Journal of the Faculty of Engineering and Architecture of Gazi University 34:3 (2019) 1281-1293

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Differential evolution based multiple kernel fuzzy clustering

Emrah Hançer*
Department of Computer Technology and Information Systems, Mehmet Akif Ersoy University, Burdur, 15400, Turkey

Highlights:

- Using multiple kernel functions, a fuzzy clustering objective function is proposed.
- The proposed objective function is integrated into the differential evolution framework.
- A comparative analysis of the proposed clustering algorithm with well-known partitional clustering algorithms is performed.

Keywords:

- Kernel clustering
- Fuzzy c-means
- Differential evolution

Article Info:
Research Article
Received: 29.12.2017
Accepted: 10.04.2018

DOI:
10.17341/gazimmfd.460525

Correspondence:
Author:Emrah Hancer
e-mail: ehancer@mehmetakif.edu.trph
one: +90 248 213 87 47

Graphical/Tabular Abstract

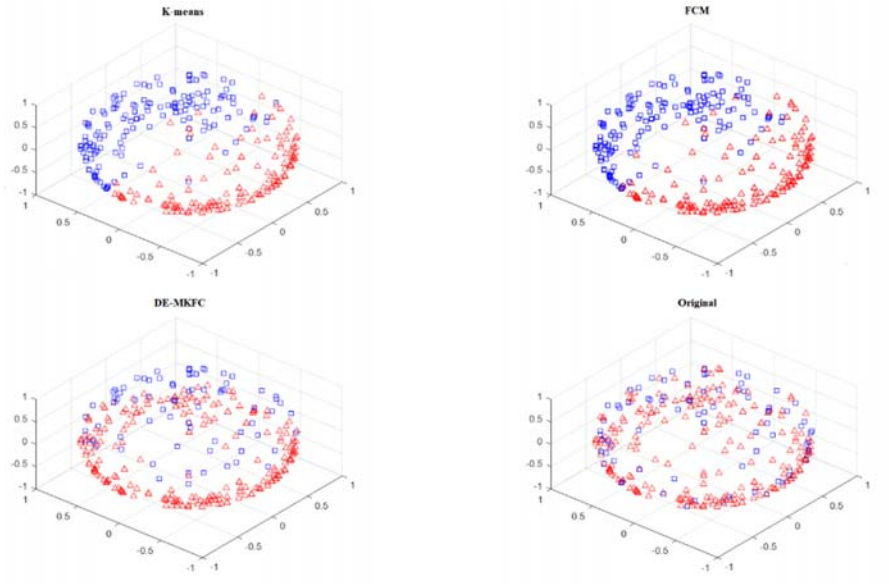


Figure A. The visual results of Haberman dataset.

Purpose: The overall goal of this paper is to propose a multiple kernel fuzzy clustering approach using differential evolution (DE) to detect not only arbitrarily shaped clusters but also overlapping clusters in the data.

Theory and Methods:
The overall schedule of the proposed DE-based multiple kernel fuzzy clustering algorithm (DE-MKFC) is as follows. First, the data is first normalized and then transformed to the kernel space using a type of exponential function. Then, the population is randomly initialized with *K* number of cluster centroids. The quality of each individual is evaluated using the multiple kernel version of the fuzzy within-cluster distance.

Results:
To verify the effectiveness of the proposed DE-MKFC algorithm, it is analyzed and compared with representative clustering approaches, such as K-means, FCM, and MKFC on a variety of well-known benchmark datasets in terms of external validity indexes. The results show that DE-MKFC can detect non-arbitrary shaped clusters and generally achieves significantly better cluster performance than existing clustering algorithms. The effectiveness of DE-MKFC can also be illustrated in Fig. A.

Conclusion:
The main goal of this paper is to develop a multiple kernel fuzzy clustering algorithm. This aim is achieved by developing an objective function based on the principles of multiple kernel clustering and then integrating this objective function in DE framework. According to the results, the proposed algorithm can improve the quality of clusters and detect the overlapping clusters. It should be notified that multiple kernel fuzzy clustering is considered for the first time using an evolutionary technique.

Journal of the Faculty of Engineering and Architecture of Gazi University 34:3 (2019) 1281-1293

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Diferansiyel gelişim tabanlı çoklu kernel bulanık kümeleme

Emrah Hançer*
Mehmet Akif Ersoy Üniversitesi, Bilgisayar Teknolojisi ve Bilişim Sistemleri Bölümü, Burdur, 15039, Türkiye

Ö N E Ç I K A N L A R

- Çoklu kernel fonksiyonları kullanılarak bulanık bir kümeleme amaç fonksiyonu geliştirilmiştir
- Geliştirilen amaç fonksiyonu diferansiyel gelişim yapısına entegre edilmiştir
- Önerilen kümeleme algoritmasının bilinen bölünmeli kümeleme algoritmalarıyla karşılaştırmalı analizi yapılmıştır

Makale Bilgileri
Araştırma Makalesi
Geliş: 29.12.2017
Kabul: 10.04.2018

DOI:
10.17341/gazimmfd.460525

Anahtar Kelimeler:
Çekirdek kümelemesi, bulanık c-araçları, diferansiyel evrim

ÖZET

K-means, bulanık c-means ve k-medoids gibi temel kümeleme algoritmaları hızlı, basit ve uygulanabilirliği kolay olması sebebiyle günümüzde hala popülerliğini korumaktadır. Ancak bu algoritmaların etkinlikleri genellikle kompakt küme setlerinin tespit edilmesi ile sınırlı kalmaktadır. Bu sebepten ötürü araştırmacılar bu algoritmaların etkinliğini arttırmak için çeşitli yöntemler üzerinde çalışmaktadır. Bu yöntemlerden birisi de kernel fonksiyonları ile veri setlerinin doğrusal uzaydan doğrusal olmayan bir uzaya aktarılarak ele alınmasıdır. Ancak tek kernel kullanımı kümelemede istenilen performansı vermeyebilmektedir. Bu çalışmada diferansiyel gelişim algoritması tabanlı çoklu kernel kümeleme algoritması önerilmiştir. Elde edilen sonuçlar neticesinde önerilen çoklu kümeleme algoritmasının bilinen kümeleme algoritmalarından daha iyi performans gösterdiği görülmüştür.

Differential evolution based multiple kernel fuzzy clustering

H I G H L I G H T S

- Using multiple kernel functions, a fuzzy clustering objective function is proposed.
- The proposed objective function is integrated into differential evolution framework.
- A comparative analysis of the proposed clustering algorithm with well-known partitional clustering algorithms is performed

Article Info
Research Article
Received: 29.12.2017
Accepted: 10.04.2018

DOI:
10.17341/gazimmfd.460525

Keywords:
Kernel clustering, fuzzy c-means, differential evolution

ABSTRACT

Fundamental clustering algorithms such as k-means, fuzzy c-means and k-medoids still survive popularity nowadays due to their efficiency, simplicity and applicability. However, their effectiveness is largely limited to spherical cluster sets. Accordingly, researchers have been studying on various techniques to improve to the effectiveness of fundamental clustering algorithms. One of the technique to improve the clustering performance of such algorithms is to map datasets from linear space to non-linear feature space using kernel functions. However, using an individual kernel may not be sufficient to obtain high clustering performance. In this study, differential evolution based multiple fuzzy kernel clustering algorithm is proposed. The results show that the proposed algorithm performs better than well-known clustering algorithms.

*Sorumlu Yazar/Corresponding Author: emrahhanc@gmail.com / Tel: +90 248 325 6534

1. GİRİŞ (INTRODUCTION)

Makine öğrenmesi, öğrenebilme yeteneğine sahip ve verinin ait olduğu olguya göre veriyi modelleyen veya işleyen algoritmalarla ilgilenen yapay zekâ disiplinin bir alt dalıdır. Makine öğrenmesi prensibini takip eden algoritmalar dinamik yapıda olup, eldeki veriye göre model ortaya çıkararak çalışırlar. Makine öğrenmesi yöntemleri temel olarak tahmin edici ve özetleyici (açıklayıcı) olmak üzere iki grupta incelenir [1]. Tahmin edici yöntemler eldeki verinin analiz edilerek ortaya çıkabilecek durumlara uygun teşhisi ortaya koyabilen modeli ortaya çıkarırken, özetleyici yöntemler eldeki veriye ait bilgiyi analiz ederek bu verideki öz ve anlamlı bilgiyi ortaya çıkarır. En çok bilinen özetleyici amaç için kullanılan makine öğrenmesi yöntemleri kümeleme, ayrıklık tespiti ve birliktelik kurallarının keşfidir. En çok kullanılan tahmin edici amaç için kullanılan makine öğrenmesi yöntemleri ise regresyon analizi ve sınıflandırmadır. Bu çalışmada kümeleme üzerine yoğunlaşılacaktır.

Kümeleme bir veriyi küme içi benzerliğin maksimum ve kümeler arası benzerliğin minimum olacak şekilde alt kümelere (gruplara) ayırma işlemidir. Kümeleme algoritmaları sistematik olarak hiyerarşik, bölünmeli, yoğunluk tabanlı, alan tabanlı ve model tabanlı olmak üzere beş grupta incelenebilir [2]. Bu algoritmaların arasında basit ve uygulanabilirliği kolay olması sebebiyle bölünmeli algoritmalar literatürde önemli yer tutmaktadır.

K-means algoritması [3] bölünmeli kümeleme algoritmalarının temeli niteliğindedir. K-means, rasgele oluşturulmuş K tane küme merkezine Öklid mesafesi prensibine göre veri setindeki elemanları atar. İlgili küme merkezleri bu atama işleminden sonra güncellenir ve durdurma kriteri sağlanıncaya kadar atama-güncelleme işlemi tekrarlanır. K-means algoritmasına benzer mekanizmaya sahip K-medoids [4] ise, K tane en iyi temsil edilecek nesneyi bulma prensibine göre çalışır. Temsil rolünü üstlenecek nesnenin bünyesindeki nesnelere olan ortalama mesafesinin minimum olması beklenir. Bir diğer bilenen bölünmeli kümeleme algoritması fuzzy c-means (FCM) [5] veri setindeki her eleman için mevcut kümelere ait üyelik katsayısı tanımlar ve bu katsayıları güncelleme prensibine göre çalışır. Durdurma kriteri sağlandığında, verideki elemanlar en fazla üyelik katsayısına sahip olan kümelere atanır. Her ne kadar FCM algoritması, K-means ve K-medoids algoritmalarından homojen dağılımlı veri setlerinde daha iyi performans sağlasa da her üç algoritma da başlangıç koşullarına bağımlı ve gürültüden olumsuz olarak etkilenebilmektedir. Bu algoritmalarda ayrıca yerel yakınsama problemleri de ortaya çıkabilmektedir [6]. İlgili bölünmeli algoritmaların yerel yakınsama ve başlangıç koşullarından etkilenme sorunlarını çözmek amacıyla evrimsel hesaplama teknikleri kullanılarak birçok kümeleme yöntemi geliştirilmiştir. Özellikle genetik algoritma [7], parçacık sürü optimizasyon [8], diferansiyel gelişim [9] ve yapay arı koloni algoritması [10] tabanlı geliştirilen literatürde birçok çalışma mevcuttur. Ancak geliştirilen bu yöntemlerin de yerel yakınsama problemlerini tamamıyla ortadan kaldırdığını söylemek mümkün değildir [2]. İlgili bölünmeli algoritmaların performansını arttırmaya yönelik öne sürülen bir diğer çözüm ise kernel fonksiyonları ile veri setlerinin linear uzaydan linear olmayan uzaya aktarılarak ele alınmasıdır. Ancak tüm veri setlerine uygun bir kernel fonksiyonu mevcut değildir ve hangi kernel fonksiyonunun kullanılacağı ciddi bir soru işaretidir. Bu yüzden tek bir belirlenmiş kernel kullanmak yerine birden fazla baz kernel içeren çoklu kombinasyonun kümeleme algoritmalarında kullanılması düşünülmüştür. Zhao vd. [11] çoklu kernel maksimum margin kümeleme yöntemini (MMC) önermiştir. Ancak önerilen yöntem daha çok spesifik veri setleri için daha uygundur. Huang vd. [12], çoklu kernel fonksiyon kombinasyonu ile FCM'nin gelişmiş bir versiyonunu (MKFC) ortaya koymuşlardır. MKFC algoritması MMC algoritmasına göre daha fazla veri setine uygun ve uygulanabilirliği daha kolaydır. MMC algoritması kümelemeyi linear olmayan optimizasyon problemi olarak ele almıştır ve bu sebeple çözümü daha zordur. Wang vd. [13] ise adaptif bir çoklu kernel kümeleme algoritması (MKFC-IK) önermiştir. MKFC-IK algoritması kümeleme işleminden önce uygun olmayan kernel değerlerini düzelterek kümeleme performansını arttırmayı amaçlamıştır. MKFC-IK son zamanlarda ortaya çıkan kernel kümeleme algoritmalarından daha iyi performans gösterse de klasik kümeleme algoritmalarıyla karşılaştırmalı bir performans analizi yapılmamıştır.

Sonuç olarak çoklu kernel kümeleme literatüre son yıllarda dâhil olmuş yeni bir araştırma konusu olup, yapılan çalışmalar henüz başlangıç düzeyindedir. Ayrıca kümeleme problemlerine global araştırma yeteneği nedeniyle sıklıkla uygulanan evrimsel hesaplama tekniklerinin de çoklu kernel kümelemeyi ele alan bir çalışması literatürde henüz mevcut değildir. Bu çalışmanın temel amacı diferansiyel gelişim algoritması tabanlı bulanık bir çoklu kernel kümeleme algoritması (DE-MKFC) geliştirmektedir. Bu amacı gerçekleştirmek için FCM ve iki farklı kernel fonksiyonunun kombinasyonuyla oluşan bir amaç fonksiyonu geliştirilmiş ve geliştirilen amaç fonksiyonu diferansiyel gelişim algoritmasına entegre edilmiştir. Çalışmayla ilgili temel olarak aşağıdaki hususlar ele alınmıştır:

- DE-MKFC algoritmasının bilinen kümeleme algoritmaları ile sayısal performans analizi
- DE-MKFC algoritmasının bilinen kümeleme algoritmaları ile görsel performans analizi

Makalenin organizasyonu şu şekildedir: Giriş Bölümünü takiben Bölüm 2'de diferansiyel gelişim algoritması, FCM, çoklu kernel kümeleme ve literatür taramasını içine alan ön bilgilere yer verilmiştir. Bölüm 3'te önerilen çoklu kernel kümeleme algoritması sunulmuştur. Bölüm 4'te çalışmada kullanılan veri setleri ve parametre değerlerini içine alan deneysel dizayn açıklanmıştır. Bölüm 5'te algoritmaların

sayısal ve görsel kümeleme sonuçlarına yer verilmiş ve bu sonuçlarla ilgili performans analizleri yapılmıştır. Son bölümde de yapılan çalışma ile ulaşılan genel sonuçlar açıklanmış ve ileride yapılacak olan çalışmalara ışık tutulmaya çalışılmıştır.

2. ÖN BİLGİLER (BACKGROUND)

2.1. Diferansiyel Gelişim (Swarm Intelligence)

Diferansiyel gelişim algoritması (DE) [14] Storn ve Price tarafından 1995 yılında geliştirilmiş popülasyon tabanlı bir evrimsel hesaplama tekniğidir. DE algoritması yapı olarak basit, uygulanabilirliği kolay ve spesifik problemlere uyarlanabilmesi sebebiyle evrimsel hesaplama teknikleri içinde önemli bir yere sahiptir. Ayrıca birçok çeşitli problemde diğer evrimsel hesaplama tekniklerine göre daha iyi performans sağladığı görülmüştür [15]. DE temel olarak mutasyon, çaprazlama ve seçme olmak üzere üç evrimsel operatörden oluşmaktadır. Mutasyon çözüm uzayında umut vadeden bölgelere yönlendirirken, çaprazlama popülasyon içinde bilgi paylaşımını sağlayarak daha iyi çözümleri ortaya çıkarır. DE algoritmasının temel adımları şöyledir:

- Ön hazırlık: Bu aşamada başlangıç popülasyonu üretimi (Eş. 1) gerçekleştirilir. Popülasyon büyüklüğü en az 4 olmalıdır.

$$x_{ij} = x_j^{min} + U(0,1)\left(x_j^{max} - x_j^{min}\right) \quad (1)$$

Burada x_{ij} *i.* bireyin *j.* pozisyonunu, *i={1,2,...,NP} j={1,2,...,D}*, *NP* popülasyon büyüklüğü ve *D* toplam popülasyon sayısını ifade eder; x_j^{max} ve x_j^{min} pozisyonların alt ve üst sınırlarını belirtir ve $U_j(0,1)$ *j.* pozisyon için 0 ve 1 arasında rasgele üretilmiş bir sayıdır.

- Mutasyon: Her bir X_i bireyi için popülasyondan rasgele üç tane X_{r1}, X_{r2} ve X_{r3} bireyi seçilir ve (Eş. 2) ile mutasyona uğramış $\bar{X}_i$ bireyi üretilir:

$$\bar{X}_i = X_{r1} + F(X_{r2} - X_{r3}) \quad (2)$$

Burada *F* parametresi [0,1] arasında tanımlanmış ölçekleme faktörüdür.

- Çaprazlama: Mutasyon işleminden sonra her bir X_i ve onun mutasyona uğramış bireyi $\bar{X}_i$ arasında Eş. 3 ile çaprazlama işlemi uygulanarak bir deneme bireyi üretilir.

$$u_{id} = \begin{cases} \overline{x_{id}}, & U(0,1)_d \le CR \text{ or } d = rn_i \\ x_{id}, & otherwise \end{cases} \quad (3)$$

Burada *CR* kullanıcı tarafından tanımlanmış çaprazlama oranı, $U(0,1)_d$ *d* pozisyonu için [0,1] arasında üretilmiş rasgele bir değer, rn_i *i.* birey için rasgele seçilmiş bir pozisyon ve u_{id} ise $U_i = \{u_{i1}, u_{i2}, \dots, u_{id}, \dots, u_{iD}\}$ deneme vektörünün *d.* pozisyonudur.

- Seçme: Çaprazlama işlemi tamamlandıktan sonra X_i popülasyon bireyi ve U_i üretilmiş deneme bireyi arasında seçim yapılır. Eğer U_i problem için daha iyi çözümse, X_i yerine popülasyonun gelecekteki gelişim aşamalarında temsil edilir. Aksi takdirde X_i temsil edilmeye devam eder.

2.2. Bulanık C-Means (Fuzzy C-Means)

Bölünmeli kümeleme algoritmaları katı ve yumuşak olmak üzere iki kategoride ele alınabilir [16]. Katı algoritmalar veri elemanının tek bir kümeye ait olabileceği prensibini benimserken, yumuşak algoritmalar veri elemanın birden fazla kümeye üye olabileceği prensibiyle çalışır. Yumuşak kümeleme prensibine göre çalışan en çok bilinen algoritma bulanık c-means (FCM) algoritmasıdır [5]. FCM algoritmasında veri elemanlarının her birisi 0 ve 1 arasında bir üyelik değeri ile mevcut kümelere atanır. Bir veri elemanın sahip olduğu üyelik değerlerinin toplamı 1'e eşittir. İlgili veri elemanı hangi kümeye daha yakınsa o kümeye ait üyelik değeri diğerlerine nazaran daha büyük olması beklenir. FCM temel olarak aşağıdaki amaç fonksiyonunu minimize etmeyi amaçlar (Eş. 4):

$$J(U,C) = \sum_{i=1}^{N}\sum_{j=1}^{K} u_{ij}{}^{m}\left\|x_i - c_j\right\|, \quad \sum_{j=1}^{K} u_{ij} = 1 \quad (4)$$

Burada *m* bulanıklık katsayısını ifade etmektedir ve 1'den büyük bir değere sahip olmalıdır; C={c_1, c_2, …, c_j, …, c_K} küme merkez setini, c_j *j.* kümenin merkezini ve *K* toplam küme sayını ifade eder; x_i veri setindeki *i.* elemanı ve *N* veri setindeki toplam eleman sayısını ifade eder; $U = [u_{ij}]_{i=1\dots N,\ j=1\dots K}$ $N \times K$ boyutunda üyelik matrisini ifade eder. FCM algoritmasında öncelikle rasgele olarak başlangıç *U* üyelik matrisi üretilir. Akabinde (Eş. 5) ile küme merkezleri hesaplanır. Küme merkezlerinin hesaplanmasının akabinde (Eş. 6) ile *U* üyelik matrisi güncellenir. Güncellenen *U'* matrisi kullanılarak (Eş. 4) ile *J(U',C)* hesaplanır. Eğer | *J(U',C)* - *J(U,C)* |< ϵ ise, (Eş. 5) ve (6) ile işlem tekrarlanır. Aksi takdirde algoritmanın çalışması durdurulur.

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \quad (5)$$

$$u_{ij} = \frac{1}{\sum_{j'=1}^{K}\left(\frac{\left\|x_i - c_j\right\|}{\left\|x_i - c_{j'}\right\|}\right)^{\frac{2}{m-1}}} \quad (6)$$

2.3. Literatür Taraması (Related Works)

Kernel kümeleme temel olarak eldeki verinin standart uzaydan kernel fonksiyonları ile daha yüksek uzaya taşınarak kümeleme işleminin yapılmasıdır [17]. Yüksek uzaylarda verilerin taşınıp kümelenmesiyle daha kaliteli kümeler elde edilmesi beklenir. Çoklu kernel kümeleme ise tanımlanmış birden fazla kernel kombinasyonu kullanarak kümeleme kalitesini arttırmayı amaçlar. Çoklu kernel kümeleme son yıllarda gündeme gelen ve henüz daha birkaç yıllık bir geçmişe sahip olan bir araştırma konusudur. Çoklu kernel kümeleme algoritmaları temel olarak iki kategoride incelenebilir. Bunlardan ilki üzerinde veri elemanlarının

küme içi ve küme dışı benzerlik ilişkilerini konu edinen konsensüs matrisisin low-rank optimizasyon problemi olarak ele alınıp minimize edilmesi prensibine bağlı olarak çalışır [17,18]. Xia vd. [19] her bir kernel için bir dönüşüm matrisi oluşturmuş ve bu matris Markow Chain yöntemine kümeleme için giriş parametresi olarak hazırlamıştır. Kumar ve Daume [20] çoklu kernel kümeleme ve spektral kümelemeyi bir araya getirerek bir kümeleme algoritması önermiştir. Bu algoritmada da benzerlik matrisinin güncellenmesi prensibine dayanarak çalışmaktadır. Çoklu kernel kümeleme algoritmalarının diğer kategorisi ise, çoklu kernel katsayılarının güncellenmesi ve birleştirilmiş çoklu kernel bileşenlerinin kümelemede kullanılması prensibine dayanır [17]. Bu grubun en bilinen örneklerinden olan multiple kernel fuzzy clustering (MKFC) ve multiple kernel k-means (MKKM) algoritmaları [12] iteratif olarak kernel FCM ve kernel k-means uygulayarak kernel katsayılarını güncellemeye çalışır. Lokal çoklu kernel k-means (LMKMM) algoritması [21] kernel gruplarını örneklenmiş adaptif ağırlıklandırma katsayıları ile birbirine bağlayarak çalışır. MKKM'nin bir üst versiyonu olarak robust LMKKM algoritması, MKKM'deki karesel hata toplamı yerine L2,1 normunu kullanır. Wang vd. [13] kümeleme işleminden önce uygun olmayan kernel katsayılarını güncelleyerek kümeleme performansını arttırmaya çalışmıştır. İlgili çalışma mevcut çoklu kernel kümeleme algoritmalarına göre daha iyi kümeleme performansı göstermiştir.

Her ne kadar çoklu kernel kümeleme üzerine yapılan çalışmalar genel olarak kümelemede umut vadeden sonuçlar elde etse de bazı noktalarda eksiklikleri söz konusudur. Öncelikle ilgili çalışmaların çoğunluğu sadece belli dağılıma sahip spesifik veri setleri üzerine yoğunlaşmış ve farklı dağılım gösteren veri setleri üzerinde performansları değerlendirilmemiştir. İkinci olarak geliştirilen çoklu kernel kümeleme yöntemlerinin bilinen bölünmeli kümeleme algoritmaları ile karşılaştırmalı performans analizleri genellikle yapılmamıştır. Bunun yerine daha çok kendi grubuna dahil çoklu kernel kümeleme algoritmaları ile karşılaştırmalı analizler yapılmıştır. Bir diğer önemli eksiklik ise, çoklu kernel kümeleme konusu henüz literatürde evrimsel hesaplama teknikleri ile ele alınmamıştır. Sonuç olarak çoklu kernel kümeleme henüz yeni bir araştırma konusu olup, bu konu ile ilgili araştırma ve çalışmalar halen devam etmektedir.

3. DE-MKFC ALGORİTMASI (DE-MKFC ALGORITHM)

Mevcut çoklu kernel kümeleme algoritmaları umut vaat eden sonuçlara ulaşmış olsa da bu yöntemlerin henüz klasik kümeleme yöntemlerinden daha iyi sonuçlar sağladığı hususunda net bir sonuç ortaya konulmuş değildir. Ayrıca çoklu kernel kümelemenin henüz bir evrimsel hesaplama tekniği ile ele alınmış bir uygulaması da mevcut değildir. Bu konudaki eksiklik sebebiyle bu çalışmada diferansiyel gelişim tabanlı çoklu kümeleme algoritması (DE-MKFC) geliştirilmiştir. DE algoritması mevcut evrimsel hesaplama teknikleri içinde birçok problemde başarı ile uygulanması, daha az parametre içermesi ve uygulanabilirliği daha kolay olması sebebiyle bu çalışmada tercih edilmiştir [15].

Bu bölümde öncelikle kümelemenin genel işleyişi sunulacak. İkinci kısımda verinin kernel kümeleme için hazırlanması aşaması anlatılacak. Akabinde çoklu kernel amaç fonksiyonu verilecek ve son olarak DE-MKFC ile kümeleme işlemi açıklanarak bölüm sonlandırılacaktır.

3.1. Kümeleme Genel Yapısı (Overall Structure of Clustering)

Çalışmada uygulanan genel kümeleme yapısı Şekil 1'de sunulmuştur. Öncelikli olarak veri setinde eğer etiket bilgisi mevcutsa, veri seti etiketli ve etiketsiz olarak ikiye ayrılır. Daha sonra etiketsiz veri kümeleme algoritması kullanılarak kümelerine ayrılır ve verilere ait kümeleme etiketi bilgisi elde edilir. Son olarak veri etiketi ve kümeleme etiketi bilgisi kullanılarak küme doğrulama işlemi gerçekleştirilir ve algoritmanın ilgili veri seti üzerindeki kümeleme performansı ölçülür. Veri setindeki etiket bilgisi sadece doğrulama işleminde kullanılır. Bir diğer söylemle kümeleme esnasında veri etiketi bilgisi algoritma tarafından bilinmemektedir.

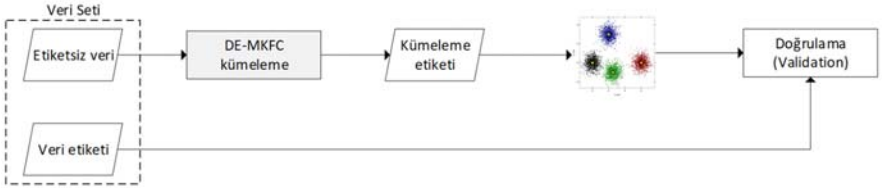
3.2. Veri Setinin Hazırlanması (Preparation of Dataset)

Kernel, veri elemanlarını haritalama yöntemi ile daha uygun bir şekilde temsil etmek için kullanılan bir yoldur. Farklı büyüklükteki resimler, farklı uzunlukta protein dizilimleri, üç boyutlu nesneler ve farklı uzunluk ve formattaki metin dokümanların karşılaştırılmasında önemli bir rol üstlenmiştir.

Verilen bir Z verisi (dokümanlar, resimler vb.) için kernel fonksiyonu κ: $Z \times Z \rightarrow \Re$ şeklinde tanımlanır. Kernel fonksiyonları Z verisindeki iki farklı elemanın (z ve z') arasındaki benzerliği ölçer. Temel bir kernel fonksiyonu şu özellikleri sağlamalıdır:

- $\forall z, z' \in Z, \kappa(z,z') = \kappa(z',z)$,
- $\forall z, z' \in Z, \kappa(z,z') \geq 0$.

X uzayından N tane eleman ve M tane öznitelikten oluşan $(N \times M)$ bir $Z = \{z_1, z_2, \ldots, z_N\}$ verisi seçilmiş olsun. Z veri



Şekil 1. Genel yapı (Overall structure)

setinin gram matrisi $K(Z,\kappa) \in \Re^{N\times N\times M}$ öyle ki $(K)_{ij} = \kappa(z_i,z_j)$'dir. Eğer $\forall\, Z \subset X$ ise, K matrisi pozitif tanımlıdır ve dolayısıyla κ mercel kernelidir. Mercel kerneli temel olarak şu eşitlik ile ifade edilebilir:

$$K_{ij} = \kappa(z_i, z_j) = \phi(z_i)^T \phi(z_j) \tag{7}$$

Burada $\phi(.)$: $X \rightarrow M$ tanımlı bir basis fonksiyondur. M boyutu için herhangi bir kısıtlama olmayıp sonsuz olarak da tanımlı olabilir. Basis fonksiyonu ($\phi(.)$) mercel kerneller için κ'ya ait öz (eigen) fonksiyonlarının lineer kombinasyonu olarak tanımlanabilir:

$$\begin{aligned} &\phi = \omega_1\phi_1 + \omega_2\phi_2 + \cdots + \omega_M\phi_M \\ &\omega_1 + \omega_2 + \cdots + \omega_M = 1,\ \ \omega_k \geq 0 \end{aligned} \tag{8}$$

Bu çalışmada veri setinin daha yüksek bir uzayda temsil edilebilmesi için gauss kernel fonksiyonu kullanılmıştır. Gauss kernel değerleri (Eş. 9) ile hesaplanır. Kernel değerleri hesaplanmadan önce veri setine normalizasyon işlemi yapılarak elde edilecek kernel değerlerinin tutarlı bir düzeyde olması amaçlanmıştır.

$$\kappa(z_i, z_j = \exp\left(-\sum_{d=1}^{D} \frac{\|z_{jd}(:) - z_{id}(:)\|^2}{\sigma_d}\right) \tag{9}$$

Burada $z_i = \{z_{i1}, z_{i2}, \ldots, z_{id}, \ldots, z_{iM}\}$ veri setindeki M boyutlu *i.* veri elemanını ve σ_d *d.* özniteliğin ölçekleme faktörünü ifade eder:

$$\sigma_d = \frac{\left(\max\limits_{i=1,\ldots N} z_{id} - \min\limits_{i=1,\ldots N} z_{id}\right)^2}{v} \tag{10}$$

Burada v kullanıcı tarafından tanımlanan bir katsayıdır.

3.3. Amaç Fonksiyonu (Objective Function)

(Eş. 4) ile tanımlanmış olan FCM amaç fonksiyonu farklı bir yüksek uzaya haritalanmak istendiğinde şu forma dönüştürülür:

$$J(U,C) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}{}^m (\phi(z_i) - c_k)^T (\phi(z_i) - c_k) \tag{11}$$

Burada $\phi(z_i) = \sum_{d=1}^{M} \omega_d \phi_d(z_i)$ olup (Eş. 7) ile tanımlanmıştır.

$D_{ik}^2 = (\phi(z_i) - c_k)^T (\phi(z_i) - c_k)$ şeklinde tanımlarsak, (Eş. 11)'i daha kısa ve anlaşılır şekilde yazabiliriz:

$$J(U,C) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}{}^m D_{ik}^2 \tag{12}$$

D_{ik}^2 parçacığı ϕ basis fonksiyonunun ağırlıklandırma (ω) parametresi cinsinden ise şu şekilde yazılır:

$$D_{ik}^2 = \sum_{d=1}^{M} \alpha_{ikd} \omega_d{}^2 \tag{13}$$

Burada α_{ikd} katsayısı (Eş. 14) ile şu şekilde tanımlanmıştır:

$$\alpha_{ikd} = \kappa_d(z_i, z_i) - 2\sum_{j=1}^{N} \hat{u}_{jk}\kappa_d(z_i, z_j) + \sum_{j=1}^{N}\sum_{j'=1}^{N} \hat{u}_{jk}\hat{u}_{j'k}\kappa_d(z_j, z_{j'}) \tag{14}$$

Burada $\hat{u}_{jk}$normalize edilmiş üyelik değeridir ve (Eş. 15) ile ifade edilir.

$$\hat{u}_{jk} = \frac{\hat{u}_{jk}}{\sum_{k=1}^{K} \hat{u}_{jk}} \tag{15}$$

(Eş. 13)'ü (Eş. 12)'de yerine koyduğumuzda amaç fonksiyonun son hali şu şekilde olur:

$$J(U,w) = \sum_{i=1}^{N}\sum_{j=1}^{K} u_{ij}{}^m \sum_{d=1}^{M} \alpha_{ijd}\omega_d{}^2 \tag{16}$$

Üyelik değerleri belirlenmişse, (Eş. 16) şu şekilde ifade edilebilir:

$$J(U,w) = \sum_{d=1}^{M} \beta_d \omega_d{}^2 \tag{17}$$

Burada β_d katsayısı şu şekilde ifade edilir:

$$\beta_d = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ik}{}^m \alpha_{ikd} \tag{18}$$

(Eş. 18) sınırlamalı bir optimizasyon problemi olarak ele alınabilir ve gerekli hesaplamaların ardından ω_d şu şekilde hesaplanır:

$$\omega_d = \frac{\frac{1}{\beta_d}}{\frac{1}{\beta_1} + \frac{1}{\beta_2} + \cdots + \frac{1}{\beta_M}} \tag{19}$$

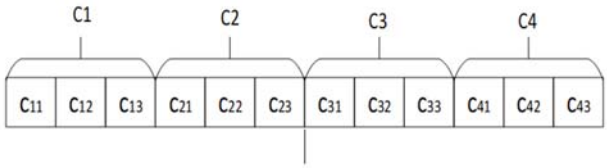
3.4. DE-MKFC ile Kümeleme (Clustering with DE-MKFC)

DE-MKFC'de her X_i bireyi K tane rasgele üretilmiş veya veri setinden seçilmiş küme merkezinden ($\{c_{i1}, c_{i2}, \ldots, c_{iK}\}$) oluşur. M boyutlu bir veri için X_i bireyinin uzunluğu $K\times M$'dir. Örneğin, M=3 ve K=4 için bir bireyin sahip olacağı uzunluk Şekil 2'de görüleceği üzere 12 olacaktır.

DE-MKFC'de bir bireyin amaç fonksiyon değerinin hesaplanabilmesi için öncelikle veri elemanı atanamayan boş bir küme olup olmadığı kontrol edilir. Çünkü boş kümenin ortaya çıkması erken yakınsama problemlerine neden olabilmektedir. Boş kümenin saptanabilmesi için veri elemanları en yakın Öklid mesafesine sahip kümeye atanır. Şayet veri elemanı atanamayan bir küme ortaya çıkarsa X_i bireyi için küme merkezleri rasgele tekrar üretilir. X_i bireyinde boş küme ortaya çıkmaması durumunda (Eş. 6) ile ilgili küme merkezleri ve veri elemanları kullanılarak U üyelik matrisi elde edilir. Ardından (Eş. 16) ile *J(U,w)* amaç fonksiyon değeri hesaplanır. Bir bireyin amaç fonksiyonu değerinin nasıl hesaplandığını kısaca maddeler halinde özetleyecek olursak:

- Bireyindeki küme merkezleri kullanılarak (Eş. 6) ile U matrisi elde edilir.
- U matrisi (Eş. 15) ile normalize edilir.

- Normalize edilmiş U matrisi kullanılarak α_{ikd} katsayı değerleri (Eş. 14) ile hesaplanır.
- α_{ikd} ve U matrisi kullanılarak β_d katsayı değerleri (Eş. 18) ile hesaplanır.
- β_d katsayıları kullanılarak ω_d ağırlıklandırma değerleri (Eş. 19) ile hesaplanır.
- $J(U,w)$ amaç fonksiyon değeri (Eş. 16) ile hesaplanır.



Şekil 2. 3 boyutlu uzaydaki 4 kümeli bir birey
(An individual with 4 clusters in 3-dimesional space)

DE algoritmasında (Eş. 2) ile tanımlanmış mutasyon işlemi bireyin tüm pozisyonlarında gerçekleşmesi, kümeleme merkezlerinin yerlerinin hızlı bir şekilde değişmesine ve bu da küresel-yerel arama dengesinin bozulmasına neden olacaktır. Bu sebepten ötürü (Eş. 2) DE-MKFC'de her merkezin rasgele tek bir pozisyonunda olacak şekilde modifiye edilerek (Eş. 20) haline gelmiştir.

$$x_{ij_p} = x_{r1j_p} + F\left(x_{r2j_p} - x_{r3j_p}\right),$$
$$(k-1) * dim + 1 \leq j_p \leq k * dim, \forall\, k = 1,2,\dots,K \quad (20)$$

Burada *dim* parametresi X_i bireyin uzunluğunu temsil edip $K \times M$ kadardır ve j_p X_i bireyindeki her bir c_{ij} küme merkezindeki rasgele seçilen pozisyonu temsil eder.

DE-MKFC algoritmasında küme merkezleri rasgele olarak üretilmektedir ve dolayısıyla algoritmanın performansı başlangıç koşullarına bağımlıdır. Bir diğer söylemle kümeleme performansı oluşturulan küme setlerine bağlı olarak değişim gösterilecektir. Başlangıç koşullarının olumsuz etkisini azaltmak amacıyla DE-MKFC'de her iterasyon sonunda bireylere olasılıklı olarak 10 iterasyonluk bir K-means algoritması uygulanır. Eğer ilgili birey için üretilen [0,1] arasındaki rasgele sayı kullanıcı tarafından belirlenen T eşik değerden küçükse K-means uygulanır. Aksi takdirde uygulanmaz. T eşik değeri 0.1 gibi küçük bir değer alınmıştır.

DE-MKFC kümeleme algoritmasının kaba kodu Şekil 3'te sunulmuştur. Şekil 3'ten görüleceği üzere giriş değerleri olarak veri seti, kernel matrisi, küme sayısı ve DE algoritma parametreleri alınır. Gerekli hesaplamaların ardından en iyi bireyin kümeleme etiketi çıkış olarak verilir. Kümeleme etiketi kullanılarak da kümeleme performans analizi işlemleri gerçekleştirilir.

4. DENEYSEL DİZAYN (EXPERIMENTAL DESIGN)

Bu çalışmada farklı sayıda eleman, öznitelik ve sınıf sayısına sahip toplam 10 veri seti kullanılmış ve ilgili veri setleri Tablo 1'de sunulmuştur. İlgili veri setlerinin 9'u UCI makine öğrenmesi veri tabanından [22], birisi de sentetik verilerden seçilmiştir. Veri setlerinin seçiminde [23]'deki önerilen veri setlerine ağırlık verilmiştir.

Önerilen algoritmanın performans analizi belirtilen veri setleri üzerinde birçok çeşitli kümeleme algoritması ile karşılaştırılarak yapılmıştır. K-means [3], FCM [4] ve K-medoids [5] gibi temel bölünmeli kümeleme algoritmalarının yanında ortalama karesel hatanın minimizasyonunu baz alan temel DE bölünmeli [24] ve PSO bölünmeli kümeleme [25] algoritmaları ve çoklu kernel kümemelenin en bilinen örneği olan MKFC [12] kümeleme

Giriş: N eleman ve M öznitelikli veri $Z = \{z_p\}_{p=1}^{N}$, $N \times N \times M$'lik kernel matrisi $\{\kappa_d\}_{d=1}^{M}$, K küme sayısı, DE parametreleri (NP, max_{cycle}, CR ve F);
begin
  Popülasyondaki her bireye (X_i) K tane rasgele oluşturulmuş küme merkezi ata ve denklem (16) ile $f(X_i)$ değerini hesapla;
  for $t = 1$ **to** max_{cycle} **do**
    foreach *birey* X_i **do**
      Rasgele üç tane r_1, r_2 ve r_3 bireyi seç;
      foreach *küme* k **do**
        $(k-1) * K * M + 1 \leq j_p \leq k * K * M$ aralığında rasgele bir j_p pozisyonu seç;
        Seçilen j_p pozisyonunda denklem (20) ile mutasyon yap;
      end
      X_i ile mutasyona uğramış $\hat{X}_i$ arasında denklem (3) ile U_i üret;
      $f(U_i)$ değerini denklem (16) ile hesapla;
      if $f(U_i) < f(X_i)$ **then**
        U_i deneme bireyini X_i bireyinin yerine ata;
      end
      if $rand(0,1) < T$ **then**
        X_i bireyine K-means uygula;
      end
    end
  end
end
Çıkış: X_{best} en iyi bireyin kümeleme etiketi;

Şekil 3. DE-MKFC algoritmasının kaba kodu (The pseudo code of DE-MKFC algorithm)

algoritması deneysel çalışmalarda kullanılmıştır. Bu algoritmaların kümeleme performanslarının değerlendirilmesi için kullanılan dışsal indeksler Tablo 2'de sunulmuştur. Tablo 2'deki kullanılan sembollerin açıklamaları aşağıda belirtilmiştir:

- R: Kümeleme sonucu elde edilen etiketler, Q: Veri seti etiketleri,
- I: R setinin sahip olduğu kümeler, J: Q setinin sahip olduğu kümeler,
- TP (True Positive): R kümesinde olup gerçekten de Q kümesinde olanlar,
- TN (True Negative): (Yanlışlıkla) R kümesinde olup aslında Q kümesinde olanlar,
- FP (False Positive): R kümesinde olmayıp gerçekten de Q kümesinde olmayanlar,
- FN (False Negative): (Yanlışlıkla) R kümesinde olmayıp aslında Q kümesinde olanlar,
- H(R) R kümesinin entropisi olup, $-\sum_{i=1}^{I} P(i)\log P(i)$ şeklinde tanımlanır.

DE ve PSO algoritmasının parametre değerlerine literatürdeki bilgiler ve yapılan denemeler sonucunda karar verilmiştir. Popülasyon büyüklüğü ve maksimum çevrim (iterasyon) sayısı ilgili çalışmada [25] olduğu gibi sırasıyla 20 ve 100 olarak seçilmiştir. CR oranı ve F faktörü ise sırasıyla 0.8 ve 0.7 olarak belirlenmiştir. DE-MKFC, FCM ve MKFC algoritmalarının tamamında μ katsayısı 1.08 olarak alınmıştır [12].

Tablo 2. Çalışmada Kullanılan Dışsal İndeksler
(External Indexes Used in Experimental Study)

İndeks	Formülü
Fowlkes and Mallows (FM)	$\frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$
Jaccard Index (Jac)	$J(R,Q)=\frac{\lvert R\cap Q\rvert}{\lvert R\cup Q\rvert}$
Normalized Mutual Information (NMI)	$\frac{\sum_{i=1}^{I}\sum_{j=1}^{J} P(i,j)\log\frac{P(i,j)}{P(i)P(j)}}{\sqrt{H(R)H(Q)}}$
Rand Index (RI)	$\frac{TP+TN}{TP+FP+FN+TN}$

5. DENEYSEL ÇALIŞMALAR
(EXPERIMENTAL STUDIES)

Kümeleme performans sonuçları her bir veri seti için 30 koşma üzerinden Tablo 3'te ortalama ve standart sapma cinsinden verilmiştir. Standart sapma değerleri parantez içinde belirtilmiştir. Sonuçlar arasında anlamlı bir farklılık olup olmadığına ilişkin değerlendirmeler yapabilmek amacıyla DE-MKFC ve diğer algoritmalar arasında Wilcoxon Rank Sum Test uygulanmış ve sonuçları Tablo 4'te verilmiştir. Tablo 4'te yer alan sembollerin anlamı şu şekildedir:

- '+': DE-MKFC algoritmasının ilgili algoritmadan daha iyi sonuçlar ürettiğini,
- '-': DE-MKFC algoritmasının ilgili algoritmadan daha kötü sonuçlar ürettiğini, ve
- '=': DE-MKFC algoritmasının ilgili algoritmayla benzer sonuçlar ürettiğini ifade eder.

5.1. Kümeleme İndeks Sonuçları (Results of Clustering Indexes)

Tablo 3'ten görüleceği üzere önerilen DE-MKFC algoritması tüm kümeleme indeks kriterleri için genel olarak tüm veri setlerinde diğer algoritmalara göre daha iyi performansı göstermiştir. Özellikle Glass, Haberman, Horse, Sentetik, Thyroid ve TAE veri setlerinde DE-MKFC ve diğer kümeleme algoritmalar arasındaki fark net bir şekilde ortaya çıkmıştır. Örneğin DE-MKFC algoritması Haberman veri setinde FM ve Jac indeksleri için sırasıyla 0.6922 ve 0.5293 değerlerini elde ederken, diğer algoritmalarda bu değerler 0.55 ve 0.37 seviyesinde kalmıştır. Yine Thyroid veri setinde DE-MKFC algoritması FM ve Jac indekslerinde sırasıyla 0.6495 ve 0.4770 değerlerini elde ederken, diğer algoritmalarda bu değerler 0.52 ve 0.34 seviyesinde kalmıştır. DE-MKFC'nin diğer algoritmalardan daha iyi kümeleme performansı gösterdiği Tablo 4'te Wilcoxon Rank Sum Test sonuçları ile de desteklenmiştir. Tablo 4'ten görüleceği üzere 5 veri setinde tüm indeksler için DE-MKFC diğer algoritmalardan daha anlamlı sonuçlar elde etmiştir. Diğer veri setlerinde ise, en az iki veya üç indekste diğer algoritmalardan daha anlamlı sonuçlar elde etmeyi başarmıştır.

Karşılaştırmada kullanılan diğer algoritmalar değerlendirildiğinde, en iyi bilinen klasik bölünmeli algoritmalardan K-means ve FCM genel olarak tüm veri setlerinde birbirine yakın kümeleme performansı elde ettiği Tablo 3'te görülmektedir. Sadece Sentetik veri setinde K-means FCM'ye göre daha iyi sonuçlar elde etmiştir. Bir diğer bölünmeli kümeleme kategorisindeki K-medoids algoritması FCM ve K-means algoritmalarından Horse, Sentetik, Spect ve Thyroid gibi veri setlerinde daha kaliteli kümeler oluşturmuştur. K-means algoritmasından esinlenerek oluşturulmuş olan DE-MSE ve PSO-MSE algoritmalarının ise diğer bölünmeli algoritmalara oranla net bir üstünlük sağladığını söylemek mümkün değildir. Yer yer bazı veri setleri ve indeksler için daha iyi sonuçlar elde etse de bu başarısını diğer veri setleri ve indeksler için sürdürememiştir. Çoklu kernel kümelemenin ilk örneklerinden olan MKFC algoritması ise diğer bölünmeli yöntemlerden genel olarak daha iyi performans gösterdiği görülmektedir. Bir diğer söylemle DE-MKFC'nin ardından en iyi ikinci performansa sahiptir. Ancak MKFC algoritması da Haberman ve Sentetik gibi bazı veri setlerinde DE-MKFC'nin performans olarak oldukça gerisinde kaldığı görülmektedir. Sadece Fisher veri setinde DE-MKFC'yi geçebilmeyi başarmıştır. Sonuç olarak önerilen DE-MKFC algoritması kümeleme analizinde diğer algoritmalara karşı gözle görülür üstünlük sağladığı ve kümeleme analizinde kullanılabilecek potansiyele sahip olduğu görülmüştür.

Tablo 3. Kümeleme İndeks Sonuçları (Results of Clustering Indexes)

Veri Setleri	İndeks	DE-MSE	PSO-MSE	MKFC	K-medoid	FCM	K-means	DE-MKFC
Breast	FM	0,9426 (0,003)	0,9462 (0,001)	0,9463 (5,6E-16)	0,9437 (2,2E,16)	0,9462 (4,5E-16)	0,9462 (4,5E-16)	0,9463 (5,6E-16)
	JAC	0,8979 (0,006)	0,8979 (0,001)	0,8981 (3,3E-16)	0,8934 (4,5E-16)	0,8978 (5,6E-16)	0,8978 (5,6E-16)	0,8981 (3,3E-16)
	NMI	0,8074 (0,012)	0,8041 (0,001)	0,7983 (2,2E-16)	0,7943 (5,6E-16)	0,8037 (2,2E-16)	0,8037 (2,2E-16)	0,7983 (2,2E-16)
	RI	0,9417 (0,004)	0,9417 (0,0004)	0,9416 (4,5E-16)	0,9389 (2,2E-16)	0,9416 (4,5E-16)	0,9416 (4,5E-16)	0,9416 (4,5E-16)
Fisher	FM	0,7387 (0,010)	0,7267 (0,006)	0,7481 (2,2E-16)	0,7362 (1,1E-16)	0,7362 (1,1E-16)	0,7362 (1,1E-16)	0,7424 (0,002)
	JAC	0,5854 (0,012)	0,5706 (0,008)	0,5938 (4,5E-16)	0,5825 (3,3E-16)	0,5825 (3,3E-16)	0,5825 (3,3E-16)	0,5869 (0,003)
	NMI	0,6545 (0,017)	0,6302 (0,010)	0,7023 (4,5E-16)	0,6551 (2,2E-16)	0,6551 (2,2E-16)	0,6551 (2,2E-16)	0,6883 (0,007)
	RI	0,8234 (0,005)	0,8169 (0,003)	0,8157 (2,2E-16)	0,8236 (5,6E-16)	0,8236 (5,6E-16)	0,8236 (5,6E-16)	0,8128 (0,001)
Glass	FM	0,3636 (0,020)	0,3783 (0,014)	0,3652 (0,033)	0,3551 (2,2E-16)	0,3685 (0,009)	0,3659 (2,8E-16)	0,3921 (0,016)
	JAC	0,2201 (0,015)	0,2320 (0,011)	0,2231 (0,024)	0,2107 (1,4E-16)	0,2225 (0,005)	0,2209 (5,6E-17)	0,2417 (0,013)
	NMI	0,3112 (0,037)	0,3279 (0,030)	0,2541 (0,080)	0,3151 (0)	0,3320 (0,011)	0,3305 (1,1E-16)	0,3667 (0,030)
	RI	0,7041 (0,009)	0,7030 (0,009)	0,6795 (0,028)	0,7171 (1,1E-16)	0,7133 (0,009)	0,7107 (1,1E-16)	0,7147 (0,009)
Haberman	FM	0,5521 (0,001)	0,5516 (0)	0,6074 (0,070)	0,5505 (2,2E-16)	0,5520 (1,1E-16)	0,5514 (0)	0,6922 (2,2E-16)
	JAC	0,3785 (0,001)	0,3782 (0)	0,4383 (0,075)	0,3772 (1,6E-16)	0,3786 (2,2E-16)	0,3781 (0)	0,5293 (2,2E-16)
	NMI	0,0036 (0,002)	0,0041 (0,001)	0,0363 (0,043)	0,0004 (2,7E-19)	0,0066 (1,7E-18)	0,0051 (0,001)	0,0886 (2,8E-17)
	RI	0,5004 (0,001)	0,5001 (0)	0,5467 (0,059)	0,4988 (3,3E-16)	0,5005 (0)	0,4993 (0)	0,6189 (1,1E-16)
Horse	FM	0,4662 (0,034)	0,4510 (0,00)	0,5064 (0,006)	0,4965 (0,004)	0,4406 (2,8E-16)	0,4510 (1,6E-16)	0,5174 (0,033)
	JAC	0,3017 (0,030)	0,3006 (0,021)	0,3389 (0,005)	0,3267 (0,004)	0,2784 (0)	0,2866 (1,6E-16)	0,3479 (0,028)
	NMI	0,1033 (0,025)	0,1031 (0,025)	0,0777 (0,002)	0,1081 (0,006)	0,1225 (4,2E-17)	0,1358 (8,4E-17)	0,1033 (0,033)
	RI	0,5625 (0,023)	0,5625 (0,020)	0,5603 (0,001)	0,5894 (0,005)	0,5531 (1,1E-16)	0,5625 (2,2E-16)	0,5571 (0,033)
Sentetik	FM	0,6549 (0,061)	0,7407 (0,041)	0,8737 (0,083)	0,8216 (2,2E-16)	0,7750 (0,042)	0,8421 (2,2E-16)	0,9569 (0,006)
	JAC	0,4850 (0,066)	0,5826 (0,051)	0,7826 (0,124)	0,6972 (4,5E-16)	0,6326 (0,058)	0,7272 (2,2E-16)	0,9175 (0,011)
	NMI	0,7464 (0,056)	0,8230 (0,031)	0,9169 (0,049)	0,8518 (2,2E-16)	0,8477 (0,024)	0,8853 (5,7E-16)	0,9608 (0,011)
	RI	0,9060 (0,017)	0,9277 (0,013)	0,9637 (0,022)	0,9569 (6,7E-16)	0,9423 (0,012)	0,9617 (3,3E-16)	0,9896 (0,001)
Spect	FM	0,5385 (0,032)	0,5206 (0,004)	0,5708 (0,024)	0,5974 (0,011)	0,5542 (2,2E-16)	0,5575 (4,5E-16)	0,5588 (0,005)
	JAC	0,3688 (0,030)	0,3519 (0,004)	0,3982 (0,022)	0,4214 (0,010)	0,3832 (1,1E-16)	0,3864 (2,2E-16)	0,3877 (0,004)
	NMI	0,0431 (0,044)	0,0201 (0,003)	0,0286 (0,021)	0,0202 (0,002)	0,0434 (3,5E-17)	0,0423 (4,9E-17)	0,0532 (0,007)
	RI	0,5258 (0,031)	0,5060 (0,002)	0,5266 (0,017)	0,5236 (0,002)	0,5356 (1,1E-16)	0,5356 (1,1E-16)	0,5419 (0,004)

Tablo 3. Kümeleme İndeks Sonuçları (Results of Clustering Indexes)

Veri Setleri	İndeks	DE-MSE	PSO-MSE	MKFC	K-medoid	FCM	K-means	DE-MKFC
Thyroid	FM	0,5305 (0,014)	0,5389 (0,005)	0,5773 (2,2E-16)	0,5423 (0,008)	0,5286 (3,3E-16)	0,5258 (2,2E-16)	0,6495 (0,051)
	JAC	0,3508 (0,129)	0,3103 (0,009)	0,4043 (0)	0,3601 (0,007)	0,3482 (1,1E-16)	0,3461 (0)	0,4770 (0,060)
	NMI	0,3023 (0,156)	0,3585 (0,005)	0,3853 (1,3E-16)	0,3148 (0,005)	0,2998 (5,6E-17)	0,2967 (5,6E-17)	0,4587 (0,017)
	RI	0,5765 (0,012)	0,5830 (0,006)	0,5802 (3,3E-16)	0,5895 (0,006)	0,5767 (2,2E-16)	0,5738 (1,1E-16)	0,6668 (0,042)
TAE	FM	0,3661 (0,0217)	0,3662 (0,031)	0,3910 (0,013)	0,4238 (0,006)	0,3606 (0,028)	0,3417 (0,002)	0,4541 (2,2E-16)
	JAC	0,2241 (0,0166)	0,2244 (0,023)	0,2403 (0,008)	0,2688 (0,005)	0,2202 (0,021)	0,2060 (0,002)	0,2817 (5,6E-17)
	NMI	0,0521 (0,028)	0,0542 (0,039)	0,0547 (0,012)	0,1349 (0,0043)	0,0520 (0,037)	0,0273 (0,004)	0,0845 (2,8E-17)
	RI	0,5675 (0,017)	0,5721 (0,019)	0,5366 (0,014)	0,6143 (0,007)	0,5748 (0,018)	0,5640 (0,002)	0,5421 (2,2E-16)
WBCD	FM	0,7940 (0,086)	0,8457 (0,015)	0,8739 (1,1E-16)	0,8533 (5,6E-16)	0,8533 (5,6E-16)	0,8562 (3,3E-16)	0,8743 (0,003)
	JAC	0,6659 (0,111)	0,7329 (0,022)	0,7760 (4,5E-16)	0,7441 (3,3E-16)	0,7441 (3,3E-16)	0,7485 (1,1E-16)	0,7768 (0,005)
	NMI	0,4932 (0,1432)	0,5556 (0,022)	0,6120 (1,1E-16)	0,5712 (4,5E-16)	0,5712 (4,5E-16)	0,5768 (3,3E-16)	0,6130 (0,007)
	RI	0,7840 (0,0928)	0,8372 (0,015)	0,8660 (2,2E-16)	0,8452 (1,1E-16)	0,8452 (1,1E-16)	0,8481 (5,6E-16)	0,8664 (0,003)

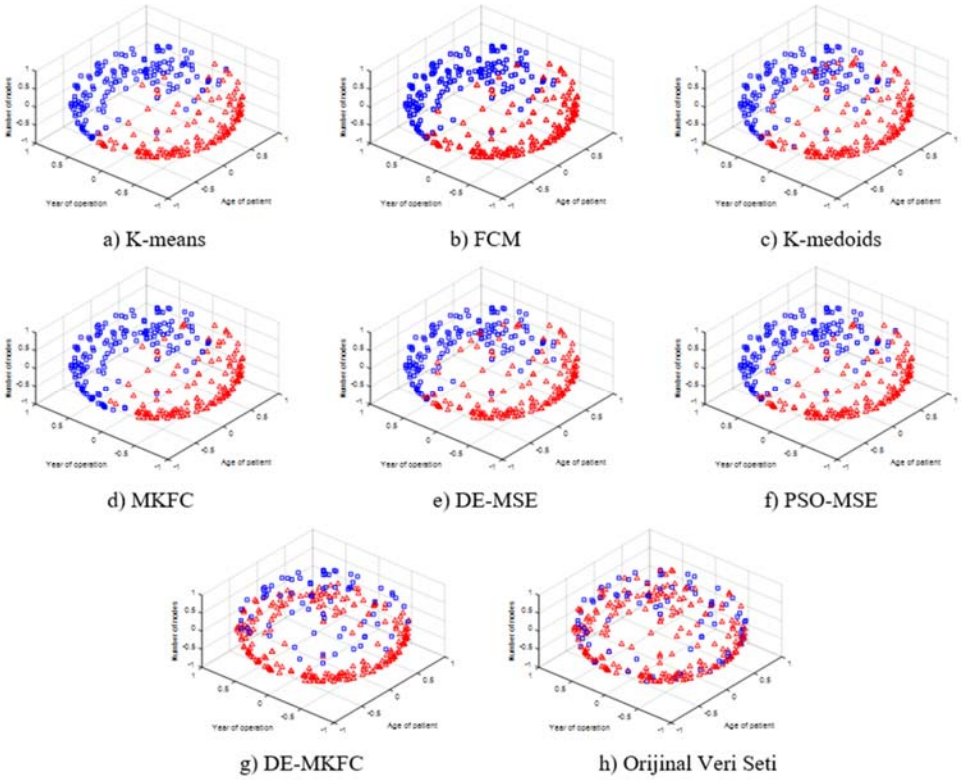
5.2. Görsel Sonuçlar Üzerine Değerlendirme
(Discussions on Visual Results)

Bu bölümde DE-MKFC algoritmasının diğer kümeleme algoritmalarına oranla nasıl daha iyi performans sağladığının anlaşılması için veri elemanlarının kümelere dağılımını ifade eden üç boyutlu grafiklere yer verilmiştir. Görsel değerlendirme için Haberman ve Thyroid veri setleri seçilmiş ve bu veri setlerine ilişkin kümeleme algoritmaları sonuçları orijinal veri seti dağılımlarıyla birlikte sırasıyla Şekil 4 ve 5'te sunulmuştur. Her bir algoritma için Tablo 3'teki ortalama indeks değerine yaklaşan kümeleme etiketleri üç boyutlu modellerin oluşturulmasında kullanılmıştır. Ayrıca 5 özniteliğe sahip Thyroid veri setinin üç boyutlu modele uyarlanabilmesi için bu veri setine ait ilk üç öznitelik kullanılmıştır.

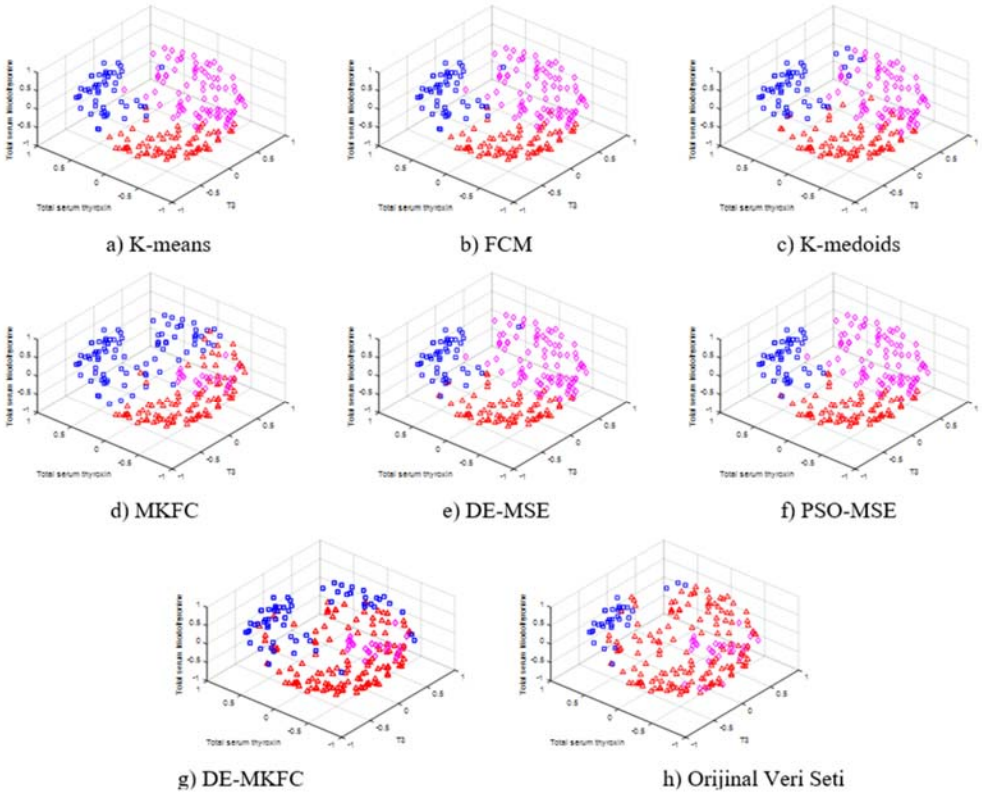
Şekil 4.h'de görüleceği üzere Haberman veri setinde kümeler (orijinal veri etiketleri) homojen bir dağılım göstermemekte ve özellikle üst bölgede farklı kümelerdeki veri elemanlarının iç içe geçtiği görülmektedir. Elde edilen kümeleme sonuçlarına bakıldığında K-means, FCM, K-medoids, DE-MSE ve PSO-MSE algoritmalarının tamamı veri elemanlarını genel olarak yarı yarıya iki yarım küre olacak şekilde kümelere ayırmıştır. Dolayısıyla elde edilen kümeleme dağılımları Şekil 4.h'deki orijinal dağılıma görsel olarak hiç benzerlik göstermemektedir. MKFC algoritmasının kümeleme dağılımı orijinal dağılıma nispeten daha benzemekle beraber, Tablo 3'te görüleceği üzere elde ettiği sonuçlar tutarlılık göstermeyebilmektedir. Bu durum algoritmanın erken yakınsama problemlerine duyarlı olmasından kaynaklanabilmektedir. Önerilen DE-MKFC algoritmasının kümeleme dağılımı ise orijinal veri dağılımına çok yakın bir sonuca ulaşmış ve birbiri ile çakışan kümelerin tespit edilmesinde önemli bir başarı göstermiştir. Şekil 4.h'de Thyroid veri seti elemanlarının kümelere göre olması gereken dağılımı verilmiştir. Kırmızı ile belirtilen veri elemanları genel olarak tüm bölgeye dağılmışken, mavi ile belirtilen veri elemanları daha çok kuzeybatı tarafında bulunmaktadır. Geriye kalan ve pembe ile gösterilen veri elemanları ise daha çok güneydoğu tarafında yayılmış olup kırmızı bölge ile iç içe geçmiş durumdadır. Sonuç olarak kümelerin homojen olarak dağıldığı bir veri setinden söz ötmek mümkün değildir. Bu sebepten ötürü Haberman veri setinde olduğu gibi Thyroid veri setinde de K-means, FCM, K-medoids, DE-MSE ve PSO-MSE gibi sadece yakın komşuluk ilişkisi prensibine dayanan algoritmaların iyi performans sağlayamadığı Şekil 4'te görülmektedir. MKFC algoritması yakın komşuluk ilişkisine dayalı algoritmalara oranla orijinal dağılama benzer bir küme dağılım elde etmiştir. Ancak MKFC'nin elde ettiği dağılımda mavi ile gösterilen bölge biraz fazla olarak iç bölgelere yayılmıştır. Önerilen DE-MKFC algoritması ise MKFC'ye nispeten küme dağılımını daha düzgün olarak yapmıştır.

Tablo 4. Wilcoxon Rank Sum Test Sonuçları (Results of Wilcoxon Rank Sum Test)

Veri Setleri	İndeks	DE-MSE	PSO-MSE	MKFC	K-medoid	FCM	K-means
Breast	FM	+	+	=	+	+	+
	JAC	+	+	=	+	+	+
	NMI	-	-	=	+	-	-
	RI	-	-	=	+	=	=
Fisher	FM	+	+	-	+	+	+
	JAC	+	+	-	+	+	+
	NMI	+	+	-	+	+	+
	RI	-	-	-	-	-	-
Glass	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	+	+	+	-	=	+
Haberman	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	+	+	+	+	+	+
Horse	FM	+	+	+	+	+	+
	JAC	+	+	=	+	+	+
	NMI	=	=	+	=	-	-
	RI	=	=	=	-	=	=
Sentetik	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	+	+	+	+	+	+
Spect	FM	+	+	-	-	+	=
	JAC	+	+	-	-	+	=
	NMI	+	+	+	+	+	+
	RI	+	+	+	+	+	+
Thyroid	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	+	+	+	+	+	+
TAE	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	-	-	+	-	-	-
WBCD	FM	+	+	+	+	+	+
	JAC	+	+	+	+	+	+
	NMI	+	+	+	+	+	+
	RI	+	+	+	+	+	+



Şekil 4. Haberman veri setinin görsel sonuçları (Visual results of Haberman dataset)



Şekil 5. Thyroid veri setinin görsel sonuçları (Visual results of Thyroid dataset)

6. SONUÇ (CONCLUSION)

Bu çalışmanın amacı evrimsel hesaplama tekniği ile dizayn edilmiş bir çoklu kernel kümeleme algoritması geliştirmektir. Bu amaca ulaşmak için çoklu kernel tabanlı amaç fonksiyonu dizayn edilmiş ve bu amaç fonksiyonu bir evrimsel hesaplama tekniği olan diferansiyel gelişim ile entegre edilerek DE-MKFC kümeleme algoritması geliştirilmiştir. Yapılan sayısal ve görsel deneyler sonucunda da DE-MKFC kümeleme algoritmasının bilinen kümeleme algoritmalarından ve çoklu kernel kümelemenin ilk örneği kabul edilen MKFC algoritmasından daha iyi sonuçlar elde ettiği görülmüştür. Ayrıca geliştirilen DE-MKFC algoritması dışında literatürde evrimsel hesaplama tabanlı çoklu kernel kümeleme algoritması mevcut değildir. İleriki aşamada daha iyi amaç fonksiyonları dizayn edilmeye çalışılarak daha etkili çoklu kernel kümeleme algoritmaları geliştirilmeye çalışılacaktır.

KAYNAKLAR (REFERENCES)

1. Köylü M. Yapay Arı Koloni Algoritması Tabanlı Veri Madenciliği Algoritmalarının Geliştirilmesi, Doktora Tezi, Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, 2015.
2. Hancer E., Karaboga D., A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number, Swam and Evolutionary Computation, 32, 49-67, 2017.
3. MacQueen, J., Some methods for classification and analysis of multivariate observations, 5th Berkeley Symp. Math. Stat. Probability, 281–297, 1967.
4. Park H-S, Jun C-H., A simple and fast algorithm for K - medoids clustering, Expert Systems with Applications, 36, 3336-3341, 2009.
5. Bezdek J.C., Pattern Recognition with Fuzzy Objective Function Algoritms. Plenum Press, New York 1981.
6. Omran, M., Particle Swarm Optimization Methods for Pattern Recognition and Image Processing, Ph.D. Thesis, University of Pretoria, Environment and Information Technology, 2004.
7. Kudava P., Clustering Genetic Algorithm, 18th International Workshop on Database and Expert Systems Applications, 138-142, 2007.
8. Var der Merwe D.W., Engelbrecht A.P., Data clustering using particle swarm optimization, IEEE Congress on Evolutionary Computation, 215-220, 2003.
9. Chen G., Luo W. Zhu T., Evolutionary clustering with differential evolution, IEEE Congress on Evolutionary Computation, 1382-1389, 2014.
10. Ozturk C., Hancer E., Karaboga D., Automatic clustering with global best artificial bee colony algorithm, Journal of the Faculty of Engineering and Architecture of Gazi University, 29 (4), 677-687, 2014.
11. Zhao B., Kwok J. T, Zhang C., Multiple Kernel Clustering, SIAM International Conference on Data Mining, 638-649, 2009.
12. Huang H. C., Chuang Y. Y., Chen C. S., Multiple Kernel Fuzzy Clustering, IEEE Transactions on Fuzzy Systems, 20 (1), 120-134, 2012.
13. Wang Y., Liu X., Dou Y., Li R., Multiple Kernel Clustering Framework with Improved Kernels, Twenty-Sixth International Joint Conference on Artificial Intelligence, 2999-3005, 2017.
14. Storn, R., Price, K., Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization, 11 (4), 341-359, 1997.
15. Das, S., Suganthan, P. N., Differential Evolution: A Survey of the State-of-the-Art", IEEE Transactions on Evolutionary Computation, 15 (1), 4-31, 2011.
16. Bulut F., Amasyalı M.F., Classification in mixture of experts using hard clustering and a new gate function, Journal of the Faculty of Engineering and Architecture of Gazi University, 31 (4), 1017-1025, 2016.
17. Al-Zand H.A.R., Bölünmeli Kümeleme Algoritmalarının Farklı Veri Yoğunluklarında Karşılaştırılması, Yüksek Lisans Tezi, Gazi Üniversitesi, Bilişim Enstitüsü, 2013.
18. Liu X., Dou Y, Yin J, Wang L, Zhu E., Multiple Kernel k-Means Clustering with Matrix-Induced Regularization, Thirtieth AAAI Conference on Artificial Intelligence, 1888-1894, 2016.
19. Xia R., Pan Y., Du. L, Yin J., Robust multi-view spectral clustering via low-rank and sparse decomposition, Twenty-Eighth AAAI Conference on Artificial Intelligence, 2149-2155, 2014.
20. Kumar A., Daume H., A Co-training Approach for Multi-view Spectral Clustering, 28th International Conference on International Conference on Machine Learning, 393-400, 2011.
21. Gonen M., Margolin A. A., Localized data fusion for kernel k-means clustering with application to cancer biology, 28th Annual Conference on Neural Information Processing Systems, 1305-1313, 2014.
22. Lichman, M., UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science, 2013.
23. Clustering bechmark datasets. https://cs.joensuu.fi/sipu/datasets. Erişim Tarihi: 23.12.2017.
24. Falco I. D., Cioppa A. D., Tarantino E., Facing classification problems with Particle Swarm Optimization, Applied Soft Computing, 7 (3), 652-658, 2007.
25. Paterlini S., Krink T., Differential evolution and particle swarm optimisation in partitional clustering, Computational Statistics & Data Analysis, 50 (5), 1220-1247, 2006.