Erciyes University Journal of the Institute of Science and Technology
Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi
ISSN 1012-2354
Cilt (Volume): 30, Sayı (Issue): 1, Ocak/January-2014
http://fbe.erciyes.edu.tr/

Bölümleyici kümeleme algoritmalarının farklı veri yoğunluklarında karşılaştırılması

***Hüssein Ridha Ali ALZAND[1], Hacer KARACAN[2]**
[1]*Gazi Üniversitesi, Bilişim Enstitüsü, Bilgisayar Bilimleri Anabilim Dalı*
[2]*Gazi Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü*

Anahtar Kelimeler: kümeleme algoritmaları, kümeleme analizi.

ÖZET

Teknolojinin yaygın kullanılmasının neticesinde hacmi her geçen gün artan büyük veri yığınları ortaya çıkmaya başlamıştır. Bu kadar büyük boyutta verinin analizi ve içindeki herhangi bir bilgiye ulaşmak basit inceleme yöntemleriyle oldukça zor olduğundan veri madenciliği devreye girmiştir. Veri madenciliği, çok büyük veri tabanlarından, önceden bilinemeyen, geçerli ve kullanılabilir bilginin çıkarılma işlemi olarak ifade edilmektedir. Başka bir deyişle veri madenciliği, çok büyük veri tabanlarındaki ya da veri ambarlarındaki veriler arasında bulunan ilişkiler, örüntüler, değişiklikler, sapma ve eğilimler, belirli yapılar gibi ilginç bilgilerin ortaya çıkarılması işlemidir. Veri madenciliği alanında son zamanlarda yaygın bir şekilde kullanılan yöntemlerden biri kümeleme yöntemidir. Kümeleme, veri kümesindeki bilgileri farklı kümelere ayırarak küme içindeki verilerin özelliklerinin benzerlik oranı minimum ve kümeler arasında benzerlik oranını maksimum yapmaktadır. Bu çalışmada bölümleyici kümeleme yöntemleri ele alınarak farklı dağılımlı veri setleri üzerinde bölümleyici kümeleme algoritmalarının karşılaştırması gerçekleştirilmiştir. Bölümleyici kümeleme algoritmaları arasından "k-ortalama" ve "çekirdek k-ortalama" algoritmaları seçilmiştir. Farklı dağılımlı veri setlerini kümeleyerek iki algoritmanın hızı, kümeleme kalitesi ve bellek kaplaması açısından bilgiler elde edilmiş ve bu bilgiler ışığında iki algoritmanın karşılaştırma sonuçları sunulmuştur.

Comparison of partitioning-based clustering algorithms on differently distributed data

Key Words: clustering algorithms, clustering analysis.

ABSTRACT

As a result of widespread use of technology, large volumes of collected data began to emerge. It is impossible to discover and analyze any information in large data like this, so in this case data mining comes into play. Data mining is a process that discovers unpredictable and usable knowledge from databases. In other words, data mining is defined as the process of finding relation patterns, changes, deviations and trends, as well as interesting information specific structures from large databases. One of the widely used data mining methods is a method of clustering. Clustering divides the data set into different clusters, and it tries to make the likelihood ratio as minimum inside the cluster and as maximum among other clusters depending on the options in the database. In this study, partitioning-based clustering methods are discussed by applying them on data sets with different distribution patterns. We used "k-means" and "kernel k-means" partitioning algorithms for clustering data sets. By applying clustering operations on differently distributed data sets, we compared the speed, clustering quality and the size of memory usage for these algorithms. The information that we gathered by this comparison is presented and discussed in the related sections of this paper.

*Sorumlu Yazar (Corresponding author) e-posta: husseinalzand@yahoo.com

1. Giriş

Her geçen gün, teknolojide büyük gelişmeler yaşanmaktadır. Bu gelişmelerle birlikte veri boyutları da aynı oranda artmaktadır. Yüksek kapasiteli işlem yapma gücünün ucuzlamasının bir sonucu olarak, veri saklama da kolaylaşmıştır. Veri tabanlarında saklanan veri, bir arşive benzetilirse, bu veri arşivi tek başına değersizdir ve kullanıcı için çok fazla bir anlam ifade etmez. Ancak bu veri arşivi, belirli bir amaç doğrultusunda sistematik olarak işlenir ve analiz edilirse, değersiz görülen veri yığınında, amaca yönelik sorulara cevap verebilecek çok değerli bilgilere ulaşılabilir [1].

Veri madenciliği verilerin içindeki bilgilerin, ilişkilerin, değişikliklerin, hataların ve istatistiksel olarak önemli olan bilgi ve değerlerin otomatik olarak keşfedilmesidir [2]. Büyük miktardaki veri kümesinde bulunan nesnelerin keşfedilmesinde ve bu verilerle ilgili tahminler yapılmasında kullanılabilecek ilişkilerin çıkarılması olarak tanımlanabilir [3].

Veri Madenciliğinin büyük bir dalını temsil eden ve son zamanlarda çok ilgi çeken belirleyici yöntemlerden biri kümeleme yöntemidir. Bu yöntemle veri tabanındaki nesneler genel olarak belli sayıda kümelere ayrılmaktadır. Aynı özelliği taşıyan nesneler aynı grupta olurken birbirinden farkı çok olan nesnelerin farklı gruplarda yer almaları sağlanmaktadır. Yöntemle hedeflenen gruplar arası farkı maksimuma yükseltirken gruplar içi farkın minimum olmasıdır. Kümeleme verileri gruplandırarak birkaç homojen gruba dağıtılmasını amaçlayan bir veri analiz aracıdır. Üstünde çalışılan fenomeni anlamak veya yorumlamak için çok sayıda bilim dalında verilerin kümelenmesine ihtiyaç duyulmaktadır [4].

Kümeleme algoritmalarından olan merkez esaslı k-ortalama algoritması bu alanda en yaygın kullanıma sahiptir. K-ortalama algoritması nesnelerin merkezlere uzaklıklarını hesaplayarak grupların belirlenmesini sağlar. K-ortalama algoritmasının yaygın kullanımı nedeni ile araştırmacılar bu algoritmanın geliştirmesine ve daha iyi performansla çalışmasına yönlenmişlerdir. Algoritmanın küme kalitesinin artırılması açısından yapılan çok sayıda çalışma mevcuttur. Mat Isa ve arkadaşları [5] k-ortalama algoritmasının performansının artırması için nesnelerin optimum merkezinin bulunması için belirsiz k-ortalama algoritması geliştirmiştir. Standart k-ortalama algoritması ile bulanık k-ortalama algoritması arasında bir karşılaştırma yapılmış ve bulanık yöntemle nesnelerin daha başarılı ve etkili şekilde merkezleri bulduğu ifade edilmiştir [5]. Wang, gürültülü veri tabanının normal halindeyken ve gürültülü nesneleri çıkararak k-ortalama algoritmasını uygulamış ve karşılaştırma yapmıştır [6]. Gerçekleştirilmiş olan bu çalışmada önce gürültülü noktalar kümeden çıkarılmış ve sonrasında küme merkezleri hesaplanarak k-ortalama algoritmasının daha etkili ve kaliteli sonuç verecek şekilde çalışması sağlanmıştır [6]. Diğer bir çalışmada ise k-ortalama algoritmasında sadece Öklid formülünün kullanılması yetersiz görülerek performans arttırmak amacıyla her nesneye bir değişim katsayısı eklenerek hesaplama gerçekleştirilmiştir [7]. Li ve arkadaşları ise 2009 yılında gerçekleştirdikleri çalışmada k-ortalama algoritmasını büyük hacimli bir veri tabanında uygulanması ve aynı veri tabanında parçalara bölerek uygulanması arasında kalite açısından bir karşılaştırma üzerinde standart yöntemle sıkıntılı çalıştığı için çok büyük olan veri setlerini küçük parçalı hücrelere bölerek ızgaralı bir yapı haline dönüştürmüş ve kümeleme işlemini bu veri üzerinde gerçekleştirmişlerdir. Bu yöntemde her hücrenin kendi merkez ağırlığı olduğundan veri kontrolünün kolaylaştığı gösterilmiştir [8]. k-ortalama küme kalitesini [9] ve performansını [10,12,13] artırmak için yukarıda anılan çalışmalara ek olarak çeşitli farklı yöntemler geliştirilmiş ve kümenin ölçümünü optimize ederek en uygun küme merkezi bulmayı hedefleyen çalışmalar gerçekleştirilmiştir. Bunun yanı sıra bizim çalışmamızda olduğu gibi k-ortalama algoritmasını literatürde mevcut diğer algoritmalarla karşılaştıran uygulamalar da mevcuttur [11,14].

Bu çalışmada veri madenciliği kümeleme algoritmalarından k-ortalama algoritması ele alınıp çalışma yöntemi anlatılmış ve "çekirdek" (kernel) yöntemini kullanan çekirdek k-ortalama algoritmasının nasıl kümeleme yaptığı ve hangi yöntemleri kullandığı açıklanmıştır. Ayrıca bu iki algoritmayla ilgili bir karşılaştırma programı gerçeklenerek bu iki algoritmanın farklı veri setleri üzerinde uygulanmasıyla elde edilen karşılaştırmalı sonuçlar sunulmuştur. Karşılaştırılan sonuçlarda çekirdek k-ortalama algoritmasının standart k-ortalama algoritmasından daha kaliteli kümeleme yaptığı gösterilmektedir.

2. Bölümleyici Kümeleme Algoritmaları

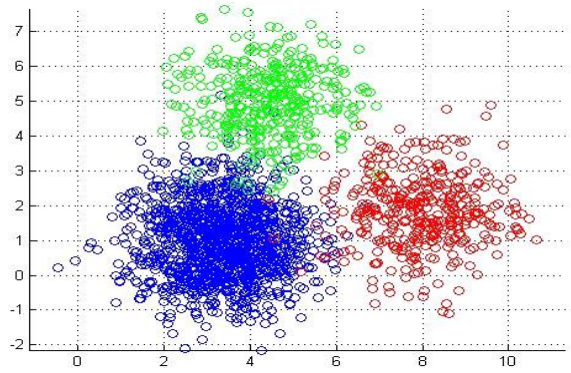
Veri Madenciliği büyük hacimde olan ve genellikle denetimsiz kalan veri setlerini makul ve kullanılır bir hale getirmek amacıyla kullanılmaktadır. Veri Madenciliği veri kümesini makul bir hale getirmektense veri kümesinin içinden bilgileri keşfedip anlaşılan, kullanılır, geçerli ve faydalı hale getirmeyi hedefler. Bu sebeple "Veri Tabanlarında Bilginin Keşfi" (Knowledge Discovery in Database-KDD) [15], "Bilgi çıkarımı" (Knowledge Extraction), "Veri ve Örüntü Analizi" (Data/PatternAnaysis), "Veri Arkeolojisi" (Data Archeology), "Veri Eşeleme" (Data Dredging) [16] gibi isimlendirmeleri de mevcuttur.

Veri madenciliği algoritma türlerinden biri olan bölümleyici kümeleme algoritmaları veri kümesinden n tane giriş parametre kabul eden ve bu parametreleri k tane kümeye bölen algoritmalardır. Bu tekniği kullanan tüm algoritmalar merkez noktaların kümeyi temsil etmesi esasına dayanır. Bölümleyici algoritmaların kolay uygulanabilirliği ve verimli sonuçlar vermesi nedeniyle yaygın bir şekilde kullanılmaktadır. Bu algoritmalar arasında yaygın olarak kullanılanlar k-ortalama, k-medoid, clara ve clarans, c-means algoritmalarıdır. Bu çalışmada karşılaştırması yapılan k-ortalama ve çekirdek k-ortalama algoritmaları aşağıda detaylı olarak açıklanmış ve devam eden bölümlerde karşılaştırma sonuçları sunulmuştur.

2.1. K-Ortalama Algoritması

K-ortalama algoritması, veritabanındaki n tane nesnenin k adet kümeye bölümlenmesini sağlar. Kümeleme sonucu küme içi elamanlar arasındaki benzerlikler çok iken, kümeler arası (inter-cluster) elamanları arasındaki benzerlikler çok düşüktür. K-ortalama algoritması iki ana parçadan oluşmaktadır birinci parça k merkez değerlerini rastgele belirler, belirlenen k merkez değeri girdi olarak sunulmuş olmalıdır. Diğer parça ise veri kümesindeki her nesnenin belirlenen k merkez

gerçekleştirmişlerdir [8]. k-ortalama algoritması çok büyük hacimdeki veri kümeleri en az ise o merkezin kümesine dâhil olur [17]. Daha sonra, her bir kümenin ortalama değeri hesaplanarak yeni küme merkezleri belirlenir ve tekrar nesne-merkez uzaklıkları incelenir. Şekil- değerlerine uzaklıkları ve değerini kıyaslar. Ortaya çıkan her nesnenin her k merkezine uzaklık değeri göze alınarak bu nesnelerin hangi kümelere ait olduğu belirlenir, nesnenin hangi merkeze uzaklığı en az ise o merkezin kümesine dâhil olur [17]. Daha sonra, her bir kümenin ortalama değeri hesaplanarak yeni küme merkezleri belirlenir ve tekrar nesne-merkez uzaklıkları incelenir. Şekil-1'de k-Ortalama algoritmasının örnekleri guruplara ayırması gösterilmektedir.1'de k-ortalama algoritmasının örnekleri guruplara ayırması gösterilmektedir.



Şekil 1. k-ortalama algoritması ile guruplara ayrılmış veri kümesi nesneleri K=3 [18]

Nesneler özeliklerine göre koordinat uzayında temsil edilir ve koordinatın her vektörü nesnenin bir özelliğini temsil ed
m
iş
Nesne koordinatı ve merkez koordinatı arasındaki mesafeyi kıyaslamak için Öklid uzaklık fonksiyonu kullanılır [19]:

$$d(x_i,y_i) = [\sum_{i=1}^{n}(x_i - y_i)^2]^{1/2}$$

D(Xi,Yi) : bulunması istenen iki nokta arasındaki mesafeyi temsil etmektedir. $\sum_{i=1}^{n}(x_i - y_i)^2$: her koordinatın x ve y eksen değerlerinin toplam farklarının karesini temsil etmektedir. Dikkate alınması gereken şey koordinatla tüm merkezler arasında bu uzaklık formülü uygulanması ve en kısa mesafeye sahip olan merkezin nesneyi kendi gurubuna almış olmasıdır. Nesnelerin kümelemesinde değişiklik olmayana kadar bu işlem tekrarlanır [20].

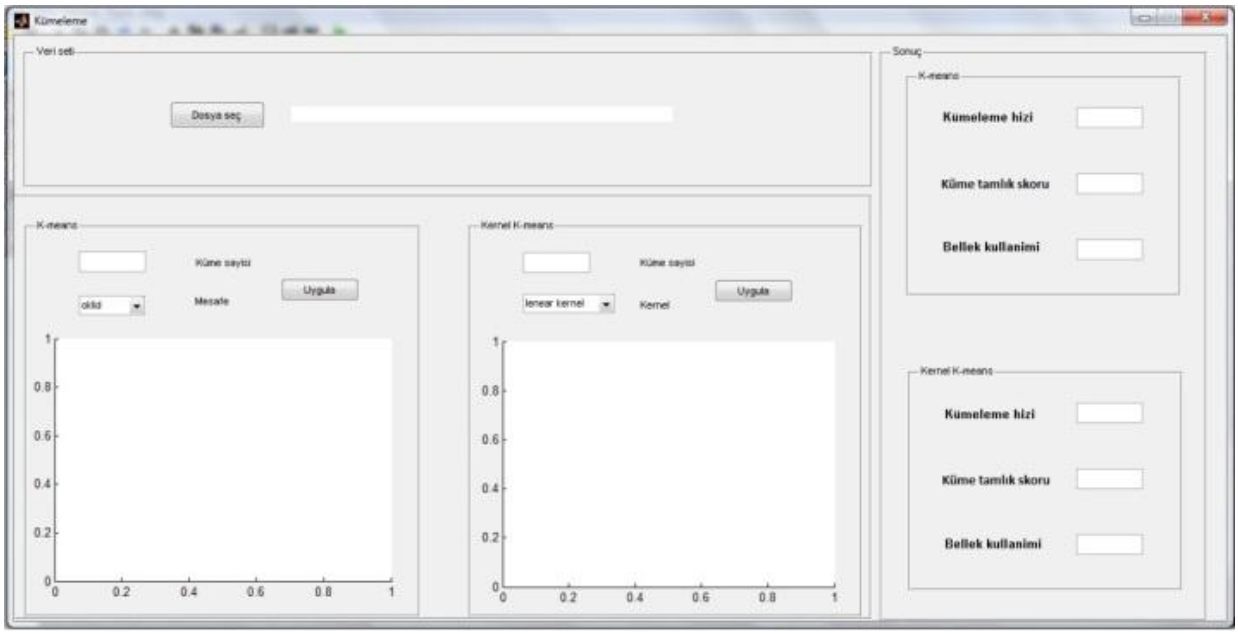
2.2. Çekirdek -ortalama Algoritması

Çekirdek esaslı kümeleme standart kümelemeden farklı uzaylarda çalışır. Veri kümesindeki nesne vektörlerini alıp lineer olmayan fonksiyonları kullanarak vektörleri çok boyutlu uzaya taşır ve kümeleme işlemlerini o boyut ortamında yapar. Vektörlerin yüksek uzaylara taşınıp kümelenmesi daha kolay, daha etkili ve kaliteli kümeler elde edilmesini sağlamaktadır [21].

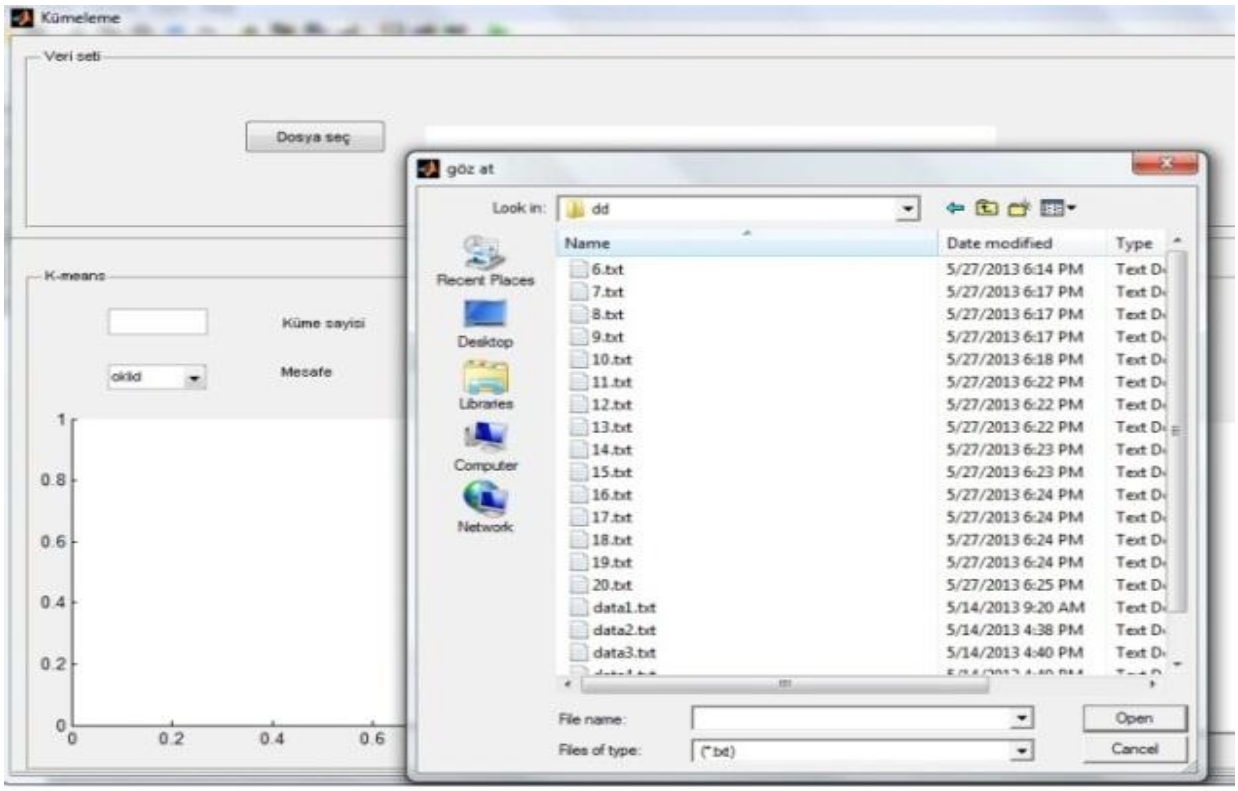
Çekirdek k-ortalama algoritması standart k-ortalama kümeleme algoritmasının lineer olmayan uzantısıdır. Bu metot tekrarlamalı bir metottur. Başlangıçta veri noktalarının lineer olmayan dönüşümlerini $\phi(.)$ kullanarak giriş uzayından çok boyutlu uzaya dönüştürür ve sonra işlemi gerçekleştirerek kümeleme hata oranını minimize etmeye çalışır. Başka bir deyişle, çekirdek k-ortalama algoritması, standart k-ortalama algoritmasının işini çok boyutlu uzayda gerçekleştirir. Veri kümesindeki x ve y noktalarının çok boyutlu uzaydaki $\phi(x).\phi(y)$ dönüşümünün hesaplanması için çekirdek fonksiyonu adı verilen K(x,y) fonksiyonu kullanılır
(K: DXD → R). Çekirdek k-ortalama algoritmasının tekrarlamalı işlevinde algoritmanın K(xi,xj) değerinin hesaplanması gerçekleştirilir. Tekrarlamalı adımlar başlamadan önce nesne girişi Kij=K(xi,xj) olacak şekilde çekirdek matrisi (H = [Kij]nxn) hazırlanır. Ayrıca, küme sayısının (k) belirlenmesi ve başlangıç noktaların belirlenmesi algoritmanın gereksinimlerindendir [22].

3. Uygulama

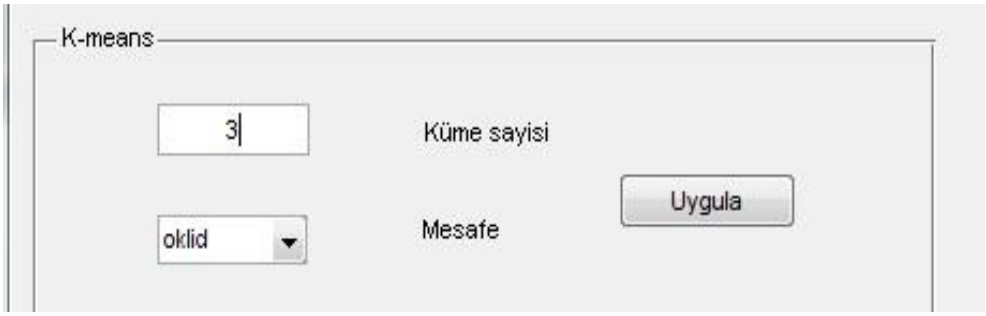
Geliştirilen uygulama standart k-ortalama ve çekirdek k-ortalama algoritmalarının farklı veri setleri üzerinde uygulanmasını sağlamaktadır. Uygulamada veri kümesi veri ekranından seçilir ve grup sayısı girdi olarak alınır. Veri kümesi ve grup sayısı girildikten sonra uygulama butonuna basılmasıyla algoritma çalışmaya başlar ve sonuçlar elde edilir. Uygulamanın ara yüzü Şekil-2'de gösterilmektedir.
Uygulama MATLAB 7.7.0 ve Python 2.7.3 sürümünde geliştirilmiştir. Veri kümesi uygulamanın en üst parçasında bulunan veri kümesi kısmından yüklenmektedir. Veri kümesini programa yüklemek için "dosya seç" tuşuna tıklandığında dosya seçim penceresi (Bkz.Şekil-3) açılır ve buradan istenen veri kümesi eklenir. Seçilen dosyayı algoritmaları kullanarak guruplara ayırmak için uygulamanın aşağıdaki kısmında biri standart k-ortalama ile ilgili diğeri çekirdek k-ortalama ile ilgili olan iki parça bulunmaktadır. Her iki algoritmayı da çalıştırmak için küme sayısı girmek gerekmektedir ve bu işlem için müstakil kutular bulunmaktadır.
Küme sayısı girişi yapıldıktan sonra her algoritma ile ilgili uygulama butonu tıklanarak algoritma çalıştırılır ve veri kümesi guruplara ayrılır. Şekil-4 küme girişi ve uygulama butonunu göstermektedir.
Uygulama, algoritmaları çalıştırarak veri kümesini kümelere ayırır ve kümelerin görüntüsünü uygulamanın aşağı kısmında bulunan çizim kutusuna yerleştirir. Kümeleri göstermek için her küme nesnelerinin vektörünü farklı renklerle çizmektedir. Örnek olarak veri kümesinin kümelere ayrılmış halinin çıkış çizimi Şekil-5'te gösterilmektedir. Ayrıca, kümeleme işlemi tamamlandıktan sonra kümelerin çiziminin yanında uygulama iki algoritmanın performansı hesaplanmaktadır.
Uygulama, kümeleri hesaplarken algoritmaların işlemi tamamlama hızını, bellek kullanımını ve kümeleme işleminin tamlık değerini hesaplamaktadır. Tamlık değeri kümeleme algoritmasının gerçek kümelere ne kadar yakın küme yaptığını yansıtmaktadır. Tamlık değerinin hesaplanma yöntemi olarak her algoritmanın kümeleme sonuç çıktısının alınarak veri kümesinin gerçek kümeleri ile karşılaştırılması esasına



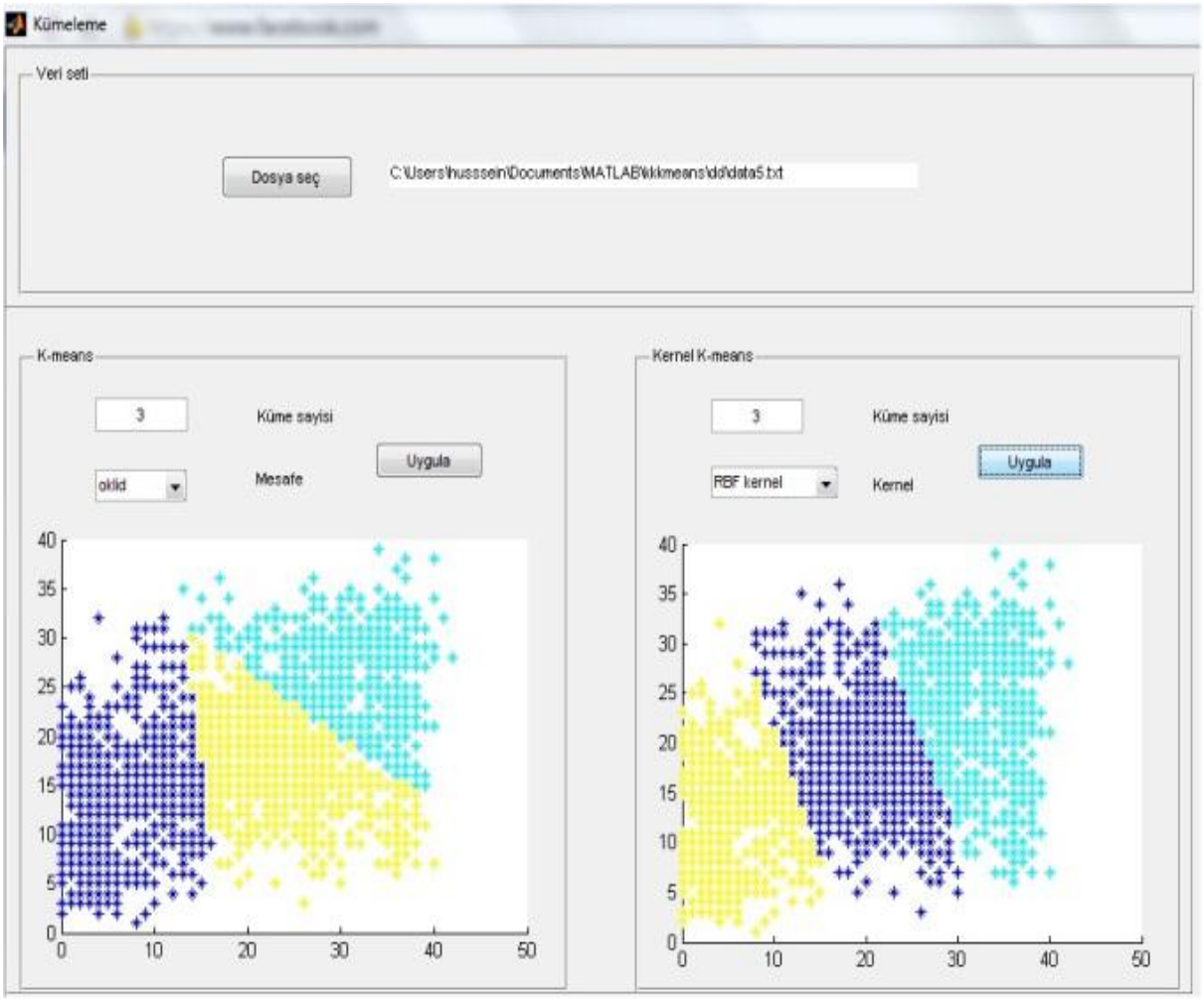
Şekil 2. Uygulama Arayüzü



Şekil 3. Dosya Seçme Ekranı



Şekil 4. Küme sayısı kutusu ve uygulama tuşu



Şekil 5. Çizim Ekranı

dayanır. Burada gerçek kümeye benzerliği açısından, 0.0 ile 1.0 arasında değişen oranlı bir değer elde edilmektedir. Tamlık skoru değeri 1.0 değerine ne kadar yakın ise o algoritmanın kümeleme işleminin o kadar başarılı sonuçlar vermekte olduğu anlaşılmaktadır [24].

Tablo 1. Farklı veri kümeleriyle elde edilen kümeleme hızları.

Veri kümesi boyutu	Veri dağılımı	K-ortalama algoritması Kümeleme hızı (Saniye)	Çekirdek k-ortalama algoritması Kümeleme hızı (Saniye)
Büyük	Kitleli	4.8	8.1
Büyük	Düzgün	5.14	9.51
Orta	Kitleli	1.64	1.43
Orta	Düzgün	1.88	2.80
Küçük	Kitleli	0.17	0.65
Küçük	düzgün	0.73	0.78

Tablo 2. Farklı veri kümeleriyle elde edilen küme tamlık skorları

Veri kümesi boyutu	Veri dağılımı	K-ortalama algoritması Küme tamlık skoru	Çekirdek k-ortalama algoritması Küme tamlık skoru
Büyük	Kitleli	0.952	0.973
Büyük	Düzgün	0.763	0.797
Orta	Kitleli	0.831	0.911
Orta	Düzgün	0.327	0.361
Küçük	Kitleli	0.659	0.662
Küçük	düzgün	0.221	0.238

Tablo 3. Farklı veri kümeleriyle elde edilen bellek kaplama miktarları

veri kümesi boyutu	Veri dağılımı	K-ortalama algoritması Bellek kaplama miktarı (Byte)	Çekirdek k-ortalama algoritması bellek kaplama miktarı (Byte)
Büyük	Kitleli	325676	200365016
Büyük	Düzgün	325676	200365016
Orta	Kitleli	91884	14704136
Orta	Düzgün	314646	17943935
Küçük	Kitleli	21938	405799
Küçük	düzgün	167470	4081686

Uygulamada kitleli ve düzgün dağılımlı olmak üzere iki farklı türde dağılım gösteren ve küçük (300-800 nesne), orta (1500-3000 nesne), büyük (5000-7500 nesne) olarak üç farklı veri yoğunluğuna sahip çok boyutlu veri setleri kullanılmıştır. Tablolarda (Tablo 1-3) uygulamanın her iki algoritma için farklı özellikteki bu veri setlerine uygulanmasıyla elde edilen sonuçlar daha önce açıklanan kümeleme hızı, tamlık skoru ve küme bellek kaplama miktarı özellikleri açısından sunulmaktadır.

4. Bulgular ve Tartışma

Söz konusu iki algoritma için yapılan karşılaştırmada kümeleme hızı 0.17 ile 9.51 saniye arasında değişmektedir. Kümelemenin hızı veri kümesinin hacmine göre değişiklik göstermiştir. Bekleneceği üzere veri kümesinin hacmi büyüdükçe kümeleme süresinde artış olmaktadır. Standart k-ortalama algoritmasının, çekirdek k-ortalama algoritmasından daha hızlı çalıştığı görülmüştür. Veri setlerinin dağılım şekline göre inceleme yapıldığında kitleli yapıya sahip (düzgün dağılım gerçekleştirmemiş olan) veri setlerinde uygulamanın çok daha az zaman harcadığı ortaya çıkmıştır. İki algoritma arasındaki zaman harcamasına bakıldığında çekirdek k-ortalama algoritmasının standart k-ortalama algoritmasında daha iyi performans gösterdiği tek durum orta boyutlu kitleli dağılıma sahip veri setleriyle işlem yapıldığı durumdur.

Diğer bir karşılaştırma kriteri küme tamlama skorudur. Tamlama skorunda iki algoritmanın karşılaştırılması sonucu çekirdek k-ortalama algoritmasıyla standart k-ortalama algoritmasına oranla gerçek kümelere daha yakın ve gerçekçi sonuçlar elde edilmiştir. Ayrıca, veri kümesinin boyutu artıkça algoritmaların sonuç kalitelerinin arttığı görülmektedir. çekirdek k-ortalama algoritmasının küme tamlama üstünlüğü tüm veri kümesi türleri için korunmuştur.

Kümeleme bellek kaplamasına bakıldığında yine veri kümesinin hacminin etkili olduğu görülmektedir. Veri kümesinin hacmi yükseldikçe kullanılan bellek miktarında yükseliş oluşmaktadır. Çekirdek k-ortalama algoritmasının kullandığı bellek miktarı, tüm veri kümesi türleri için, k-ortalama algoritmasına göre daha fazla olarak ölçülmüştür. Buradaki ilgi çekici çıkarım kullanılan bellek miktarının kitleli dağılıma sahip veri setleri için düzgün dağılımlı olanlara göre çok daha az oluşudur. Bunun en çarpıcı örneği, standart k-ortalama algoritmasının orta boyutlu kitleli dağılıma sahip veri kümesine uygulandığı durumda, nesne sayısı artışı olmasına rağmen, küçük boyutlu düzgün dağılıma sahip veri kümesine uygulandığı durumdan çok daha az bellek kullanımı gerçekleşmiş olması olarak gösterilebilir. Çekirdek k-ortalama algoritmasının yüksek başarım oranının ve bellek gereksinim artışının algoritmanın lineer olmayan yapısından kaynaklandığı düşünülmektedir.

Yukarıda yapılan karşılaştırma k-ortalama algoritması ve çekirdek k-ortalama algoritmaları arasındaki yükselen kümeleme algoritmalarının versiyonlarının daha iyi performansla çalıştığını göstermeyi hedeflemektedir. K-ortalama algoritmasının gelişmiş seviyesini temsil eden çekirdek esaslı çekirdek k-ortalama algoritmasının kümeleme işlemini yaparken kümeleme tamlığının standart k-ortalama algoritmasından daha üstün olması beklentisinin çalışma sonuçlarıyla desteklendiği tablolarda küme tamlık skoru değerlerini karşılaştırırken net bir şekilde görülmektedir. Çekirdek k-ortalama algoritmasına uygulanmış olan küçük, orta ve büyük olmak üzere farklı hacim yapılarına sahip veri setlerinin ve kitleli dağılımlı ve düz dağılımlı olmak üzere farklı dağılım hallerinin sonucunda beklenen performansın elde edildiği görülmüştür. Beklenen başarılı performans standart k-ortalama algoritmasından daha kaliteli kümeleme elde edilmesidir.

Çekirdek k-ortalama algoritmasının daha kaliteli kümeleme yaptığı veri kümesi hacmi ve boyutları büyüdükçe daha net bir şekilde görülmektedir.

Kaynaklar

1. Keselj, V. and Liu , H., 2007. Combined mining of Web server logs and web contents for classifying user navigation patterns and predicting users future requests, ScinceDirect digital library,61(2),304-330.
2. Dujovne, E., Huillier, G. and Vela´squez, D., 2007. Extracting significant Website Key Objects: A Semantic Web mining approach, ScinceDirect digital library, 24(8), 1532-1541.
3. Internet: Istanbul Üniversitesi "Veri Tabanlarında Bilgi Keşfi ve Veri Madenciliği", 13.12.2012. http://www.istanbul.edu.tr/isletme/dergi/nisan2000/1.HTM.
4. Bouveyron, C. and Brunet-Saumard, C., 2012. Model-based clustering of high-dimensional data: A review, Elsevier.
5. Isa, N.A.M, and Noraini Sulaiman, S., 2010. Adaptive Fuzzy- Clustering Algorithm for Image Segmentation, IEEE digital library, 56(4), 2661 – 2668.
6. Su, X. and Wang, J., 2011. An improved k-ortalama clustering algorithm, IEEE digital library, 12229842(978-1-61284-485-5), 44-46.
7. Fan, A. and Ren, S., 2011. k-ortalama Clustering Algorithm Based On Coefficient Of Variation, IEEE digital library, 12439403(978-1-4244-9304-3), 2076 – 2079.
8. Chen, J., Li, D. and Shen, H., 2009. A Fast k-ortalama Clustering Algorithm Based on Grid Data Reduction, IEEE digital library, 9980042(1095-323X), 1 – 6.
9. Eswara Reddy, B., Viswanath , P. and Hitendra Sarma, T., 2012. A hybrid approach to speed-up the k-ortalama clustering method, Springer-Verlag , 4(2), 107-117.
10. Chang, D. and Xian, W., 2009. A genetic algorithm with gene rearrangement for k-ortalama clustering. Pattern Recognition, IEEE digital library, 42(7), 1210-1222.
11. Bagirov, A.M., Ugon, J. and Webb, D., 2011. Fast modified global k-ortalama algorithm for incremental cluster construction, Pattern Recognition, SinceDirect,36(2), 451-461.
12. Binti, W., Herawan, T., Maseri, W., Mohd, A.H. and K.F.Rabbi, 2011. An Improved Parameter less Data Clustering Technique based on Maximum Distance of Data and Lioyd k-ortalama Algorithm, SinceDirect, 1, 367-371.
13. Jana Prasanta and K.,Reddy, D., 2012. Initialization for k-ortalama clustering using Voronoi diagram, Elsevier Ltd., 4, 395–400.
14. Brunsch, T., Röglin, H., 2012. A bad instance for k-ortalama++, in press, Elsevier.
15. Mozafari, B., Thakkar, H. and Zaniolo, C., 2008. A Data Stream Mining System, IEEE digital library, 978-0-7695-3503-6 (10453400), 987 – 990.
15. Kaya, H. ve Köymen, K., 2008. Veri Madenciliği Kavramı Ve Uygulama Alanları, Maltepe ünüversitesi-istanbul.
16. Na, S., Xumin, L., and Yong, G., 2010. Research on k-ortalama Clustering Algorithm An Improved k-ortalama Clustering Algorithm, IEEE digital library, 978-1-4244-6730-3(11261758), 63 – 67.
17. Jiawei, H., 2006. Cluster Analysis, Data Mining: Concepts and Techniques, 13, Elsevier Inc., U.S.A, 383-464.
18. Albayrak S., and Tekbir, M., 2010. Recursive-Partitioned DBSCAN, IEEE digital library, 978-1-4244-9672-3(11688290), 113 – 116.
19. Su, X. and Wang J., 2011. An improved k-ortalama clustering algorithm, IEEE digital library, 978-1-61284-485-5(12229842), 44 – 46.
20. Foresti, G.L., Piciarelli, C., Micheloni, C., 2013. Çekirdek-based clustering, IET digital library, 19(42), 113-114.
21. Eswara Reddy, B., Hitendra Sarma, T., Viswanath, P., 2012. Speeding-up the çekirdek k-ortalama clustering method: A prototype based hybrid approach, SinceDirect, 34(5), 564-573.
22. Eswara Reddy, B., Hitendra Sarma, T., Viswanath, P., 2012. A Fast Approximate Çekirdek k-ortalama Clustering Method For Large Data sets, IEEE digital library, 978-1-4244-9477 (11), 545-550.
23. Hirschberg J., and Rosenberg, A., 2007. V-Measure: A conditional entropy-based external cluster evaluation Measure, citeseer,410–420.