Parçacık Sürü Optimizasyonu İle Küme Sayısının Belirlenmesi

Yasin ORTAKCI[1], Cevdet GÖLOĞLU[2]

[1] Karabük Üniversitesi, Bilgisayar Mühendisliği Bölümü, Karabük

[2] Karabük Üniversitesi, Makine Mühendisliği Bölümü, Karabük

yasinortakci@karabuk.edu.tr, cgologlu@karabuk.edu.tr

Özet: Kümeleme problemlerinde kümelenin doğru yapıldığı kadar, küme sayısının tespiti de önemli bir sorundur. Literatürde kümeleme problemlerinin çözümüne yönelik birçok algoritma geliştirilmiştir. Bu algoritmaların çoğunda çözülecek problemdeki küme sayısının önceden bilinmesini gerekmekte ve bu ön bilginin algoritmaya bir parametre olarak girilmesine ihtiyaç duyulmaktadır. Gerçek hayattaki birçok kümeleme probleminde ise veri setine ait küme sayısı önceden bilinmemektedir. Önerilen yöntemde, kümeleme problemi bir optimizasyon problemi olarak ele alınmış ve güçlü bir arama algoritması olan Parçacık Sürü Optimizasyonu ile çözülmeye çalışılmıştır. Bu geliştirilen yöntem ile veri setleri mesafeye dayalı olarak kümelere ayrılmakta ve küme sayısını da tatmin edici oranda doğru bulunmaktadır. Yöntemin kümeleme başarımı, ayrık yapıdaki kümelere sahip yapay bir veri seti ve geçişli kümeleri barındıran zambak çiçeği verileri kullanılarak ölçülmüştür. Özellikle yapay veri seti üzerinde %100'e yakın, zambak verileri üzerinde ise %70'e yakın kümeleme başarımı elde edilmiştir.

Anahtar Sözcükler: Kümeleme, Parçacık Sürü Optimizasyonu, Kümeleme Doğruluk İndeksi.

Determination Of Cluster Numbers Via Particle Swarm Optimization

Abstract: Determination of cluster numbers in clustering problems is crucial as well as the correctness of the clusters. In literature, a number of algorithms for the solution of clustering problems has been developed. In most of them, the cluster numbers should be known as an input parameter to the algorithm before starting to solution. However, most of the clustering problems in real life, the number of clusters that belongs to the dataset is not known initially. In the proposed method, the clustering problem is accepted as an optimization problem and it is solved with Particle Swarm Optimization which is a powerful search algorithm. By the method developed, datasets are separated to clusters based on distances and the method is able to find the number of clusters in satisfactory rates. The performance of clustering method has been measured by using an artificial dataset with discrete scatter and using iris dataset with nested scatter. The clustering success rates obtained are almost 100% at artificial dataset while over 70% at iris dataset.

Keywords: Clustering, Particle Swarm Optimization, Clustering Validity Index.

1.Giriş

Kümeleme birçok bilim dalında kullanılan ve çözümüne yönelik birçok algoritmanın geliştirildiği makine öğrenme yöntemidir. Kümeleme problemlerin çözümünde kullanılan birçok geleneksel algoritma yerel minimumlara takılma riski taşımaktadır. Bu problemin üstesinden gelebilmek için Parçacık Sürü Optimizasyonu (PSO) gibi evrimsel arama algoritmaları kullanılabilir [1, 2, 3]. Ayrıca literaturde yapılan çalışmalara bakıldığında bir çok kümeleme probleminde küme sayısı bir ön bilgi olarak bilinmesi gerekmektedir. Gerçek hayatta karşılaşılan bir çok problemde ise veri topluluğunun kaç farklı kümeye ayrılacağı önceden bilinemez. Bu sebeple kullanılan kümeleme algoritmasının verileri doğru gruplamasının yanı sıra oluşacak muhtemel küme sayısını da doğru tahmin etmesi gerekir.

Yapılan bu çalışmada da PSO bir kümeleme algoritması olarak kullanılmış ve doğru küme sayısının yanında en uygun kümeleme seçeneği bulunmaya çalışılmıştır. Uygulamada PSO'nun algoritmasında herhangi bir değişiklik yapılmadan sadece optimize edilecek parametrelere küme merkezlerinin yanı sıra küme sayısı da eklenmiştir. Çalışmanın devamında ikinci bölümde kümeleme konusunda genel bilgiler verilmiş, üçüncü bölümde PSO'nun yapısı ve çalışma mantığından bahsedilmiş, dördüncü bölümde kümeleme problemlerinin PSO'ya nasıl uygulanacağı anlatılmış, geriye kalan bölümlerde de yapılan deneysel çalışmalar ve sonuçlarından bahsedilmiştir.

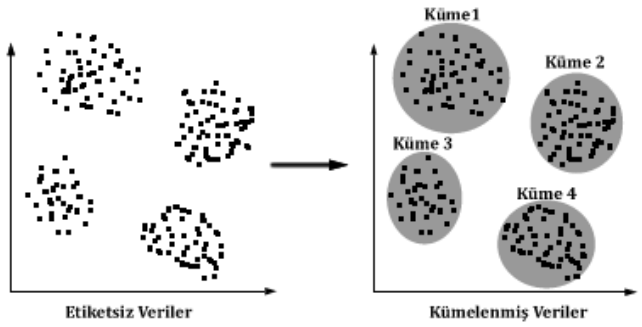
2.Kümeleme

Kümeleme, vasıfsız veri topluluklarının herhangi bir öğreticiye ihtiyaç duymadan farklı gruplara ayrılmasıdır. Bu gruplar kendi içinde benzer özelliklere sahip veriler içerirken, oluşturulan her bir veri grubu da diğer gruplardan farklı özellik gösterirler. Kümelemenin amacı n tane elemandan oluşan bir veri setini farklı özelliklere sahip k tane veri kümesine bölmektir. Bu ayrıştırma işlemi yapılırken k tane kümenin birbirinden maksimum derecede farklılık göstermesi ve kendi içlerinde ise maksimum derecede benzerlik göstermesi istenir [1,4,5]. Şekil 1'de bir çok sayıda elemandan oluşan bir veri seti mesafeye dayalı benzerlik ölçütüne göre dört farklı kümeye ayrılmıştır.

Bir veri topluluğu kümelere ayırırken birçok farklı kümeleme seçeneği oluşturulabilir. Bu kümeleme seçenekleri bazı kriterlere göre değerlendirilip en iyi kümeleme seçeneğini bulmak bir optimizasyon problemi olarak ele alınabilir. Amaç verilerin ayrılacağı grup sayısını ve her bir kümenin ağırlık merkezini bulmaktır.

Verilerin gruplandırılması sırasında üç temel nokta ele alınır:

- Verilerin gruplandırılacağı optimum küme sayısının belirlenmelidir. Verilerin doğru gruplandırılmasının yanı sıra grup sayısının belirlemesi de kümeleme işleminde değerlendirmeye alınır.



Şekil 1. Veri kümeleri

- Verilerin gruplandırılması sırasında kullanılacak benzerlik ölçütünün belirlenmesidir. Bu benzerlik ölçütü aynı küme içerisndeki verilerin maksimum oranda benzerlik göstermesini sağlarken diğer gruptaki verilerle maksimum oranda farklılık göstermesini sağlamalıdır. En yaygın olarak kullanılan benzerlik ölçütü mesafeye dayalı benzerliktir.

- Kümeleme işlemine en hızlı şekilde gerçekleştirecek yöntemin belirlenmesidir.

Farklı kümelere ayrılacak n tane elemandan oluşan veri setini $P = \{\overrightarrow{P_1}, \overrightarrow{P_2}, \overrightarrow{P_3}, \dots, \overrightarrow{P_n}\}$ şeklinde ifade edelim. Veri setindeki her bir elemanı temsil eden P_i, d boyutlu bir vektördür. $\overrightarrow{P_i} = \{p_{i,1}, p_{i,2}, p_{i,j}, \dots, p_{i,d}\}$ ve $p_{i,j}$ değeri veri setindeki $i.$ noktanın $j.$ boyutundaki gerçek değeri ifade eder. $i = 1,2,\dots,n$ ve $j = 1,2,\dots,d$'dir.

Kümeleme algoritmaları bu veri setindeki bütün noktaları k farklı sınıfa ayırarak, $C = \{C_1, C_2, \dots, C_k\}$ gibi bir sınıflandırma yapar. Kümeleme işlemi sonucu oluşan her bir grubu diğer gruplardan mümkün olduğunca farklı, kendi içersindeki elemanların da mümkün olduğunca benzer olması gerekir. Sınıflandırma işlemi sonucu oluşan kümelerin şu özellikleri taşıması gerekir [2]:

- Her bir kümeye en az bir eleman atanmalıdır.
  $C_i \neq \emptyset \; ve \; \forall \; i \in \{1,2,..k\}$

- İki farklı kümenin ortak elemanı olmamalıdır.
  $C_i \cap C_j = \emptyset \; ve \; \forall \; i \neq j \; ve \; i,j \in \{1,2,..k\}$

- Her bir veri bir kümeye atanmalıdır.
  $\cup_{i=1}^{k} C_k = P$

Kümeleme işleminin sırasında veri setindeki iki eleman arasındaki benzerlik miktarı genel olarak öklit mesafesi kullanılarak belirlenir. Örneğin $\overrightarrow{P_i}$ ve $\overrightarrow{P_j}$ gibi d boyutlu iki örnek eleman arasındaki benzerlik miktarı şu şekilde hesaplanır [4]:

$$d(\overrightarrow{P_i}, \overrightarrow{P_j}) = \sqrt{\sum_{dim=1}^{d} \left(p_{i,dim} - p_{j,dim}\right)^2} = \left\|\overrightarrow{P_i}, \overrightarrow{P_j}\right\| \qquad (1)$$

3. Parçacık Sürü Optimizasyonu

Parçacık Sürü Optimizasyonu (PSO) 1995 yılında Russell Eberhart ve James Kennedy tarafından bulunmuş populasyon tabanlı, evrimsel bir optimizasyon algoritmasıdır [6]. Kuş ve balık sürülerinin yiyecek arayışları ve tehlikeden kaçışları esnasındaki toplu hareketlerinden esinlenerek modellenmiştir. PSO hızlı sonuç bulması, az parametre gerektirmesi ve yerel optimumlara takılma riskinin az olması sebebiyle diğer birçok arama algoritmasına üstünlük kurmuştur.

PSO, parçacık adı verilen ve her biri ilgili probleme farklı bir çözüm önerisi getiren elemanlardan oluşur. Bu parçacık topluluğuna sürü adı verilir. Sürüdeki bütün parçacıklar çözüm uzayında rastgele değerler alarak arama işlemine başlarlar. Her bir parçacık konum *(X)* ve hız *(V)* vektörü olmak üzere iki vektörel bileşene sahiptir. Pozisyon vektörü parçacığın konum bilgisini, hız vektörü ise parçacığın konum değiştirme miktarı ve yön bilgisini tutar. İteratif bir algoritma olan PSO'nun her bir iterasyonunda parçacıkların hız bileşenleri dolayısı ile konum bileşenleri güncellenir. Bir parçacığın yeni hız vektörü daha önceki iterasyonlarda elde ettiği tecrübeden, sürünün genel tecrübesinden ve rastgelelikten faydalanılarak Eşitlik 2'ye göre hesaplanır.

$$V_{ij}^{t+1} = wV_{ij}^{t} + c_1 r_1 (P_{ij} - x_{ij}) + c_2 r_2 (G_{gbest} - x_{ij}) \qquad (2)$$

t : İterasyon sayısı;
c_1, c_2 : Hızlandırma katsayıları;
r_1, r_2 : [0-1] aralığındaki rastgele üretilen sayılar;
w : Atalet ağırlığı;
P_{ij} : Parçacığın yerel en iyi değeri;
G_{gbest}: Sürünün en iyi değeri;

P_{ij}, parçacığın son iterasyonda bulduğu en iyi konum değeridir ve yerel en iyi değer olarak adlandırılır. G_{gbest}, ise sürüdeki o ana kadar bulunmuş en iyi konum değeridir ve global en iyi değer olarak adlandırılır. Parçacığın yeni hız değeri, bir önceki hız değeri ile yerel en iyi değerinden ve global en iyi değerinden faydalanarak bulunur [7, 8].

Yeni konum vektörü ise Eşitlik 3'de görüldüğü gibi eski konum vektörüne yeni konum değeri eklenerek hesaplanır.

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \qquad (3)$$

Parçacıkların konum değerleri ele alınarak uygunluk fonksiyonunda parçacıkların sundukları çözümlerin kaliteleri belirlenir. Uygunluk fonksiyonu parçacıkların konum değerlerini giriş parametresi olarak alan ve çıktı olarak sayısal değerler üreten bir değerlendirme fonksiyonudur. Minimizasyon problemlerinde uygunluk değerleri küçük olan parçacıklar büyük olanlara tercih edilirken, maksimizasyon problemlerinde uygunluk değeri büyük olan parçacıklar küçük olana tercih edilir.

Sonuç olarak PSO'da her bir parçacık rastgele konum ve hız değerleri ile aramaya başlatılır.Her bir iterasyonda hız ve konum değerleri güncelenip uygunluk fonksiyonunda uygunluk değeri üretilir. Yine her bir iterasyonda parçacıkların yerel en iyi değeri ve sürünün global en iyi değeri güncelenir. Belirli bir iterasyon sonucunda sürünün en iyi değeri PSO'nun probleme sunduğu çözüm olur [9].

PSO Algoritması:

BEGIN
pop Tane Parçacığa Hız ve Konum Değerleri Ata
REPEAT
  FOR i=1 **TO** pop
    Uygunluk Değerini Hesapla;
    P_{best} Değerini Güncelle;
    g_{best} Değerinin Güncelle;
    Hız ve Pozisyon Değerlerini Güncelle;
  END FOR
UNTIL Sonlandırma Kriteri
END

---

4. PSO İle Kümeleme

Daha öncede belirtildiği gibi PSO'daki her bir parçacık problem için muhtemel bir çözüm önerisidir. Kümeleme problemlerinde de sürüdeki her bir parçacık küme merkezlerini temsil eder ve en uygun küme merkezlerini bulmaya çalışırlar. Bulunan bu merkezlere olan uzaklıklarına göre veri setindeki bütün elemanlar en yakın kümeye atanarak en uygun kümeleme seçeneği bulunmaya çalışılır.

Sürünün başlangıç populasyonunu X; $X = \{X_1, \dots, X_i, \dots, X_{pop}\}$; ve pop, sürüdeki parçacık sayısı; şeklinde ifade edilsin. Kümeleme işlemleri sırasında da her bir parçacık oluşturulan kümelerin merkezini temsil eden birer vektördür. Her bir parçacık, $X_i = \{\overrightarrow{o_1}, \overrightarrow{o_2}, \dots, \overrightarrow{o_j}, \dots, \overrightarrow{o_k}\}$ ve k, oluşturulan küme sayısı, o_j, j. kümenin ağırlık merkezi şeklinde ifade edilsin. Her bir kümenin ağırlık merkezi $\overrightarrow{o_j} = \{o_{j,1}, o_{j,2}, \dots, o_{j,d}\}$ d boyutlu bir vektördür. Ağırlık merkezlerinin boyutu veri setindeki kümelecek her biri elemanın boyutuna eşittir. Her bir boyutta gerçek değerler vardır.

Küme sayısının önceden bilinmediği kümeleme problemlerinde doğru kümelemeyi bulma gibi veri setinin kaç kümeye ayrılacağı da diğer bir optimizasyon konusudur. Yapılan çalışmada bu iki birbirine bağlı optimizasyon problemini için PSO'da farklı parçacıklar kullanılmayacaktır. Bunun yerine her iki probleme de çözüm önerisi aynı parçacık üzerinden getirilecektir. Yani bir parçacık hem verilerin kaç ayrı kümeye ayrılacağı konusunda hem de kümelemenin en doğru şekilde yapılması konusunda çözüm üretecektir [9]. Buna göre sürüdeki i.parçacık;

$$X_i = \left\{\overrightarrow{o_1}, \overrightarrow{o_2}, \dots, \overrightarrow{o_j}, \dots, \overrightarrow{o_k}, z_1, z_2, \dots, z_j, \dots, z_k\right\}$$

şeklinde bir yapıya sahip olacaktır. Burada $\vec{o}$ küme merkezleri vektörü, $\vec{z}$ ise değerlendirme vektörüdür. Yani bir parçacık hem küme merkezleri vektörü, hem de değerlendirme vektörünü değerlerini tutmaktadır. Gösterimdeki k sayısı kullanıcı tarafından girilen

olabilecek maksimum küme sayısıdır. z_j ise [0,1] aralığında değerler alabilen ve j. kümenin aktif olup olmadığını gösteren bir değerdir.

EĞER $z_j \geq 0{,}5$ ise
j. kümenin ağırlık merkezi **AKTİF**
DEĞİLSE (Yani $z_j < 0{,}5$ ise)
j. kümenin ağırlık merkezi **PASİFTİR**.

Yani z_j belirtilen eşik değerine eşit ya da eşik değerinden daha büyük ise ise j. küme aktif, eşik değerinden küçük ise pasif durumdadır. Örneğin değerlendirme vektörü;

z= {0,45; 0,63; 0,37; 0,91; 0,23; 0,11; 0,88; 0,20; 0,54; 0,17}

şeklinde olan bir parçacık için, eşik değerini 0,5 olarak alınırsa küme merkezleri vektörünün ($\vec{o}$) 2., 4., 7., 9. elemanları aktiftir. Bu durumda i. parçacığın dikkate alınması gereken hali;

$$X_i = \{\overrightarrow{o_2}, \overrightarrow{o_4}, \overrightarrow{o_7}, \overrightarrow{o_9}, z_2, z_4, z_7, z_9\}$$

şeklinde olacaktır. Sonuç olarak d boyutlu, en fazla k kümeden oluşan i. parçacığın temsili gösterimi,

$$X_i = \left\{ o^i_{1,dim}, o^i_{2,dim}, \dots, o^i_{k,dim}, z^i_1, z^i_2, \dots, z^i_k \right\};$$

i :1,2,…,p;
dim :1,2,…,d;
d :Veri setindeki herbir noktanın boyutu;
p :Sürüdeki parçacık sayısı;
k :Maksimum küme sayısı;

şeklinde olacaktır. Bu formülasyona göre bir parçacık $(k \times d + k)$ tane gerçek değer içeren haneden oluşacaktır. Aynı şekilde parçacığın hız vektörü de konum vektörü ile aynı yapıda olacaktır [1].

Her bir parçacığın ürettiği farklı kümeleme çözümünün kalitesinin belirlenmesi için bir değerlendirme fonksiyonuna ihtiyaç vardır. Kümeleme Doğruluk İndeksi (KDİ) olarak adlandırılan bu değerlendirme fonksiyonu, kümeleme seçeneklerinin sayısal tabana dayandırılarak uygunluğunun değerlendirildiği fonksiyondur. Bu çalışmada kullanılan Kümeleme Doğruluk İndeksi [1, 10]:

$$\text{KDİ} = \frac{\frac{1}{k}\sum_{j=1}^{k}\left\{\frac{1}{|c_j|}\sum_{\forall p_i \in C_j} max_{\forall p_k \in C_j}\{d(p_i,p_k)\}\right\}}{\frac{1}{k}\sum_{j=1}^{k}\{min_{t\in k \vee t\neq j}\{d(o_j,o_t)\}\}} \quad (4)$$

sadeleştirirsek,

$$\text{KDİ} = \frac{\sum_{j=1}^{k}\left\{\frac{1}{|c_j|}\sum_{\forall p_i \in C_j} max_{\forall p_k \in C_j}\{d(p_i,p_k)\}\right\}}{\sum_{j=1}^{k}\left\{min_{t\in k \vee t\neq j}\{d(o_j,o_t)\}\right\}} \quad (5)$$

şeklindedir.

$$o_j = \frac{1}{|C_j|}\sum_{p_i \in C_j} p_i \;\; j = 1,2,\dots,k \quad (6)$$

k :Toplam küme sayısı;
C_j :j. Küme;
$|C_j|$:j. kümedeki toplam eleman sayısı;

5. Uygulama

Bu çalışmada, PSO kümeleme uygulaması olarak önce üç boyutlu noktalar içeren ve üç kümeden oluşan 120 elemanlı yapay bir veri seti, ardından kümeleme algoritmalarını test için birçok çalışmada kullanılan zambak çiçeği (iris flower) veri seti kümelenecektir. Zambak verisi 150 noktadan oluşan dört boyutlu bir veri setidir. Bu veri setinin her bir elemanı, çiçeğinin santimetre cinsinden çanak yaprak uzunluğu, çanak yaprak genişliği, ayakçak yaprak uzunluğu ve ayakçak yaprak genişliğini olmak üzere dört farklı değeri içeren dört boyutlu bir değişkendir. Zambak veri seti elemanları üç farklı kümeden oluşmaktadır. Yapay veri setinin ve zambak verilerinin kümelere göre dağılımı ve eleman sayıları Tablo 1'de verilmiştir.

Veri Adı	Küme Adı	Kümenin Eleman Sayısı
Yapay Veri	1.Küme	40
	2.Küme	30
	3.Küme	50
Zambak Verisi	Iris Setosa	50
	Virginica	50
	Versicolor	50

Tablo 1. Yapay veri ve Zambak veri setlerinin kümelerinin eleman sayıları

Kümeleme işlemi için geliştirilen kod Microsoft Visual Studio 2008 C# ortamında geliştirilmiştir. Geliştirilen bu kod Window 7 Professional 64 bit işletim sistemin sahip Intel Core2 Duo 2,63GHz işlemcili ve 4 GB RAM'li bir dizüstü bilgisayar üzerinde çalıştırılmıştır. Uygulamada kullanılacak parametrik değerler; hızlandırma katsayıları $c_1 = 1{,}49$, $c_2 = 1{,}49$, atalet ağırlığı $w = 0{,}72$, maksimum iterasyon sayısı $t_{max} = 1000$, parçacık sayısı $p = 20$, hız güncelleme limiti V_{max} değer aralığının %1'idir. Elde edilen sonuçlar programın her bir yöntem için 20'şer defa birbirinden bağımsız çalıştırılması ile elde edilmiştir.

Hazırlanan programın 20 defa çalıştırılması sonucunda yapay veri setinin küme sayısı 19 kez 3, 1 kez de 5 olarak bulunmuştur. Elde edilen kümeleme sonuçlarının %95'inde gerçek küme sayısı olan 3 değeri bulunmuştur. Yine zambak veri setinin küme sayısı 14 kez 3, 6 kez 2 bulunmuştur. Zambak için elde edilen kümeleme sonuçlarının %70'inde zambak verisinin gerçek küme sayısı olan 3 değeri bulunmuştur. Kümeleme başarımı ise doğru küme sayısının bulunduğu denemeler üzerinden ölçülmüş ve yapay verinin kümeleme başarımı %100, zambak verisinin ise %68 olarak bulunmuştur. Kümeleme başarımı doğru kümeye ayrılan nokta sayısının toplam nokta sayısına oranlanması ile elde edilmiştir. Elde edilen sonuçlar Tablo 2'de gösterilmiştir.

Veri Seti	Programın Çalıştırılma Sayısı	Bulunan Küme Sayısı	Küme Sayısını Bulma Başarımı	KDİ Değerleri	Kümeleme Başarımı
Yapay Veri	19	3	%95	0,664	%100
	1	5		0,698	
Zambak Verisi	14	3	%70	0,595	%68
	6	2		0,626	

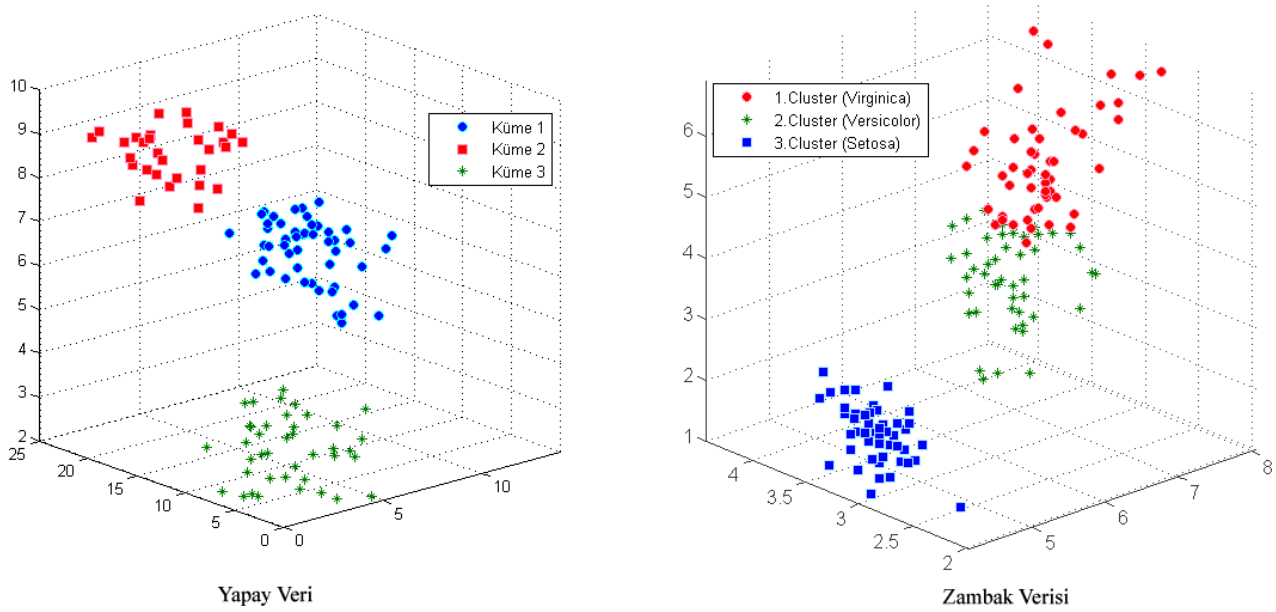
Tablo 2. Uygulamanın çalıştırması sonucu elde edilen değerler

6.Sonuç ve Öneriler

Yapılan çalışmanın sonucunda kullanılan yöntemin kümeleme problemlerinde küme sayısını bulma konusunda başarılı olduğu görülmektedir. Özellikle yapay veri seti için %95'lik bir başarım elde edilmiştir. Zambak veri seti için ise %70'lik bir başarım söz konusudur. Bu sonuçlar da kullanılan yöntemin küme sayısı önceden bilinmeyen her hangi bir veri setinin küme sayısını büyük bir oranda doğru tahmin edeceğini göstermektedir. Ayrıca bu yöntemin veri setinin doğru kümelere ayrılması yani kümeleme başarımı konusunda yapay veri için %100'lük, zambak verisi için %68'lik başarı ortaya çıkmaktadır.

Hem küme sayısının bulunması hem de doğru kümeleme yapılabilmesi konusunda uygulanan yöntemin yapay veri seti üzerinde zambak veri setinden daha iyi sonuçlar verdiği görülmektedir. Bunun nedeni yapay veri seti ile zambak veri setinin noktalarının bulundukları uzaydaki dağılımıdır. Şekil 2'de yapay veri setinin ve zambak veri setinin noktalarının üç boyutlu düzlem de dağılımı ayrı ayrı gösterilmektedir. Zambak veri setindeki noktalar aslında dört boyutludur. Fakat Şekil 2 çizdirilirken üç boyutu ele alınarak çizim işlemi gerçekleştirilmiştir. Şekile bakıldığında yapay verinin kümelerinin birbirinden ayrık bir yapıda olduğu, zambak verisinin ise kümelerinin iç-içe girdiği görülmektedir. Zambak verisi üç boyutlu düzlemde insan gözüyle bile ayırt edilemeyecek bir kümeleme yapısına sahiptir ve bu ölçüde zor bir kümeleme problemidir. Oysa kümeleme algoritmalarında genellikle noktalar mesafeye dayalı olarak kümelere ayrılmaktadır. Bu uygulamada da kullanılan formüller verileri mesafeye dayalı olarak kümelere ayırmaktadır.

Sonuç olarak uygulanan yöntem ayrık yapıda olan yapay verinin küme sayısını %95 oranında, iç içe girmiş kümelere sahip zambak verisinin küme sayısını %70 oranında doğru tahmin etmektedir. Bu sonuçlar tatmin edici sonuçlar olup bu yöntemin kümeleme problemlerinde kullanabileceği kanısına varılmıştır.



Şekil 2. Yapay veri seti ve zambak verilerin 3 boyutlu düzlemde dağılımı

Kaynaklar

[1] Chen, C. Y., Feng H. M. and Ye F., "Automatic particle swarm optimization clustering algorithm", ***International Journal Of Electrical Engineering***, 13 (4): 379-387 (2005).

[2] Das, S., Abraham, A. and Konar, A., "Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm", ***Science Direct Pattern Recognition Letters***, 29 (5): 689-699 (2008).

[3] Premalatha, K. and Natarajan, A. M. ,"A new approach for data clustering based on pso with local search", ***Computer and Information Science***, 1 (4): 139-145 (2008).

[4] Das S., Abraham, A. and Konar, A., "Automatic clustering using an improved differential evolution algorithm", ***Systems Man and Cybernetics Part A: Systems and Humans IEEE Transactions on***, 38 (1): 218-237 (2008).

[5] Ortakcı, Y., Göloğlu C., "Comparison of M-FPSO and C-FPSO Hybrid Methods in Clustering", ***FUZZYSS'11 The Second International Fuzzy Systems Symposium***, Hacettepe University, November 17-18, 7-10 (2011).

[6] Kennedy, J. and Eberhart, R., "Particle swarm optimization.", ***Neural Networks Proceedings IEEE International Conference on***, Perth, 1942-1948 (1995).

[7] Eberhart, R. and Kennedy, J., "A new optimizer using particle swarm theory", ***In: Sixth International Symposium on Micro Machine and Human Science***, Nagoya, 39-43 (1995).

[8] Shi, Y. and Eberhart, R., "Empirical study of particle swarm optimization", ***Proc. of the Congress on Evolutionary Computation***, Washington, 3: 1945-1950 (1999).

[9] Ortakcı Y., Parçacık Sürü Optimizasyonu Yöntemlerinin Uygulamalarla Karşılaştırılması, Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü, Karabük Üniversitesi, (2011).

[10] Omran, M. G. H., Engelbrecht A. P. and Salman A., "Dynamic clustering using particle swarm optimization with application in unsupervised image classification", ***World Academy of Science***, 199-204 (2005).