TForm1::Button3Click (TObject *Sender)

Bu fonksiyon yeniden kümeleme yapabilmek için kümelere ait eski üyelik derecelerini temizler.

void _fastcall TForm1::Button4Click (TObject *Sender)

Bu fonksiyon, tek ve iki boyutlu hallerde kümelerin grafiğini çizer.

void _fastcall TForm1::Button5Click (TObject *Sender)

Bu fonksiyon, programdan çıkışı sağlar.

void _fastcall TForm1::Button6Click (TObject *Sender)

Bu fonksiyon, üzerinde çalışılacak veri setinin seçilmesini sağlar.

void _fastcall TForm1::Button7Click (TObject *Sender)

Bu fonksiyon, başlangıç küme merkezlerini belirlemektedir.

void _fastcall TForm1::Button8Click (TObject *Sender,
TMouseButton Button, TShiftState Shift, int X, int Y)

Bu fonksiyon, çalışılan veri setinin ekranda visual olarak görüntülenmesini sağlar.

void _fastcall TForm1::Button9Click (TObject *Sender)

Bu fonksiyon, yeni veri girişi yapabilmek için çalışılacak veri setini temizler.

void _fastcall TForm1::Button10Click (TObject *Sender)

Bu fonksiyon, küme sayısını otomatik olarak belirleyerek kümeleme işleminini gerçekleştirmek için kullanılır.

void _fastcall TForm1::FormDestroy (TObject *Sender)

Bu fonksiyon, Form1 üzerinde dinamik olarak oluşturulmuş nesneleri yok ederek belleğin boşalmasını sağlar.

void _fastcall TForm1::Shape1MouseDown(TObject *Sender,
MouseButton Button, TShiftState Shift, int X, int Y)

Bu fonksiyon, kullanıcının isteklerine bağlı olarak kümelerin renklerinin belirlenmesini sağlar.

void _fastcall TForm1::Size1Click(TObject *Sender)

Bu fonksiyon, parametrelerin ayarlanması için Form 4'ü çağırır.

- *TForm2* formuna bağlı fonksiyonlar:

void _fastcall TForm2::Button1Click(TObject *Sender)

Bu fonksiyon, grafik penceresini kapatır.

- *TForm3* formuna bağlı fonksiyonlar:

void _fastcall TForm3::Button1Click(TObject *Sender)

Bu fonksiyon, ekrana visual olarak verilen veri seti penceresini kapatır.

void _fastcall TForm3::CheckBox1Click(TObject *Sender)

Bu fonksiyon, koordinatların ölçeklerini verilere bağlı olarak otomatik şekilde ayarlar.

void _fastcall TForm3::FormPaint(TObject *Sender)

Bu fonksiyon, veri tabanındaki elemanları form üzerinde visual olarak görüntülemek için kullanılır.

void _fastcall TForm3::Image1MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)

Bu fonksiyon, form üzerinde mouse'un tıklandığı yerde nokta oluşturur ve aynı zamanda bu noktanın koordinatları ölçekler dikkate alınarak veri tabanına kaydedilir.

void_fastcall TForm3::FormClose(TObject *Sender, TCloseAction&Action)

Bu fonksiyon, Form 3 kapanırken, üzerinde dinamik oluşturulmuş olan nesneleri yok eder.

void _fastcall TForm3::FormShow(TObject *Sender)

Bu fonksiyon, form görüntülenirken koordinatların başlangıç ölçeklerini ve görüntülenen eleman sayılarını belirtir.

- *TForm4* formuna bağlı fonksiyonlar:

void _fastcall TForm4::Button1Click(TObject *Sender)

Bu fonksiyon, form penceresini kapatır.

void _fastcall TForm4::RadioGroup1Click(TObject *Sender)

Bu fonksiyon, küme geçerliliği yöntemi değiştirilirken, Form1'deki ilgili bilgiyi değiştirir.

void _fastcall TForm4::Edit3Exit(TObject *Sender)

Bu fonksiyon, bulanıklaştırma üssünün değerini ayarlar.

- *TForm5* formuna bağlı fonksiyonlar:

void _fastcall TForm5::Button1Click(TObject *Sender)

Bu fonksiyon, iki boyutlu kümelemenin sonuçlarını görsel olarak ekrana çıkarır.

void _fastcall TForm5::Button2Click(TObject *Sender)

Bu fonksiyon, formu kapatır.

- Bağımsız Fonksiyonlar:

float d(int i, int j)

Bu fonksiyon, *i* ve *j* numaralı kümeler arasındaki bulanık uzaklığı hesaplar.

float Dist(struct point a, struct point b)

Bu fonksiyon, a ve b noktaları arasındaki Euclid uzaklığını hesaplar.

void ini1graphs()

Bu fonksiyon, başlangıç küme merkezlerinin grafiğini çizer.

void TwoDimensional()

Bu fonksiyon, iki boyutlu kümelemenin sonuçlarının grafiğini çizdirmek için kullanılır.

float ro(int j)

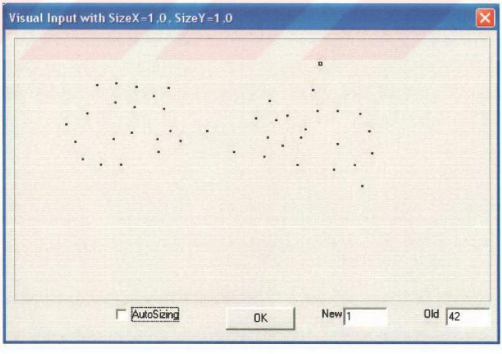
Bu fonksiyon, 3.5 kesiminde verilmiş olan *UDF* küme geçerliliği endeksinin belirlenmesinde kullanılan $\rho(x_j)$ fonksiyonunun değerini hesaplar.

6. UYGULAMALAR

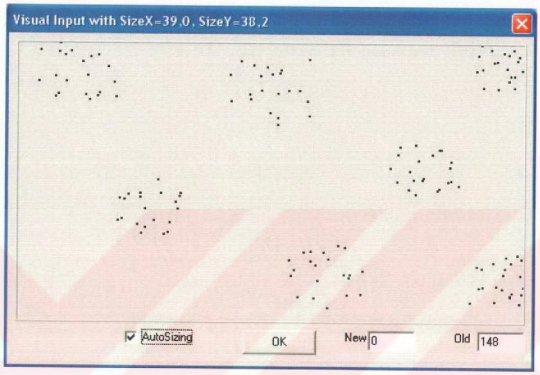
Bu bölümde iki uygulamaya yer verilmiştir. Oluşturulan program sisteminin çalışma prensiplerini göstermek ve dikkate alınan yöntemleri karşılaştırmak için ilk uygulamada, çeşitli küme sayıları için veri setleri türetilmiş, ikinci uygulamada ise IRIS veri seti kullanılmıştır.

6.1. Çeşitli Küme Sayıları İçin Türetilen Veri Setleri

Tezde dikkate alınan farklı yöntemlerin karşılaştırılması için çeşitli veri setleri türetilmiştir. Bu veri setleri türetilirken, tezde geliştirilmiş olan program sisteminde bulunan "Visual Input" teknolojisi kullanılmıştır (Şekil 6.1). Her küme sayısı için 10 tane olmak üzere, 2,3,...,7 tane küme içeren toplam 60 veri seti oluşturulmuştur. Küme geçerliliği ve başlangıç küme merkezi belirleme yöntemleri, bulanıklaştırma üssünün, $m = 2$ olduğu durum için karşılaştırılmıştır.



Şekil 6.1(a). 2 küme içeren veri setine örnek (2c-9).



Şekil 6.1(f). 7 küme içeren veri setine örnek (7c-9).

Başlangıç küme merkezleri, Mountain yöntemi (Tablo 6.1) ve K - en yakın komşuluk kuralı (Tablo 6.2) ile belirlenerek, küme sayısı, küme geçerliliği yöntemleri ile otomatik olarak bulunmuş ve sonuçlar sunulmuştur. Okunuş kolaylığı olması açısından, her iki tabloda da, farklı küme sayıları için bulunan sonuçlar, tablonun alt başlıkları (a,b,...,f) olarak numaralandırılmıştır.

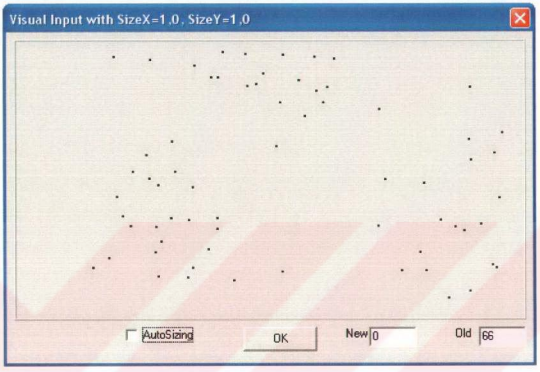
Tablo 6.3'te ise, başlangıç küme belirleme yöntemleri de dikkate alınarak yöntemlerin performansları karşılaştırılmıştır.

Tablo 6.1 ve Tablo 6.2 incelendiğinde, bölünme katsayısının, başlangıç kümeler Mountain yöntemi veya K - en yakın komşuluk kuralı ile belirlendiğinde, küme sayısının 2 ve 3 olduğu veri setlerinde %100,

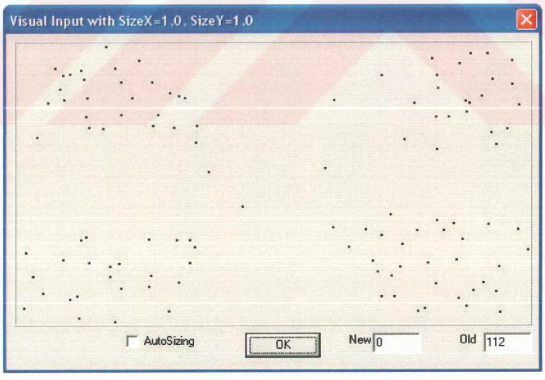
küme sayısının 4 olduğu durumda ise %90 başarı gösterdiği görülmektedir. Küme sayısı 5 olduğunda, bu başarı %60, küme sayısı 6 olduğunda ise, %70 olmuş ve yine, her iki başlangıç küme belirleme yöntemi ile aynı sonuç bulunmuştur. Ancak, optimal küme sayısının 7 olduğu veri setlerinde, başlangıç kümeler Mountain yöntemi ile belirlendiğinde %70, K - en yakın komşuluk kuralı ile belirlendiğinde ise %10 başarı sergilemiştir.

Sınıflandırma entropisi, küme sayısı 2 iken, tüm kümeleri doğru olarak belirleyebilmiştir. Küme sayısının 3 olduğu durumdan itibaren, sınıflandırma entropisinin performansı hızla düşmeye başlamıştır. Küme sayısı 3 iken, başlangıç küme belirleme yöntemlerinin her ikisi ile %60 başarı göstermiş; küme sayısı 4 olduğunda, Mountain yöntemi ile veri setlerinin %10'unun, K - en yakın komşuluk kuralı ile %20'sinin optimal küme sayısını doğru bulabilmiştir. Küme sayısı 5 iken, performans yine %20'de kalmış, küme sayısı 6 olduğunda ise hiçbir veri setinin optimal küme sayısını doğru belirleyememiştir. Her iki başlangıç küme belirleme yöntemiyle, küme sayısının 7 olduğu veri setlerinin %10'unu doğru bulmuştur.

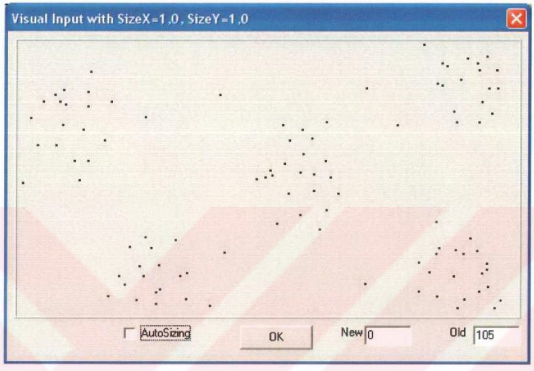
Ayrılma endeksi, diğer yöntemlere göre en kötü performansları göstermektedir. Küme sayısı 2 olduğunda, tüm veri setleri için doğru sonuç bulmuş, ancak küme sayısının 2'den fazla olduğu durumlarda performansı dramatik bir düşüş göstermiştir. Küme sayısı 3 iken veri setlerinin %30'unun optimal küme sayısını doğru olarak belirleyebilmiştir. Küme sayısı 4 iken, Mountain yöntemi ile %10 başarı göstermiştir. K - en yakın komşuluk kuralı kullanıldığında ise hiçbir veri



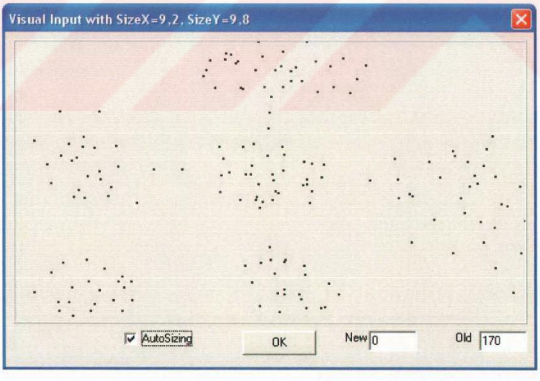
Şekil 6.1(b). 3 küme içeren veri setine örnek (3c-6).



Şekil 6.1(c). 4 küme içeren veri setine örnek (4c-6).



Şekil 6.1(d). 5 küme içeren veri setine örnek (5c-2).



Şekil 6.1(e). 6 küme içeren veri setine örnek (6c-6).

setinin küme sayısını doğru bulamamış ve küme sayılarının 5, 6 ve 7 olduğu veri setlerinde de hiç başarı gösterememiştir.

Fukuyamo-Sugeno küme geçerlilik endeksi, küme sayısının 3'ten fazla olduğu durumlarda en tutarlı sonuçları vermiştir. Küme sayısının 2 olduğu veri setleri için, doğru sonuçlar bulamamış, hatta optimal küme sayılarını 5-7 olarak belirlemiştir. Ancak, küme sayısının 3 olduğu veri setlerinde, Mountain yöntemi ile %40, K - en yakın komşuluk kuralı ile %50 performans göstermiştir. Optimal küme sayısı 4 iken, başarısı %70'lere çıkmıştır. Başlangıç kümeler, küme sayısı 5 iken Mountain yöntemi ile belirlendiğinde %100, K - en yakın komşuluk kuralı kullanıldığında ise %90 başarı sergilemiştir. Her iki başlangıç küme belirleme yöntemi ile, 6 küme olan veri setlerinin %90'ında başarılı olmuştur. Küme sayısının 7 olduğu grupta ise, başlangıç kümeler Mountain yöntemi ile belirlendiğinde %70, K - en yakın komşuluk kuralı kullanıldığında ise %100 başarı göstermiştir.

Tablo 6.1 Başlangıç kümeler Mountain yöntemi ile belirlendiğinde, küme sayısını belirlemek için farklı küme geçerliliği yöntemlerinin farklı küme sayıları için (a) 2, (b) 3, (c) 4, (d) 5, (e) 6 ve (f) 7 karşılaştırılması. (PC: Bölünme Katsayısı, CE: Sınıflandırma Entropisi, SI: Ayrılma Endeksi, FS: Fukuyamo-Sugeno Endeksi.)

Tablo 6.1(a). Küme sayısının iki olduğu durum

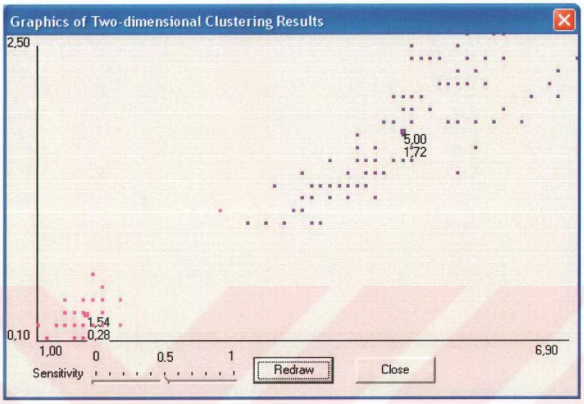
Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
2c-1	50	2	2	2	2	7
2c-2	40	2	2	2	2	7
2c-3	44	2	2	2	2	7
2c-4	39	2	2	2	2	6
2c-5	54	2	2	2	2	7
2c-6	36	2	2	2	2	7
2c-7	40	2	2	2	2	5
2c-8	58	2	2	2	2	5
2c-9	42	2	2	2	2	6
2c-10	35	2	2	2	2	6

Tablo 6.1(b). Küme sayısının üç olduğu durum

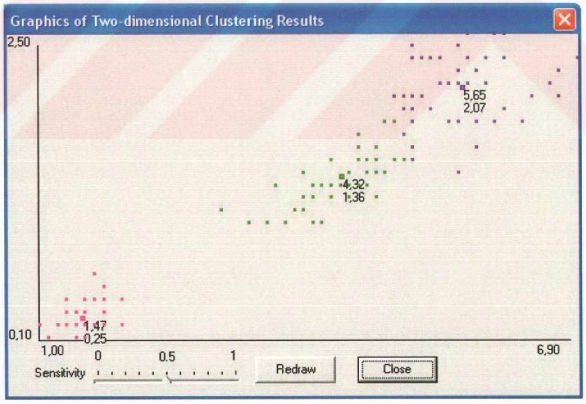
Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
3c-1	58	3	3	3	3	3
3c-2	54	3	3	3	2	6
3c-3	52	3	3	2	2	6
3c-4	52	3	3	2	2	5
3c-5	56	3	3	3	2	3
3c-6	66	3	3	2	2	5
3c-7	76	3	3	3	2	4
3c-8	62	3	3	3	3	3
3c-9	73	3	3	3	3	3
3c-10	68	3	3	2	2	7

Tablo 6.4. IRIS veri seti için küme geçerliliği yöntemlerinin performanslarının farklı başlangıç küme belirleme yöntemleri ile karşılaştırılması.

Küme sayısı	Küme geçerliliği / Başlangıç küme	PC	CE	SI	FS
2	Mountain	**0.9218**	**0.1428**	**2.6007**	-381.6591
	K-En Yakın Komşuluk	**0.9209**	**0.1442**	**2.5697**	-381.8531
3	Mountain	0.8560	0.2643	3.4461	**-448.6631**
	K-En Yakın Komşuluk	0.8560	0.2643	3.4458	**-448.6432**
4	Mountain	0.7732	0.4198	2.9769	-391.0227
	K-En Yakın Komşuluk	0.7998	0.3831	4.3519	-444.8297
5	Mountain	0.7071	0.5401	3.2694	-369.4211
	K-En Yakın Komşuluk	0.7647	0.4667	6.0241	-432.7523
6	Mountain	0.6761	0.6138	3.0462	-356.9891
	K-En Yakın Komşuluk	0.6765	0.6132	2.9717	-357.4330
7	Mountain	0.6702	0.6456	3.1388	-364.3065
	K-En Yakın Komşuluk	0.6689	0.6577	3.2100	-364.7177



Şekil 6.3. Bölünme katsayısı, sınıflandırma entropisi ve ayrılma endeksi ile bulunan kümeler (iki küme).

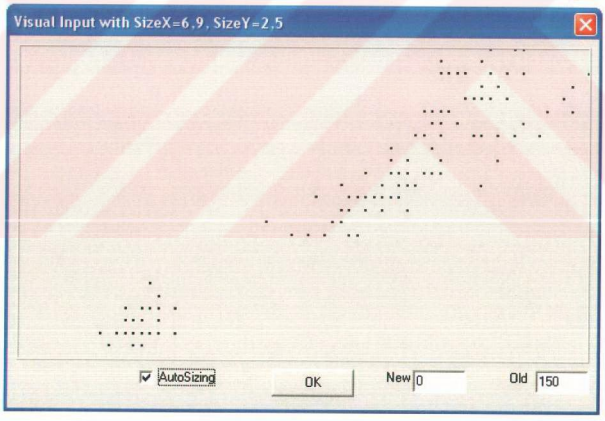


Şekil 6.4. Fukuyamo-Sugeno küme geçerliliği endeksi ile bulunan kümeler (üç küme).

6.2. IRIS Veri Seti

Bu kısımda, Anderson (1935) ve Fisher'in (1936), 3 ayrı çiçek türüne (Setosa, Versicolor ve Virginica) ilişkin ölçümleri içeren ve genel olarak IRIS olarak bilinen biyometrik veri seti değerlendirilecektir.

IRIS, kümeleme algoritmaları ve küme geçerliliği yöntemlerini test etmek için kullanılan standart bir veri setidir. IRIS veri setinde, taç (petal) ve çanak (sepal) yaprakların en ve boy ölçümlerini içeren her sınıf (tür) 50 gözlem içermektedir. Yani 150 çiçekten her biri 4-boyutlu ölçüm uzayında bir nokta olarak temsil edilmektedir.



Şekil 6.2. IRIS veri setinin iki boyutu için (taç yaprakların uzunluk ve genişlik ölçümleri) visual görüntüsü.

Pal ve Bezdek (1997), IRIS veri setindeki üç sınıftan iki tanesinin önemli ölçüde çakıştığından dolayı, küme sayısının 2 veya 3 olmasında çelişkiye düşülebileceğini söylemişlerdir. Halgamuge ve Glesner (1994), sadece iki özellik kullanılarak çok iyi bir sınıflandırma yapılabileceğini göstermişlerdir. Rezaee et al. (1998), yalnızca taç yaprak uzunluklarını kullanarak sınıflandırma yapmışlardır.

Bu kısımda, taç yaprakların uzunluk ve genişlik ölçüm değerleri kullanılarak, başlangıç küme belirleme ve küme geçerliliği yöntemlerinin performansları değerlendirilmiştir. Bu ölçümler Şekil 6.2.'de visual olarak gösterilmiştir.

Bölüm 6.1.'de yapıldığı gibi, yine farklı başlangıç küme belirleme yöntemleri kullanılarak, küme geçerliliği yöntemleriyle küme sayısı otomatik olarak belirlenmiş ve bulunan sonuçlar Tablo 6.4'te sunulmuştur. Koyu siyah yazılan değerler, her yöntemin bulduğu optimal küme sayısını göstermektedir.

Tablo 6.4 incelendiğinde, bölünme katsayısı, sınıflandırma entropisi ve ayrılma endeksinin optimal küme sayısını 2, Fukuyamo-Sugeno geçerlilik endeksinin ise 3 olarak bulduğu görülmektedir. Bulunan kümeler, sırasıyla, Şekil 6.3 ve Şekil 6.4'te gösterilmiştir.

Tablo 6.2(e). Küme sayısının altı olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
6c-1	125	6	6	2	2	6
6c-2	160	6	6	2	2	6
6c-3	155	6	6	2	2	6
6c-4	150	6	6	2	2	6
6c-5	190	6	2	2	2	6
6c-6	170	6	6	2	2	7
6c-7	140	6	2	2	2	6
6c-8	170	6	6	2	2	6
6c-9	115	6	6	2	2	6
6c-10	205	6	2	2	2	7

Tablo 6.2(f). Küme sayısının yedi olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
7c-1	110	7	7	2	2	7
7c-2	125	7	7	2	2	7
7c-3	190	7	7	2	2	7
7c-4	90	7	7	2	2	7
7c-5	130	7	7	2	2	7
7c-6	105	7	7	2	2	7
7c-7	130	7	7	2	2	7
7c-8	155	7	2	2	2	7
7c-9	148	7	7	2	2	7
7c-10	170	7	7	7	2	7

Tablo 6. 3. Küme geçerliliği yöntemlerinin performanslarının farklı başlangıç küme belirleme yöntemleri ile karşılaştırılması.

Küme sayısı	Küme geçerliliği / Başlangıç küme	PC	CE	SI	FS
2	Mountain	%100	%100	%100	%0
	K-En Yakın Komşuluk	%100	%100	%100	%0
3	Mountain	%100	%60	%30	%40
	K-En Yakın Komşuluk	%100	%60	%30	%50
4	Mountain	%90	%10	%10	%70
	K-En Yakın Komşuluk	%90	%20	%10	%70
5	Mountain	%60	%20	%0	%100
	K-En Yakın Komşuluk	%60	%20	%0	%90
6	Mountain	%70	%0	%0	%90
	K-En Yakın Komşuluk	%70	%0	%0	%90
7	Mountain	%70	%90	%0	%70
	K-En Yakın Komşuluk	%10	%10	%0	%100

Mountain yönteminin bazı veri setlerindeki farklı performansının, kullanılan α, β ve δ parametrelerinin seçiminden kaynaklandığını düşünmekteyiz. Belirli veri setlerinde kullanılabilmek için parametrelerin daha iyi ayarlanması ayrıca bir araştırmayı gerektirmektedir. Bu türlü bir girişime, Velthuizen et al. (1997) Geliştirilmiş Mountain yönteminde yer vermiştir. Ayrıca, kümeleme yalnızca $m = 2$ için uygulanmıştır. Bulanıklaştırma üssünün farklı değerleri için, bu sonuçlar farklılık gösterebilir.

Tablo 6. 2. Başlangıç kümeler *K*-en yakın komşuluk kuralı ile belirlendiğinde, küme sayısını belirlemek için farklı küme geçerliliği yöntemlerinin farklı küme sayıları için (a) 2, (b) 3, (c) 4, (d) 5, (e) 6 ve (f) 7 karşılaştırılması.

Tablo 6.2(a). Küme sayısının iki olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
2c-1	50	2	2	2	2	4
2c-2	40	2	2	2	2	7
2c-3	44	2	2	2	2	6
2c-4	39	2	2	2	2	5
2c-5	54	2	2	2	2	7
2c-6	36	2	2	2	2	7
2c-7	40	2	2	2	2	4
2c-8	58	2	2	2	2	6
2c-9	42	2	2	2	2	6
2c-10	35	2	2	2	2	5

Tablo 6.2(b). Küme sayısının üç olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
3c-1	58	3	3	3	2	3
3c-2	54	3	3	3	2	4
3c-3	52	3	3	2	2	6
3c-4	52	3	3	2	2	5
3c-5	56	3	3	3	3	3
3c-6	66	3	3	2	2	5
3c-7	76	3	3	3	2	3
3c-8	62	3	3	3	3	3
3c-9	73	3	3	3	3	3
3c-10	68	3	3	2	2	6

Tablo 6.2(c). Küme sayısının dört olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
4c-1	68	4	4	4	2	4
4c-2	100	4	4	2	2	5
4c-3	102	4	4	2	2	4
4c-4	66	4	4	2	2	6
4c-5	92	4	4	4	4	4
4c-6	112	4	4	2	2	4
4c-7	102	4	4	2	2	4
4c-8	96	4	4	2	2	4
4c-9	52	4	4	2	2	4
4c-10	82	4	2	2	2	5

Tablo 6.2(d). Küme sayısının beş olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
5c-1	100	5	5	2	2	5
5c-2	105	5	5	5	2	5
5c-3	145	5	5	5	2	5
5c-4	115	5	2	2	2	5
5c-5	150	5	2	2	2	5
5c-6	130	5	5	2	2	5
5c-7	130	5	5	2	2	5
5c-8	110	5	5	2	2	5
5c-9	120	5	3	2	2	7
5c-10	105	5	2	2	2	5

Tablo 6.1(c). Küme sayısının dört olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
4c-1	68	4	4	4	2	4
4c-2	100	4	4	2	2	5
4c-3	102	4	4	2	2	4
4c-4	66	4	4	2	2	6
4c-5	92	4	4	2	4	4
4c-6	112	4	4	2	2	4
4c-7	102	4	4	2	2	4
4c-8	96	4	4	2	2	4
4c-9	52	4	4	2	2	4
4c-10	82	4	2	2	2	6

Tablo 6.1(d). Küme sayısının beş olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
5c-1	100	5	5	2	2	5
5c-2	105	5	5	5	2	5
5c-3	145	5	5	5	2	5
5c-4	115	5	2	2	2	5
5c-5	150	5	2	2	2	5
5c-6	130	5	5	2	2	5
5c-7	130	5	5	2	2	5
5c-8	110	5	5	2	2	5
5c-9	120	5	3	2	2	5
5c-10	105	5	2	2	2	5

Tablo 6.1(e). Küme sayısının altı olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
6c-1	125	6	6	2	2	6
6c-2	160	6	6	2	2	6
6c-3	155	6	6	2	2	6
6c-4	150	6	6	2	2	6
6c-5	190	6	2	2	2	6
6c-6	170	6	6	2	2	6
6c-7	140	6	2	2	2	6
6c-8	170	6	6	2	2	6
6c-9	115	6	6	2	2	6
6c-10	205	6	2	2	2	7

Tablo 6.1(f). Küme sayısının yedi olduğu durum

Veri seti	Nokta sayısı	Expert	PC	CE	SI	FS
7c-1	110	7	7	2	2	7
7c-2	125	7	7	2	2	7
7c-3	190	7	7	2	2	7
7c-4	90	7	7	2	2	7
7c-5	130	7	7	2	2	7
7c-6	105	7	7	2	2	7
7c-7	130	7	2	2	2	6
7c-8	155	7	2	2	2	6
7c-9	148	7	6	2	2	6
7c-10	170	7	7	7	2	7

7. SONUÇ

Bu tez çalışmasında, çeşitli veri setlerinde kümeleme işleminin yapılması için literatürde sıklıkla kullanılan yöntemlerin sonucunu karşılaştırmak ve görsel olarak değerlendirebilmek imkanını sağlayan program sistemi oluşturulmuştur. Program sistemi, Borland C++ Builder 6.0 sisteminde yazılmış ve Intel Pentium IV, 1.7 GHz, 256 MB RAM teknik özellikleri olan bilgisayarda uygulanmıştır.

Program sistemin esas özelliği bir veya iki ölçülü veri setini, kümeleme sonuçlarını, başlangıç ve sonuç küme merkezlerini görsel olarak ekranda görüntüleme olanağını vermesidir. Bunun yanı sıra, belirli bir şekilde olan veri setleri türetmek isteği olduğu zaman, benzer veri setini görsel olarak ekranda oluşturma imkanı da sağlanmaktadır. Sunulan program sisteminin diğer özellikleri aşağıdaki gibi sıralanabilir:

- çeşitli kümeleme yöntemlerinin seçilebilmesi;
- bulanıklığı yansıtan üs (m) parametresinin seçilebilmesi;
- başlangıç kümeleri oluşturma yöntemlerinin seçilebilmesi;
- başlangıç küme sayısının direkt verilebilmesi;
- çeşitli küme geçerliliği endekslerine dayalı olarak küme sayısının otomatik olarak belirlenebilmesi;
- bir ve iki boyutlu veri setinin grafik olarak ekrana çıkarılması;
- bir ve iki boyutlu kümeleme sonuçlarının grafik olarak ekrana çıkarılması;
- verilere bağlı olarak ekranın ölçeklerinin ayarlanabilmesi.

Oluşturulan program sisteminin, araştırmacılara veri setini analiz etmek için çeşitli yöntem seçenekleri sunmasının yanı sıra, bulunan yeni yöntemlerin performanslarını değerlendirmek için mevcut yöntemlerle karşılaştırma olanağı sağlaması gibi avantajları da bulunmaktadır.

Tezde araştırılarak programı yazılmış yöntemler aşağıdaki gibi sıralanabilir:

- bir ve iki boyutlu veriler için bulanık kümeleme algoritması;
- başlangıç kümelerin oluşturulması için Mountain yöntemi;
- başlangıç kümelerin oluşturulması için K – En Yakın Komşuluk yöntemi;
- küme sayısının belirlenmesi için Bölünme Katsayısı yöntemi;
- küme sayısının belirlenmesi için Sınıflandırma Entropisi yöntemi;
- küme sayısının belirlenmesi için Ayrılma Endeksi yöntemi;
- küme sayısının belirlenmesi için Fukuyamo-Sugeno yöntemi.

Program sisteminde kullanılan yöntemleri karşılaştırmak için iki uygulamaya yer verilmiştir. İlk uygulamada, programda geliştirilmiş olan "Visual Input" teknolojisi kullanılarak, çeşitli küme sayıları için, farklı sayılarda eleman içeren toplam 60 veri seti türetilmiş ve bu veri setlerinde sözü geçen yöntemler karşılaştırılarak sonuçlar tablolarla özetlenmiştir (Tablo 6.1, 6.2 ve 6.3).

Tablolar incelendiğinde, gerçeğe en yakın sonuçları bölünme katsayısı ve Fukuyamo-Sugeno küme geçerliliği yöntemlerinin bulduğu söylenebilir. Bölünme katsayısının performansı, küme sayısı 4'ten fazla

olduğunda %60'lara kadar düşmekte, hatta küme sayısı 7 iken, başlangıç kümeler K - en yakın komşuluk kuralı ile belirlendiğinde küme sayısını belirleme performansı %10'a düşmektedir (Tablo 3f). Fukuyamo-Sugeno küme geçerliliği yöntemi ise, küme sayısı 2 iken, hiçbir veri setinin küme sayısını doğru belirleyememiştir (Tablo 3a). Ancak, bölünme katsayısının aksine, küme sayısı arttıkça, bu yöntemin performansında bir artış gözlenmektedir. Başlangıç kümeler hangi yöntemle belirlenirse belirlensin, performansı yaklaşık olarak aynıdır.

Ayrılma endeksi ve sınıflandırma entropisinin performansları, sırasıyla, küme sayısı 3 ve 4'ten fazla olduğunda düşmeye başlamış, hatta, küme sayısı 5'ten fazla olduğunda ayrılma endeksi, 6'dan fazla olduğunda ise sınıflandırma entropisi, hiçbir veri setinin küme sayısını doğru olarak belirleyememiştir.

İkinci uygulamada ise Anderson ve Fisher'in meşhur IRIS veri seti kullanılmış ve yöntemlerin bulduğu optimal küme sayıları karşılaştırılmıştır. Tablo 6.4, yöntemlerin her küme geçerlilik endeksinin Mountain yöntemi ve K - en yakın komşuluk kuralı ile bulunan değerlerini göstermektedir. Buna göre, IRIS veri setinin gerçekte 3 olan küme yapısını en doğru şekilde Fukuyamo-Sugeno geçerlilik endeksi belirleyebilmiştir. Diğer yöntemler ise, küme sayısını 2 olarak belirlemişlerdir. Bu değer aslında yanlış olmayan, ancak üst üste çakışan kümeleri ayırt edememiş olan endekslerin bir göstergesidir.

Kümelenen birimler görülebilir alt kümelerde çok iyi ayrılmış bile olsalar, kümeleme sırasında bu yapının bulunamamasının nedenlerinden bazıları aşağıda verilmiştir:

➢ Kümeler arasındaki farkı görmek için, birimlerin sayısal gösterimleri yeterli olmayabilir.

➢ Veriler istenilen altyapıyı yansıtsa bile, bazen kullanılan kümeleme algoritması bu altyapıyı özetleyemeyebilir. Örneğin, elipsoidal kümeler arayan bir kümeleme algoritması, başka tür kümeleri bulamayacaktır.

➢ Bazen birimlerin küme yapısı olabilir ve veriler de bu yapıyı yansıtabilir. Hatta, kullanılan algoritma da bu yapıyı bulabilecek türden olabilir. Ancak, algoritmada, kümeleri başarılı ayırt edebilecek parametreler doğru ayarlanmamış olabilir. Örneğin, Mountain yönteminin bazı veri setlerindeki farklı performansı, kullanılan α , β ve δ parametrelerinin seçiminden kaynaklanmış olabilir. Belirli veri setlerinde kullanılabilmek için parametrelerin daha iyi ayarlanması başka bir araştırma konusu olabilir. Ayrıca, bulanıklaştırma üssünün farklı değerleri için, bu sonuçlar farklılık gösterebilir.

➢ Yukarıda sayılan tüm bu engeller ortadan kaldırılmış olsa bile, geçerlilik endeksleri, bulunmuş olan kümelerin aslında çok iyi olduğunu bulmakta başarısız olabilir.

Bütün bu sayılan nedenlerden dolayı, kümelemenin çok dikkatli ve uygun yöntemler kullanmaya özen gösterilerek gerçekleştirilmesi gerekmektedir.

Gelecek araştırmalarımızda, bulanık kümeleme alanında mevcut yöntemlerin performanslarının incelenmesi ve yeni yöntemlerin oluşturulması için çalışmaların devam ettirilmesi amaçlanmaktadır.

KAYNAKLAR DİZİNİ

Agrawal, R., Imielinski, T., Swami, A., 1993, Mining Association Rules between sets of items in large databases, *ACM SIGMOD*, 27, No.3, p.207-216.

Anderson, E., 1935, The IRIS of the Gaspe peninsula, *Bull. Amer. Iris Soc.*, 59, p.2-5.

Araki, S., Nomura, H., Wakami, N., 1993, Segmentation of thermal images using Fuzzy c-means algorithm, in Proc., Second IEEE Int. Conf. On Fuzzy Systems, p.719-724.

Baykal, N., Beyan, T., 2004, Bulanık Mantık İlke ve Temelleri, Bıçaklar Kitabevi, Ankara, 413s.

Belace, N., Hansen, P., Mladenovic, N., 2002, Fuzzy J-means: a new heuristic for fuzzy clustering, *Pattern Recognition*, 35, p.2193-2200.

Bellman, R.E., Kalaba, R., Zadeh, L.A., 1966, Abstraction and pattern classification, *J. Math. Anal. Appl.*, 13, p.1-7.

Bezdek, J.C., 1973, Fuzzy Mathematics in Pattern Classification, PhD Thesis, Cornell University, Ithaca, NY.

Bezdek, J.C., 1974, Cluster validity with fuzzy sets, *J. Cybernetics*, 3, p.58-73.

Bezdek, J.C.,1975, Mathematical models for systematics and taxonomy, in Proc. 8th Int. Conf. On Numerical Taxonomy, G.Estrabrook (ed.), Freeman, San Fransisco, p.143-166.

Bezdek, J.C., Dunn J.C., 1975, Optimal Fuzzy partition: A heuristic for estimating the parameters in a mixture of normal distributions, *IEEE Trans. Comput. C*-24, p.835-838.

Bezdek, J.C., 1980(a), A Convergence theorem for the fuzzy ISODATA clustering algorithms, *IEEE Trans. Pattern Anal. Machine Intell.* Pami-2, No.1, p.1-8.

KAYNAKLAR DİZİNİ (devam)

Bezdek, J.C., 1980(b), Cluster validity with fuzzy sets, *J. Cybernetics,* 3, p.58-73.

Bezdek, J.C., 1981, Pattern Recognition with Fuzzy Objective Function Algorithms , Plenum Press, New York.

Bezdek, J.C., Coray, C., Gunderson, R., Watson, J., 1981, Detection and characterization of cluster substructure, I, II, *SIAM J. of Applied Mathematics*, 40, p.339-371.

Bezdek, J.C., Hathaway, R., Howard, L., Windham, M., 1987, Local convergence analysis of a grouped variable version of coordinate descent, *J. Optimization Theory and Appl.*, 54, p.471-477.

Bezdek, J.C., Pal, N.R., 1998, Some new indexes of cluster validity, *IEEE Trans. SMC, Part B Cybernet.* 28, p.301-315.

Bobrowski, L., Bezdek J.C., 1991, c-Means clustering with I_1 and I_∞ norms, *IEEE Trans. On Systems, Man and Cybernetics*, 21, p.545-554.

Calinski, R.B., Harasabasz, J., 1974, A denrite method for cluster analysis, *Communications in Statistics*, 3, 1-27.

Chepoi, V., Dumitrescu, D., 1992, Cluster analysis and facility location problems, Univ. Of Cluj-Napoca, Research Seminar on Computer Science, 5, p.139-186.

Chiu, S., 1994, Fuzzy model identification based on cluster estimation, *J. Intel. Fuzzy Systems*, 2, No.3, p.209-278.

Davies, D.L., Bouldin, D.W., 1979, A cluster separation measure, *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 1, p.224-227.

Dumitrescu, D., 1986, Numerical methods in fuzzy hierarchial pattern recognition, I , *Studia Univ. Babeş-Bolyai Ser. Math.*, 31(4), p.31-36.

KAYNAKLAR DİZİNİ (devam)

Dumitrescu, D., 1987, Numerical methods in fuzzy hierarchial pattern recognition, II , *Studia Univ. Babeş-Bolyai Ser. Math.*, 32(1), p.24-30.

Dumitrescu, D.,1988, Hierarchial classification for linear clusters, *Studia Univ. Babeş-Bolyai Ser. Math.*, 33(3), p.48-51.

Dumitrescu, D, 1997, Fuzzy hierarchial classification methods in analytical chemistry, in Fuzzy Logic in Chemistry, D.M. Rouvray (Ed.), Academic Press, New York, p.321-356.

Dumitrescu, D., 2000, Mathematical Principles of Classification Theory, Romanian Academy Publishing House, Bucharest.

Dumitrescu, D., Lazerini, B., Jain, L.C., 2000, Fuzzy Sets and Their Application to Clustering and Training, CRC Press LLC.

Dunn, J.C., 1973, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *J.Cybernetics*, 3, No.3, p.32-57.

Dunn, J.C., 1974, Well-separated clusters and optimal fuzzy partitions, *J. Cybernetics*, 4, p.95-104.

Edwards, A.W., Cavalli-Sforza, L.L, 1965, A method for cluster analysis, *Biometrics*, 21, p.362-375.

Fisher, R.A., 1936, The use of multiple measurements in taxonomic problems, *Annals of Eugenics*, 7, No.11, p.179-188.

Frayley, C., Raftery, A., 1998, How many clusters? Which clustering method? Answers via model-based cluster analysis, Technical Report 329, University of Washington, Dept. of Statistics.

Fu, A., Knok, C.M., Wong, M.H., 1998, Mining Association Rules database, *ACM SIGMOD*, 27, No.3, p.41-46.

KAYNAKLAR DİZİNİ (devam)

Fukuyamo, Y., Sugeno, M., 1989, A new method for choosing the number of clusters for the fuzzy c-means method, Proc. 5th Fuzzy System Symp., p. 247-250 (in Japanese).

Gath, I., Geva, A., 1989, Unsupervised optimal fuzzy clustering, *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 11, p.773-781.

Gordon, A.D., 1981, Classification, University Pres, Cambridge.

Gordon, A.D., 1996, Null Models in cluster validation, in From Data to KnowledgeUniversity (eds W. Gaul and D.Pfeifferr), p. 32-44, new York: Springer.

Halgamuge, S.K., Glesner, M., 1994, Neural networks in designing fuzzy systems for real world applications, *Fuzzy Sets and Systems,* 65, No.1, p.1-12.

Hathaway, R.J., Bezdek, J.C., 1993, Switching regression models and Fuzzy clustering, *IEEE Trans. On Fuzzy Systems*, 1, p.195-204.

Hathaway, R.J., Bezdek, J.C., 1995, Optimization of clustering criteria by reformulation, *IEEE Trans. On Fuzzy Systems*, 3, p.241-245.

Hathaway, R.J., Bezdek, J.C., 2003, Visual cluster validity for ptototype generator clustering models, *Pattern Recognition Letters*, 24, p.1563-1569.

Hardy, A., 1996, On the number of clusters, *Computational Statistics and Data Analysis*, 23, p. 83-96.

Hartigan, J.A., 1975, Clustering Algorithms, New York:Wiley.

Jang, J-S., R., Sun, C-T., Mizutani, E., 1997, Neuro-Fuzzy and Soft Computing, Upper Saddle River, NJ : Prentice Hall.

Kandel, A., Last, M., Bunke, H., 2001, Data Mining and Computational Intelligence, Physica-Verlag.

KAYNAKLAR DİZİNİ (devam)

Kass, R.E., Raftery, A.E., 1995, Bayesian Factors, *J. Amer. Stat. Ass.*, 90, p. 773-795.

Kaufman, L., Rouseeuw, P., 1990, Finding Groups in Data: An Introduction to Cluster Analysis, New York:Wiley.

Kendall, S.M., Stuart, A., 1976, The Advanced Theory of Statistics, Vol. 3, 3th ed., Charles Griffin&Company Ltd., London.

Khotanzad, A., Bouarfa, A., 1990, Image segmentation by a paralel, non-parametric histogram based clustering algorithm, *Pattern Recognition*, 23, No.9, p.961-973.

Krishnapuram, R., Keller, J.M., 1993, A possibilistic approach to clustering, *IEEE Trans. On Fuzzy Systems*, 1, p.98-110.

Krzanowski, W.J., Lai, Y.T., 1988, A Criterion for Determining the Number of Groups in a Data Set Using Sum-Of-Squares Clustering, *Biometrics*, 44, p.23-34.

Marriott, F.H.C., 1971, Practical problems in a methos of cluster analysis, *Biometrics*, 27, p.501-514.

Milligan, G.W., Cooper, M.C., 1985, An examination of procedures for determining the number of clusters in a data set, *Psychometrika*, 50, p.159-179.

Nasibov, E.N., 2000, Methods of processing of fuzzy information in decision-making problems, Baku, Elm, (Monograph in Russian).

Nasibov, E.N., 2002, A problem of identification of states of a system from fuzzy values of informational features, *Automatic Control and Computer Sciences*, 36, No.2, p.27-33.

Nasibov, E.N., 2002, Identification of states of complex Systems with estimation of admissible Measurement errors on the basis of Fuzzy information, *Cybernetics and Systems Analysis*, 38, No. 1, p.53-59.

KAYNAKLAR DİZİNİ (devam)

Nasibov, E., Senol, S., Ulutagay, G., 2004, A Visual Processing System For Fuzzy Clustering, International Conf. On Informatics 1, Çeşme, Izmir, Turkey.

Negoita, C.V., 1973, On the application of the fuzzy sets separation theorem for automatic classification in information retrieval systems, *Information Science*, 6, p.279-286.

Pal, N.R., Bezdek, J.C., 1995, On cluster validity for the fuzzy c-means model, *IEEE Trans. On Fuzzy Systems*, 3, No.3, p.370-379.

Pal, N.R., Bezdek, J.C., 1997, Correction to on cluster validity for the fuzzy c-means model, *IEEE Trans. On Fuzzy Systems*, 5, No.1, p.152-153.

Pedrycz, W., Gomide, F., 1998, An Introduction to Fuzzy Sets, Massachsetts Institute of Technology.

Rezaee, M.R., Lelieveldt, B.P.F., Reiber, J.H.C., 1997, A new cluster validity index for the Fuzzy c-mean, *Pattern Recognition Letters*, 19, p.237-246.

Roubens, M., 1978, Pattern classification problems and fuzzy sets, *Fuzzy Sets and Systems*, 1, p.239-253.

Roubens, M., 1982, Fuzzy clustering algorithms and their validity, *J. of Operational Research*, 10, p.294-301.

Runkler, T.A., Bezdek, J.C., 2003, Web mining with relational clustering, *Int. Journal of Approximate Reasoning*, 32, p.217-236.

Ruspini, E.H., 1969, A new approach to clustering, *Information and Control*, 15, p.22-32.

Ruspini, E.H., 1970, Numerical methods for Fuzzy clustering, *Information Science*, 2, p.583-593.

KAYNAKLAR DİZİNİ (devam)

Ruspini, E.H., 1973, New experimental results in fuzzy clustering, *Information Science*, 6, p.273-284.

Sato, M., Sato, Y.,Lain, L.C., 1997, Fuzzy Clustering Models and Applications, Physica-Verlag Heidelberg, New York.

Scott, A., Symons, M., 1971, Clustering methods based on likelihood ratio criteria, *Biometrics*, 27, p.387-397.

Sugar, C.A., James, G.M., 2003, Finding the number of clusters in a data set: an information-theoretic approach, *J.Am. Statis. Ass.*, 98, No.463, p.750-763.

Tibshirani, R., Walther, G., Hastie, T., 2001, Estimating the numbers in a data set via the gap statistic, J.Royal Statist. Soc., 63, Part 2, p.411-423.

Türkşen, İ.B., 1985, Bulanık Kümeler Kuramı ve Uygulamaları, *Yöneylem Araştırması Dergisi,* 4, No.1, p. 1-15.

Velthuizen, R.P., Hall, L.O., Clarke L.P., Silbiger, M.L., 1997, An investigation of mountain method clustering for large data sets, *Pattern Recognition,* 30, No.7, p.1121-1135.

Windham, M., 1981, Cluster validity for fuzzy clustering algorithms, *Fuzzy Sets and Systems*, 5, p.177-185.

Windham, M., 1982, Cluster validity for the fuzzy c-means clustering algorithm, *IEEE Trans. On Pattern Anal. Machine Intell.*, PAMI-4, p.357-363.

Wolkenhauer, O., 2001, Data Engineering: Fuzzy Systems and Data Analysis, New York : Wiley.

Xie, X.L., Beni, G., 1991, A validity for Fuzzy clustering, *IEEE Trans. On Pattern Anal. Machine Intell.*, 3, p.841-846.

KAYNAKLAR DİZİNİ (devam)

Yager, R.R., Filev, D.P., 1994(a), Approximate clustering via the mountain method, *IEEE Trans. On Systems, Man and Cybernetics,* 24(8), p.1279-1284.

Yager, R.R., Filev, D.P., 1994(b), Generation of Fuzzy rules by mountain clustering, *J.Intell. Fuzzy Systems*, 2, p.209-219.

Yen, J., Langari, R.,1999, Fuzzy Logic-Intelligence, Control and Information, Upper Saddle River, New Jersey : Prentice Hall.

Zadeh, L.A., 1965, Fuzzy Sets, *Inform. Control*, 8, p.338-353.

Zahid, N., Limouri, M., Essaid, A., 1999, A new cluster validity for Fuzzy clustering, *Pattern Recognition*, 32, p.1089-1097.

Zahid, N., Abouelala, O., Limouri, M., Essaid, A., 2001, Fuzzy clustering based on K-nearest-neighbours rule, *Fuzzy Sets and Systems,* 120, p.239-247.

EKLER (Programın kodu)

```
//---------------------------------------------------------

#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "UTwoDimensional_Clustering.h"
#include "UGozde4Unit2.h"
#include "UGozdeUnit3.h"
#include "GozdeUnit4.h"
#include "GozdeUnit5.h"
#include "UAboutBox.h"
#include "UPass.h"
#include <math.h>
#define NCLUST 7
#define NROWS 1000
#define KM0 30
#define S  2
void TwoDimensional();
void ini1graphs();
float max[S],min[S],D[S],JK;
float ro(int i);
bool dostup=false;
struct point{float x1,x2;}
v[NCLUST],vv[NCLUST],v0[NCLUST];
float u[NCLUST][NROWS];
float Dist(struct point a, struct point b);
float d(int i, int j);
float EM;
int K,N;
struct point x[NROWS],xx;
TColor
COL[NCLUST]={clFuchsia,clPurple,clGreen,clAqua,clYellow,
                                          clRed,clBlue};
TEdit *ED1[NCLUST];
TEdit *ED2[NCLUST];
TLabel *LAB[NCLUST];
TShape *REC[NCLUST];
extern TShape *ts[1000];
int Ni=-1,G1,G2;
float Na,Nb;

bool OPENTABLE=false;

//---------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma link "CSPIN"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------

fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
for(int i=0;i<NCLUST;i++){
  REC[i]=new TShape(this);
  REC[i]->Parent=Form1;
  REC[i]->Left=140;
  REC[i]->Top=72+i*24;
  REC[i]->Height=16;
  REC[i]->Width=10;
  REC[i]->Brush->Color=COL[i];
  REC[i]->OnMouseDown=TForm1::Shape1MouseDown;
  REC[i]->Shape=stRoundRect;
  REC[i]->Tag=i;
  REC[i]->Show();
}
for(int i=0;i<NCLUST;i++){
  ED1[i]=new TEdit(this);
  ED1[i]->Parent=Form1;
  ED1[i]->Left=30;
  ED1[i]->Top=70+i*24;
  ED1[i]->Height=20;
  ED1[i]->Width=50;
  ED1[i]->Show();
  ED2[i]=new TEdit(this);
  ED2[i]->Parent=Form1;
  ED2[i]->Left=85;
  ED2[i]->Top=70+i*24;
  ED2[i]->Height=20;
  ED2[i]->Width=50;
  ED2[i]->Show();
}
for(int i=0;i<NCLUST;i++){
  LAB[i]=new TLabel(this);
  LAB[i]->Parent=Form1;
  LAB[i]->Left=8;
  LAB[i]->Top=70+i*24;
  LAB[i]->Height=20;

  LAB[i]->Caption="V"+IntToStr(i+1);
```

```
  LAB[i]->Show();
}
}

//---------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
float s,EPS=0.01,xx1,xx2,minD,maxD,max1,minDelta;
bool merkez;
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}
Table1->Open();
if(Table1->IsEmpty()){
  ShowMessage("Table is empty!");
  return;
}
min[0]=min[1]=MaxInt;
max[0]=max[1]=-MaxInt;
Button3Click(Button3);
EM=StrToFloat(Form4->Edit3->Text);
N=Table1->RecordCount;
Table1->First();
while(!Table1->Eof){
  xx1=Table1->FieldByName("X1")->AsFloat;
  xx2=Table1->FieldByName("X2")->AsFloat;
  if(xx1<min[0])min[0]=xx1;
  if(xx1>max[0])max[0]=xx1;
  if(xx2<min[1])min[1]=xx2;
  if(xx2>max[1])max[1]=xx2;
  Table1->Next();
}
Button7Click(Button7);
Table1->Open();
Table1->First();
int i=0;
while(!Table1->Eof){
  x[i].x1=Table1->FieldByName("X1")->AsFloat;
  x[i].x2=Table1->FieldByName("X2")->AsFloat;
  i++;
  Table1->Next();
}
Table1->Close();
bool end=true;
while(end){
  for(int i=0;i<K;i++){
```

```
    for(int k=0;k<N;k++){
      merkez=false;
      for(int i1=0;i1<K;i1++){
        if(Dist(v0[i1],x[k])<0.0001){
          for(int t=0;t<K;t++)if(t!=i1)u[t][k]=0;
          u[i1][k]=1;
          merkez=true;
        }
      }
      if(!merkez){
        s=0.0;
        for(int j=0;j<K;j++){

if(Dist(v0[j],x[k])>0.0001)s=s+pow(Dist(v0[i],x[k])/
                          Dist(v0[j],x[k]),2.0/(EM-1));
        }
        u[i][k]=1.0/s;
      }
    }
    s=0.0;
    for(int k=0;k<N;k++)
      s=s+pow(u[i][k],EM);
    v[i].x1=v[i].x2=0;
    for(int k=0;k<N;k++){
      v[i].x1=v[i].x1+x[k].x1*pow(u[i][k],EM);
      v[i].x2=v[i].x2+x[k].x2*pow(u[i][k],EM);
    }
    v[i].x1=v[i].x1/s;
    v[i].x2=v[i].x2/s;
  }
  end=false;
  for(int i=0;i<K;i++)
    if(Dist(v[i],v0[i])>EPS)end=true;
  for(int i=0;i<K;i++)v0[i]=v[i];
}
Table1->Open();
Table1->First();
int j=0;
while(!Table1->Eof){
  Table1->Edit();
  for(int i=0;i<K;i++)
    Table1->Fields->Fields[i+3]->AsFloat=u[i][j];
  Table1->Next();
  j++;
}
switch (Form4->RadioGroup1->ItemIndex){
  case 0:{
    JK=0;
```

```
    for(int i=0;i<K;i++)
      for(int j=0;j<N;j++)JK+=pow(u[i][j],EM);
    JK=JK/N;
    Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
    for(int i=0;i<K;i++){
      ED1[i]->Text=FloatToStrF(v[i].x1,ffFixed,10,4);
      if(Form4->RadioGroup3->ItemIndex==0)ED2[i]->Text="";
      else ED2[i]->Text=FloatToStrF(v[i].x2,ffFixed,10,4);
    }
    break;
  }
  case 1:{
    JK=0;
    for(int i=0;i<K;i++)
      for(int j=0;j<N;j++)JK-=u[i][j]*log(u[i][j]);
    JK=JK/N;
    Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
    for(int i=0;i<K;i++){
      ED1[i]->Text=FloatToStrF(v[i].x1,ffFixed,10,4);
      if(Form4->RadioGroup3->ItemIndex==0)ED2[i]->Text="";
      else ED2[i]->Text=FloatToStrF(v[i].x2,ffFixed,10,4);
    }
    break;
  }
  case 2:{
    int fi;
    JK=0.0;
    for(int i=0;i<N;i++){
      fi=0;
      for(int j=0;j<N;j++)if(ro(j)<=ro(i))fi++;
      JK+=1.0*fi/N;
    }
    JK=1.0*JK/N;
    Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
    for(int i=0;i<K;i++){
      ED1[i]->Text=FloatToStrF(v[i].x1,ffFixed,10,4);
      if(Form4->RadioGroup3->ItemIndex==0)ED2[i]->Text="";
      else ED2[i]->Text=FloatToStrF(v[i].x2,ffFixed,10,4);
    }
    break;
  }
  case 3:{
    maxD=-MaxInt;
    minDelta=MaxInt;
    for(int i=0;i<K;i++){
      minD=MaxInt;
      for(int i1=0;i1<K;i1++){
        if(i1==i)continue;
```

```
        if(minD>d(i,i1))minD=d(i,i1);
      }
      if(maxD<minD)maxD=minD;
      max1=-MaxInt;
      for(int i1=0;i1<N;i1++){
        for(int i2=i1+1;i2<N;i2++){
          float z;
          z=(u[i][i1]<u[i][i2])?u[i][i1]:u[i][i2];
         if (max1<z*Dist(x[i1],x[i2]))
                 max1=z*Dist(x[i1],x[i2]);
         }
      }
      if(minDelta>max1)minDelta=max1;
    }
    JK=maxD/minDelta;
    Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
    for(int i=0;i<K;i++){
      ED1[i]->Text=FloatToStrF(v[i].x1,ffFixed,10,4);
      if(Form4->RadioGroup3->ItemIndex==0)ED2[i]->Text="";
      else ED2[i]->Text=FloatToStrF(v[i].x2,ffFixed,10,4);
    }
    break;
  }
  case 4:{
    struct point MERKEZ;
    MERKEZ.x1=MERKEZ.x2=0;
    for(int i=0;i<N;i++){
      MERKEZ.x1+=x[i].x1; MERKEZ.x2+=x[i].x2;
    }
    MERKEZ.x1/=1.0*N; MERKEZ.x2/=1.0*N;
    JK=0.0;
    for(int i=0;i<K;i++)
      for(int
j=0;j<N;j++)JK+=pow(u[i][j],EM)*(pow(Dist(x[j],v[i]),2)-
pow(Dist(MERKEZ,v[i]),2));
      Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
    for(int i=0;i<K;i++){
      ED1[i]->Text=FloatToStrF(v[i].x1,ffFixed,10,4);
      if(Form4->RadioGroup3->ItemIndex==0)ED2[i]->Text="";
      else ED2[i]->Text=FloatToStrF(v[i].x2,ffFixed,10,4);
    }
    break;
  }
}
}

//---------------------------------------------------------
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}
Table1->Active=!Table1->Active;
}

//---------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}
Table1->Open();
Table1->First();
while(!Table1->Eof){
  Table1->Edit();
  for(int i=0;i<NCLUST;i++)
    Table1->Fields->Fields[i+3]->AsFloat=0.0;
  Table1->Next();
}
for(int i=0;i<NCLUST;i++){v[i].x1=0.0; v[i].x2=0.0;}
for(int i=0;i<NCLUST;i++)ED1[i]->Text="";
for(int i=0;i<NCLUST;i++)ED2[i]->Text="";
Label6->Caption="0.0";
Form2->Image1->Canvas->FillRect(TRect(0,0,Form2->Image1->
                          Width,Form2->Image1->Height));
}

//---------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
TCanvas *T=Form2->Image1->Canvas;
float x,y;
int x1,y1,x2;
bool move;

if(!OPENTABLE||!Table1->Active){
  ShowMessage("Table is not selected or not active!");
  return;
}
```

```
if(Form4->RadioGroup3->ItemIndex==1){
  TwoDimensional();
  return;
}
Form2->Show();
Form2->Image1->Canvas->FillRect(TRect(0,0,Form2->Image1->
                              Width,Form2->Image1->Height));
if(CheckBox1->Checked)ini1graphs();
T->Pen->Width=1;
T->Pen->Color=clBlack;
T->MoveTo(20,50);
T->LineTo(20,250);
T->LineTo(550,250);
T->Pen->Width=2;
Table1->IndexName="x1_idx";
Table1->Open();
Table1->First();
while(!Table1->Eof){
  x=Table1->FieldByName("X1")->AsFloat;
  x1=20+(x-min[0])/(max[0]-min[0])*(550-20);
  T->MoveTo(x1,247); T->LineTo(x1,253);
  Table1->Next();
}
T->TextOut(10,255,FloatToStrF(min[0],ffFixed,6,2));
T->TextOut(530,255,FloatToStrF(max[0],ffFixed,6,2));
T->Pen->Width=1;
for(int i=0;i<K;i++){
  Table1->First();
  move=true;
  T->Pen->Color=COL[i];
  while(!Table1->Eof){
    x=Table1->FieldByName("X1")->AsFloat;
    x1=20+(x-min[0])/(max[0]-min[0])*(550-20);
    y1=250-150*Table1->Fields->Fields[i+3]->AsFloat;
    if(move){T->MoveTo(x1,y1);move=false;} else T-
>LineTo(x1,y1);
    Table1->Next();
  }
}
for(int i=0;i<K;i++){
  x1=20+(v[i].x1-min[0])/(max[0]-min[0])*(550-20);
  T->TextOut(x1,80,FloatToStrF(v[i].x1,ffFixed,10,4));
}
}

//---------------------------------------------------------

void __fastcall TForm1::Button5Click(TObject *Sender)
```

```
{
Form1->Close();
}

//---------------------------------------------------------

void __fastcall TForm1::FormDestroy(TObject *Sender)
{
for(int i=0;i<NCLUST;i++){
  delete ED1[i];
  delete ED2[i];
  delete LAB[i];
  delete REC[i];
}
for(int j=0; j<=Ni;j++)delete ts[j];
Ni=-1;
Table1->Active=false;
}

//---------------------------------------------------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{
try{
  if(OpenDialog1->Execute()){
    Table1->Close();
    Table1->TableName=OpenDialog1->FileName;
    Label2->Caption=OpenDialog1->FileName;
    Table1->IndexName="x1_idx";
    Table1->Open();
    OPENTABLE=true;
  }
}
catch(...){
  Table1->Close();
  Table1->TableName="";
  Table1->IndexName="";
  OPENTABLE=false;
  for(int i=0;i<NCLUST;i++){v[i].x1=0.0; v[i].x2=0.0;}
  for(int i=0;i<NCLUST;i++)ED1[i]->Text="";
  for(int i=0;i<NCLUST;i++)ED2[i]->Text="";
  Label6->Caption="0.0";
  Form2->Image1->Canvas->FillRect(TRect(0,0,Form2->Image1->
                                Width,Form2->Image1->Height));
  ShowMessage("Invalid Table!");
  return;
}
}
```

```
//---------------------------------------------------------

void __fastcall TForm1::Shape1MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
if(ColorDialog1->Execute()){
    ((TShape *)Sender)->Brush->Color=ColorDialog1->Color;
    COL[((TShape *)Sender)->Tag]=ColorDialog1->Color;
}
}

//---------------------------------------------------------

void __fastcall TForm1::Button7Click(TObject *Sender)
{
double alfa=5.4,beta=5.4,delta=0.7,maxM,maxM0;
int iM,jM,K0,ii;
int u[NCLUST][NROWS],u1[NROWS];
bool flag;
struct {
  struct point x;
  float M;
} Mv[(KM0+1)*(KM0+1)];
struct point xMer;
if(!OPENTABLE||!Table1->Active){
  ShowMessage("Table is not selected or not active!");
  return;
}
Table1->Open();
if(Table1->IsEmpty()){
  ShowMessage("Table is empty!");
  return;
}
K=StrToInt(CSpinEdit1->Value);
N=Table1->RecordCount;
G1=KM0;
G2=KM0;
Table1->First();
ii=0;
while(!Table1->Eof){
  x[ii].x1=Table1->FieldByName("X1")->AsFloat;
  x[ii].x2=Table1->FieldByName("X2")->AsFloat;
  ii++;
  Table1->Next();
}
D[0]=(max[0]-min[0])/G1;
D[1]=(max[1]-min[1])/G2;
```

```
switch(Form4->RadioGroup2->ItemIndex){
case 0:{
  int KM=(G1+1)*(G2+1);
  for(int i=0;i<KM;i++){
    Mv[i].x.x1=min[0]+(i%(G1+1))*D[0];
    Mv[i].x.x2=min[1]+(i/(G1+1))*D[1];
  }
  K0=0;
  for(int i=0;i<KM;i++){
    Mv[i].M=0;
    for(int k=0;k<N;k++)Mv[i].M+=exp(-
alfa*Dist(x[k],Mv[i].x));
  }
  flag=true;
  do{
    maxM=-MaxInt;
    for(int i=0;i<KM;i++){
       if(maxM<Mv[i].M){maxM=Mv[i].M; iM=i;}
    }
    if(flag){maxM0=maxM; flag=false;}
    v0[K0]=vv[K0]=v[K0]=Mv[iM].x;
    K0++;
    for(int i=0;i<KM;i++){
      Mv[i].M=Mv[i].M-maxM*exp(-
beta*Dist(Mv[i].x,Mv[iM].x));
      Mv[i].M=(Mv[i].M<0)? 0 : Mv[i].M;
    }
  }while((K0<K)/*&&maxM/maxM0>=delta*/);
  break;
}
case 1:{

  break;
}
case 2:{
  double g,g1,gg,s1,s2,DD;
  int j0;
  g=N/K-1;
  s1=s2=0.0;
  for(int j=0;j<N;j++){
    s1+=x[j].x1;
    s2+=x[j].x2;
  }
  s1=s1/N; s2=s2/N;
  xMer.x1=s1; xMer.x2=s2;

//---------------------------------------------------------
```

```
  for(int i1=0;i1<K;i1++)
    for(int i2=0;i2<N;i2++){u[i1][i2]=0; u1[i2]=0;}
  int j1;
  for(int i=0;i<K;i++){
    DD=-MaxInt;
    for(int j=0;j<N;j++){
      if(!u1[j]){
        if(Dist(x[j],xMer)>DD){
          DD=Dist(x[j],xMer);
          j0=j;
        }
      }
    }
    xMer=x[j0];
    u[i][j0]=1; u1[j0]=1;
    gg=0;
    for(int j=0;j<N;j++){
      if(u1[j])continue;
      if(gg<=g){u[i][j]=1; u1[j]=1; gg++; continue;}
      DD=-MaxInt;
      j1=-1;
      for(int j2=0;j2<N;j2++){
        if(!u[i][j2])continue;
        if(DD<Dist(x[j2],xMer)){DD=Dist(x[j2],xMer);j1=j2;}
      }

if(Dist(x[j],xMer)<DD){u[i][j]=1;u[i][j1]=0;u1[j]=1;u1[j1]=
0;}
    }
    g1=0; s1=0.0; s2=0.0;
    for(int i3=0;i3<N;i3++){
      if(u[i][i3]){s1+=x[i3].x1; s2+=x[i3].x2; g1++;}
    }
    xMer.x1=s1/g1; xMer.x2=s2/g1;
    v[i]=xMer;
  }
  for(int i1=0;i1<N;i1++)
    if(!u1[i1]){
      DD=MaxInt;
      int i5=-1;
      for(int i3=0;i3<K;i3++){
        if(DD>Dist(x[i1],v[i3])){DD=Dist(x[i1],v[i3]);
i5=i3;}
      }
      u[i5][i1]=1; u1[i1]=1;
    }
  for(int i=0;i<K;i++){
```

```
    s1=s2=0.0;
    g1=0;
    for(int i3=0;i3<N;i3++){
      if(u[i][i3]){s1+=x[i3].x1; s2+=x[i3].x2; g1++;}
    }
    xMer.x1=s1/g1; xMer.x2=s2/g1;
    v0[i]=vv[i]=v[i]=xMer;
  }
  break;
}
}
//K=K0;
}

//---------------------------------------------------------

float Dist(struct point a, struct point b){
switch(Form4->RadioGroup3->ItemIndex){
case 0: return fabs(a.x1-b.x1);
case 1: return sqrt(pow(a.x1-b.x1,2)+pow(a.x2-b.x2,2));
case 2: return sqrt(pow(a.x1-b.x1,2)+pow(a.x2-b.x2,2));
}
}

//---------------------------------------------------------

void __fastcall TForm1::Button8Click(TObject *Sender)
{
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}
Form3->Show();
}

//---------------------------------------------------------

void __fastcall TForm1::Button9Click(TObject *Sender)
{
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}
if(MessageDlg("Do You want to empty the table?",
    mtWarning,
TMsgDlgButtons()<<mbYes<<mbCancel,0)!=mrYes)return;
Table1->Close();
Table1->EmptyTable();
```

```
for(int i=0;i<NCLUST;i++){v[i].x1=0.0; v[i].x2=0.0;}
for(int i=0;i<NCLUST;i++)ED1[i]->Text="";
for(int i=0;i<NCLUST;i++)ED2[i]->Text="";
Label6->Caption="0.0";
Form2->Image1->Canvas->FillRect(TRect(0,0,Form2->Image1-
>Width,Form2->Image1->Height));
for(int j=0; j<=Ni;j++)delete ts[j];
Ni=-1;
}

//---------------------------------------------------------

void __fastcall TForm1::Size1Click(TObject *Sender)
{
Form4->ShowModal();
}

//---------------------------------------------------------

void TwoDimensional(){
TCanvas *T=Form5->Canvas;
float x,y,z;
int x1,y1,x2,H0,W0;
Form5->Show();
Form5->Canvas->Brush->Color=clBtnFace;
Form5->Canvas->FillRect(TRect(0,0,Form2->Image1->Width,
                              Form2->Image1->Height));
H0=Form5->ClientHeight;
W0=Form5->ClientWidth;
T->Pen->Width=1;
T->Pen->Color=clBlack;
T->MoveTo(28,10);
T->LineTo(28,H0-53);
T->LineTo(W0-8,H0-53);
T->TextOut(30,H0-50,FloatToStrF(min[0],ffFixed,6,2));
T->TextOut(W0-40,H0-50,FloatToStrF(max[0],ffFixed,6,2));
T->TextOut(0,H0-65,FloatToStrF(min[1],ffFixed,6,2));
T->TextOut(0,0,FloatToStrF(max[1],ffFixed,6,2));
for(int i=0;i<K;i++){
  if(Form1->CheckBox1->Checked){
    x1=30+(vv[i].x1-min[0])/(max[0]-min[0])*(W0-30);
    y1=H0-55-(vv[i].x2-min[1])/(max[1]-min[1])*(H0-55);
    T->Pen->Width=2;
    T->Pen->Color=clBlack;
    T->Brush->Color=clBlack;
    T->Rectangle(x1-2,y1-2,x1+2,y1+2);
    T->Brush->Style=bsClear;
  }
```

```
  T->Pen->Width=1;
  T->Pen->Color=COL[i];
  x1=30+(v[i].x1-min[0])/(max[0]-min[0])*(W0-30);
  y1=H0-55-(v[i].x2-min[1])/(max[1]-min[1])*(H0-55);
  T->TextOut(x1,y1,FloatToStrF(v[i].x1,ffFixed,10,2));
  T->TextOut(x1,y1+12,FloatToStrF(v[i].x2,ffFixed,10,2));
  T->Pen->Width=2;
  T->Rectangle(x1-2,y1-2,x1+2,y1+2);
  Form1->Table1->First();
  while(!Form1->Table1->Eof){
    x=Form1->Table1->FieldByName("X1")->AsFloat;
    y=Form1->Table1->FieldByName("X2")->AsFloat;
    z=Form1->Table1->Fields->Fields[i+3]->AsFloat;
    x1=30+(x-min[0])/(max[0]-min[0])*(W0-30);
    y1=H0-55-(y-min[1])/(max[1]-min[1])*(H0-55);
    if(z>=Form5->TrackBar1->Position/10.0)T->Rectangle(x1-
1,y1-1,x1+1,y1+1);
    Form1->Table1->Next();
  }
}
}

//---------------------------------------------------------

void __fastcall TForm1::Button10Click(TObject *Sender)
{
float mJ;
int mJk;
if(!OPENTABLE){
  ShowMessage("Table is not selected!");
  return;
}

switch (Form4->RadioGroup1->ItemIndex){
  case 0:{
    mJ=-MaxInt;
    break;
  }
  case 1:{
    mJ=MaxInt;
    break;
  }
  case 2:{
    mJ=MaxInt;
    break;
  }
  case 3:{
    mJ=MaxInt;
```

```
    break;
  }
  case 4:{
    mJ=MaxInt;
    break;
  }
}
for(int k=2;k<=7;k++){
  CSpinEdit1->Value=k;
  CSpinEdit1->Refresh();
  Button1Click(Button1);
  Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
  Label6->Refresh();
  switch (Form4->RadioGroup1->ItemIndex){
    case 0:{
      if(mJ<JK){mJ=JK; mJk=k;}
      break;
    }
    case 1:{
      if(mJ>JK){mJ=JK; mJk=k;}
      break;
    }
    case 2:{
      if(mJ>JK){mJ=JK; mJk=k;}
      break;
    }
    case 3:{
      if(mJ>JK){mJ=JK; mJk=k;}
      break;
    }
    case 4:{
      if(mJ>JK){mJ=JK; mJk=k;}
      break;
    }
  }
}
CSpinEdit1->Value=mJk;
CSpinEdit1->Refresh();
Button1Click(Button1);
Label6->Caption=FloatToStrF(JK,ffFixed,10,4);
Label6->Refresh();
}

//---------------------------------------------------------

void __fastcall TForm1::About2Click(TObject *Sender)
{
AboutBox->Label1->Visible=false;
```

```
AboutBox->Label3->Visible=true;
AboutBox->ShowModal();
}

//---------------------------------------------------------

void __fastcall TForm1::Button8MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
if(!(Shift.Contains(ssAlt)&&Shift.Contains(ssShift)))return
;
OKBottomDlg->ShowModal();
if(!dostup)return;
OKBottomDlg->Edit1->Text="";
dostup=false;
AboutBox->Label3->Visible=false;
AboutBox->Label1->Visible=true;
AboutBox->ShowModal();
}

//---------------------------------------------------------

void ini1graphs(){
TCanvas *T=Form2->Image1->Canvas;
float vi[NCLUST];
int x1,x2;
T->Pen->Width=2;
T->Pen->Color=clBlack;
for(int i=0;i<K;i++)vi[i]=vv[i].x1;
for(int i=1;i<K;i++){
  for(int j=0;j<i;j++)
    if(vi[j]>vi[i]){float t=vi[i]; vi[i]=vi[j]; vi[j]=t;}
}
if(Form1->CheckBox1->Checked){
  for(int i=1;i<K;i++){
    x1=20+(vi[i-1]-min[0])/(max[0]-min[0])*(550-20);
    x2=20+(vi[i]-min[0])/(max[0]-min[0])*(550-20);
    if(i==1){T->MoveTo(20,100); T->LineTo(x1,100);}
    if(i==K-1){T->MoveTo(x2,100); T->LineTo(550,100);}
    T->MoveTo(x1,100); T->LineTo(x2,250);
    T->MoveTo(x1,250); T->LineTo(x2,100);
  }
}
}

//---------------------------------------------------------

float d(int i, int j){
```

```
float s,s1,z;
s=s1=0.0;
for(int i1=0;i1<N;i1++){
  for(int i2=0;i2<N;i2++){
    z=(u[i][i1]<u[j][i2])?u[i][i1]:u[j][i2];
    s+=z*Dist(x[i1],x[i2]);
    s1+=z;
  }
}
return s/s1;
}

//-----------------------------------------------------

void CalcM(){
int A[6][6],B[6][6];
int KM=(G1+1)*(G2+1);
for(int i=0;i<KM;i++)
  for(int j=0;j<KM;j++){
    A[i][j]=min[0]+(i%(G1+1))*D[0];
    A[i][j]=min[1]+(i/(G1+1))*D[1];
  }
}

//-----------------------------------------------------

float ro(int j){
float min,max;
min=MaxInt; max=-MaxInt;
for(int i=0;i<K;i++){
  if(min>u[i][j])min=u[i][j];
  if(max<u[i][j])max=u[i][j];
}
return 1.0*min/max;
}
```

ÖZGEÇMİŞ

29.08.1979 tarihinde İzmir'de dünyaya gelen Gözde Ulutagay, orta ve lise öğrenimini İzmir Özel Türk Lisesi'nde tamamlamıştır. 1997 yılında Ege Üniversitesi Fen Fakültesi İstatistik Bölümü'ne girmeye hak kazanmış ve 2001 yılında mezun olmuştur. Eylül 2001'de Ege Üniversitesi Fen Bilimleri Enstitüsü İstatistik Bilim Dalında Yüksek Lisans öğrenimine başlamıştır. Aralık 2001'den bu yana aynı bölümde Araştırma Görevlisi olarak görev yapmakta olan Gözde Ulutagay, evli ve 1,5 yaşında bir kız çocuğu annesidir.