Journal of the Faculty of Engineering and Architecture of Gazi University 35:1 (2020) 337-354

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

Kd-tree and adaptive radius (KD-AR Stream) based real-time data stream clustering

Ali Şenol[1*], Hacer Karacan[2]

[1]Department of Computer Engineering, Gazi University, Ankara, 06570, Turkey
[2]Department of Computer Engineering, Ardahan University, Ardahan, 75002, Turkey

Highlights:

- Fully online data stream clustering
- Evolutionary based clustering
- Adaptive radius
- Time based summarization
- Memory for past status of clusters

Keywords:

- Data stream clustering
- Kd-Tree
- Adaptive radius
- Fully online
- Evolving clustering

Article Info:

Research Article
Received: 04.10.2018
Accepted: 18.05.2019

DOI:

10.17341/gazimmfd.467226

Correspondence:

Author: Ali Şenol
e-mail: alisenol@gazi.edu.tr
phone: +90 (478) 211 7531

Graphical/Tabular Abstract

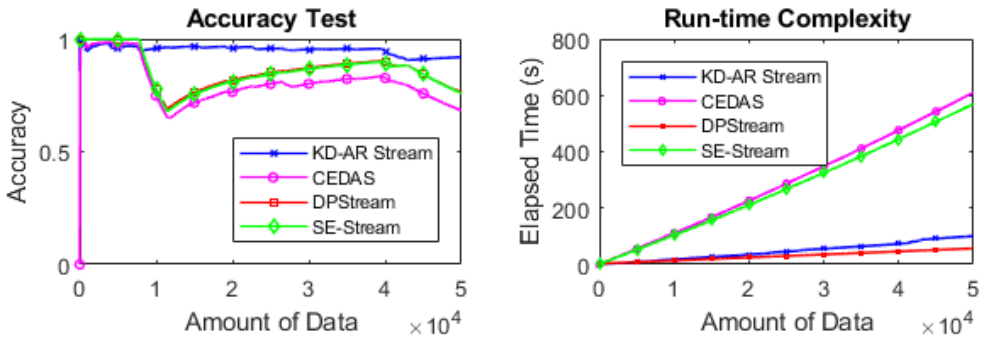


Figure A. Comparision of clustering quality and run-time complexity of algorithms on KDD dataset

Purpose: The aim of this article to propose a new data stream clustering algorithm, which has an adaptive radius, can adapt itself to the evolutionary structure of streaming data and works in a fully online manner.

Theory and Methods:
In this study, kd-tree is used to forming and splitting clusters, adaptive radius approach is used to support increasing and decreasing the size of clusters, active/inactive status of clusters is used to adapt to the evolutionary structure of streaming data and all the operations are done online. In order to create a new cluster, the data that does not belong to any cluster are placed in a kd-tree, and the *rangesearch* operation is performed on those data according to predefined variables r (the radius of candidate cluster) and N (the number of data must be in the area). After forming the clusters, the radius of each cluster could be increased or decreased over time if necessary. Some clusters may be split and some may be merged over time because of dynamically changing structure of streaming data. Inactivation and reactivation of the status of clusters is used to allow for the identification of clusters formed in the same region at a different time interval with same cluster labels in accordance with the nature of the streaming data contrary to literature. This feature increases clustering quality of the proposed method. A summarization method that consist of time window and sliding window is used to support time based summarization without reduce performance.

Results:
To verify the effectiveness of KD-AR Stream algorithm, it is compared with SE-Stream, DPStream, and CEDAS on a variety of well-known datasets in terms of clustering quality and run-time complexity. The results show that KD-AR Stream outperforms other algorithms with a higher clustering success in a reasonable time as shown in Fig. A.

Conclusion:
The aim of this study is to propose a novel data stream clustering algorithm that adapts to the dynamic structure of the streaming data. The aim achieved by using the five evolutionary process which are appearance, activation/inactivation, self-evolution, merge, and split. According to the results, the proposed method is very successful in terms of clustering quality and run-time complexity.

Journal of the Faculty of Engineering and Architecture of Gazi University 35:1 (2020) 337-354

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

K-boyutlu ağaç ve uyarlanabilir yarıçap (KD-AR Stream) tabanlı gerçek zamanlı akan veri kümeleme

Ali Şenol[1]*, Hacer Karacan[2]

[1]Gazi Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 06570 Maltepe Ankara, Türkiye
[2]Ardahan Üniversitesi Yenisey Kampüsü, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 75002, Ardahan, Türkiye

ÖNEÇIKANLAR

- Tamamen çevrimiçi çalışan bir akan veri kümeleme yaklaşımı
- Evrimsel kümeleme yeteneğine sahip kümeleme
- Uyarlanabilir yarıçap özelliği
- Literatür çalışmalarının tersine zaman tabanlı özetleme yaklaşımı
- Kümelerin geçmişini tutma özelliği

Makale Bilgileri

Araştırma Makalesi
Geliş: 04.10.2018
Kabul: 18.05.2019

DOI:
10.17341/gazimmfd.467226

Anahtar Kelimeler:
Akan veri kümeleme,
k-boyutlu ağaç,
uyarlanabilir yarıçap,
tamamen çevrimiçi,
evrimsel kümeleme

ÖZET

Akan veri kümeleme, teknolojik gelişmelere paralel olarak veri miktarının inanılmaz boyutlara ulaştığı gümünüzün popüler konularından biridir. Akan veri kümeleme yaklaşımlarında karşılaşılan en önemli problemler çoğu yaklaşımın çevrimiçi ve çevrimdışı evreden oluşması, küme sayısını tanımlama veya bu sayıya bir sınır koyma zorunluluğu, en doğru yarıçap değerini belirlemede yaşanan problemler ve önerilen modellerin kendisini gelen yeni verilere adapte etmesinde (concept evolution) yaşanan problemlerdir. Bu problemlerin yanında, neredeyse bu alandaki bütün çalışmaların sayısal miktar tabanlı bir özetleme yapması da bazı uygulamalar için ihtiyacı karşılamamaktadır. Oysa son 1 saniyede veya son 1 saatte gelen veriler şeklinde çalışan zaman tabanlı bir özetleme yaklaşımına da ihtiyaç vardır. Bu çalışmada, K-boyutlu ağaç, uyarlanabilir yarıçap tabanlı (KD-AR Stream) ve kümeleme adaptasyonu özelliğine sahip gerçek zamanlı akan verileri kümeleyen bir yaklaşım önerilmektedir. Önerdiğimiz yöntem SE-Stream, DPStream ve CEDAS algoritmaları ile hem kümeleme başarısı hem de işlem performansı açısından karşılaştırılmıştır. Elde edilen sonuçlar KD-AR Stream algoritmasının diğer algoritmalara göre yüksek bir kümeleme başarısını makul bir sürede gerçekleştirdiğini göstermektedir.

Kd-tree and adaptive radius (KD-AR Stream) based real-time data stream clustering

HIGHLIGHTS

- A fully online data stream clustering approach
- Evolutionary based clustering
- Adaptive radius
- A time based summarization approach, contrary to literature
- Memory for past status of clusters

Article Info

Research Article
Received: 04.10.2018
Accepted: 18.05.2019

DOI:
10.17341/gazimmfd.467226

Keywords:
Data stream clustering,
kd-tree,
adaptive radius,
fully online,
evolving clustering

ABSTRACT

Data stream clustering is one of the most popular topics of today's world where the amount of data reaches incredible levels in parallel with technological developments. The most important problems encountered in data stream clustering approaches are the fact that most of the approaches consists of an online and offline phases, the definition of the number of cluster, or the need to set a limitation to this number, the problems encountered in determining optimum radius value, and the problems encountered in concept evolution. The present study proposes an evolutionary based solution method, which is based on Kd-Tree and adaptive radius (KD-AR Stream) to perform real-time clustering on the streaming data. The proposed approach has been compared with SE-Stream, DPStream and CEDAS algorithms in terms of both cluster quality and execution time. The results showed that KD-AR Stream algorithm has a good clustering performance within a reasonable time by comparison with the other algorithms.

*Sorumlu Yazar/Corresponding Author: alisenol@gazi.edu.tr / Tel: +90 478 211 7531

1. GİRİŞ (INTRODUCTION)

Günümüzde kullanılan akıllı telefonlar, tabletler ve sensörler gibi teknolojik cihazlar ile Twitter, Instagram, Facebook gibi sosyal ağlar bilgisayar ortamına aktarılan veri miktarında çok önemli bir artışa neden olmaktadır [1-5]. Veri miktarının bu kadar arttığı bir ortamda artık klasik veri işleme yöntemleri yetersiz kalmaktadır. Bu nedenle yeni veri işleme yöntemlerine ihtiyaç duyulmaktadır. Akan veri kümeleme yaklaşımı söz konusu bu veriyi değerli bilgiye çevirme konusunda pek çok açıdan ihtiyacı karşılamaktadır. Akan veri kümeleme tıklama verisi [6], saldırı tespit sistemleri [7-9], finansal uygulamalar [10], bilimsel araştırmalar [11], sağlık araştırmaları [12-14], nesnelerin interneti (IoT) [15] ve mobil uygulamalar [16] gibi pek çok alanda kullanılmaktadır [17-19].

Akan veri, miktar olarak çok büyük, sonsuz ve devamlıdır [20, 21]. Veri sürekli değiştiğinden veri hakkında kestirimde bulunmak zordur. Bu nedenle küme sayısı ve yarıçap gibi parametreleri doğru bir şekilde belirlemek de güçtür. Dolayısıyla geliştirilecek yöntemlerin ya bu değerleri otomatik belirlemesi ya da esnek bir yapıda olması gerekir. Tablo 1'de de görüldüğü gibi, akan veri kümeleme alanında karşılaşılan problemleri aşmak adına çeşitli akan veri kümeleme yaklaşımları önerilmiştir. Bu alanda yapılan çalışmalar temel olarak parçalama-tabanlı, hiyerarşi-tabanlı, ızgara-tabanlı, model-tabanlı ve yoğunluk-tabanlı olarak beş ana kategoriye ayrılırlar [22].

Parçalama-tabanlı akan veri kümeleme yaklaşımlarında veri k-means gibi uzaklık tabanlı bir kümeleme yöntemi ile kümelere bölünür. Bu tür yaklaşımlarda küme şekli yuvarlaktır ve genellikle sapan verilerden etkilenirler. STREAM [23], SPE-Cluster [24] ve DCStream [25] bu yaklaşımlara örnek verilebilir. Bu çalışmaların yanında CluStream [26] ve HPStream [27] gibi parçalama-tabanlı yaklaşımların diğer yöntemlerle birlikte kullanıldığı yöntemler de mevcuttur.

Hiyerarşi-tabanlı akan veri kümeleme yaklaşımlarında veriler bir ağaç yapısında birleştirilir. Bu yaklaşımlar veri özetleme ve görselleştirme açısından avantajlıdır. Küme birleştirme ve küme bölme işlemleri sürekli olarak yapılmaktadır. Bu yaklaşımlara örnek olarak BIRCH [28], Chameleon [29], E-Stream [30], ODAC [31], SE-Stream [32], HUE-Stream [33] ve COMET-CORE [34] verilebilir. CluStream [26], ClusTree [35] ve HPStream [27] algoritmaları gibi hiyerarşi tabanlı kümelemenin diğer yöntemlerle birleştirildiği örnekler de mevcuttur.

Izgara-tabanlı (Grid-based) akan veri kümeleme yaklaşımları, veriyi ızgara (grid) denilen parçalara ayırır. Performans açısından bakıldığında hızlı yaklaşımlardır. STING [36], WaveCluster [37] ve CLIQUE [38] bu yaklaşımlara örnektir. D-Stream [39], DUCStream [40], DD-Stream [41] ve MR-Stream [42] algoritmaları ızgara ve yoğunluk tabanlı algoritmalardır.

Model-tabanlı akan veri kümeleme yaklaşımları, veri ve EM (Expectation Maximization) [43] gibi bir matematiksel model arasında bir optimizasyon bulmaya çalışır. EM algoritması k-means'in farklı bir versiyonu olarak düşünülebilir. SWEM [44], POD-Clus [45] ve SOC [46] bu yaklaşımlara örnek olarak verilebilir.

Yoğunluk tabanlı akan veri kümeleme yaklaşımlarında kümeleme yoğunluğa göre yapılır. Kümeler veri yoğunluğunun fazla olduğu alanlara göre genişlemektedir. Küme belirlemede belirli bir yoğunluk eşik değeri kullanılır. Bu sayede sapan verileri tespit etmek ve bertaraf etmek mümkündür. DBSCAN [47], OPTICS [48], DENCLUE [49], HDDStream [50], LeaDenStream [51], CODAS [52], yerel minimumdan kaçınmak için önerilmiş olan bir yöntem [53] ve HSDStream [54] bu yaklaşımlara örnek olarak verilebilir.

Bu çalışmada yoğunluk tabanlı bir akan veri kümeleme yaklaşımı olan KD-AR Stream algoritmasını öneriyoruz. Yaptığımız çalışmanın çözmeye çalıştığı uyarlanabilir yarıçap ve otomatik küme sayısının tanımlanması problemlerini çözmeye çalışan çalışmalar, mikro-küme yapısını kullanan yoğunluk tabanlı çalışmalardır [26, 35, 50, 51, 55-58]. Bu tür çalışmalarda verilerin yoğunlaştığı alanlar mikro-kümeler olarak tanımlanmakta ve belirli sayıda mikro-kümenin birleştirilmesinden de makro-kümeler oluşturulmaktadır. Bölüm 2'de detaylı olarak açıkladığımız gibi önerdiğimiz yaklaşımın mikro-küme tabanlı bu tür algoritmalardan farkı ve avantajı kümelerin geçmişi ile ilgili bilgileri tutmasıdır. Bu tür algoritmalarda eşik değerin altına düşen kümeler silinirken önerdiğimiz yaklaşımda bu tür kümeler pasif küme olarak işaretlenerek daha sonra yeniden aktive edilebilmektedir. Önerdiğimiz KD-AR Stream algoritmasının yukarıda üzerinde durduğumuz algoritmalara göre avantajlı yönlerini kısaca şöyle sıralayabiliriz:

- KD-AR Stream tamamen çevrimiçi çalışmaktadır. Çoğu akan veri kümeleme yaklaşımında çevrimiçi ve çevrimdışı aşama olarak iki evre vardır [24, 26, 29, 35, 39, 47, 52, 59, 60].
- KD-AR Stream evrimsel bir akan veri kümeleme yaklaşımıdır. Önerdiğimiz yaklaşım var olan çoğu yaklaşımın aksine [23, 24, 26, 31, 40, 58, 61] yeni küme oluşturma, var olan bir kümeyi pasif yapma, pasif bir kümeyi yeniden aktive etme, iki kümeyi birleştirme veya bir kümeyi ikiye bölme ve küme yarıçapını güncelleme gibi işlemleri gerçek zamanlı olarak yapmaktadır.
- Literatürdeki çoğu çalışmanın [27, 30, 32, 50] aksine önerdiğimiz yaklaşımda küme sayısını belirleme veya küme sayısına bir sınır koymaya ihtiyaç yoktur ve var olan çoğu çalışmanın aksine [30, 32] KD-AR Stream'de küme yarıçapı uyarlanabilir bir yapıdadır.
- KD-AR Stream'in veri özetleme yaklaşımı zaman tabanlı bir yaklaşımdır. Veriler kullanıcının belirlediği son 1 saniyede gelen veriler veya son 1 saatte gelen veriler olarak özetlenebilmektedir. Literatürde akan pencereler (sliding windows) [57, 62], mikro-kümeler [35, 58, 63],

rastgele örnekleme [64], histogram [27, 30, 32] ve taslak (sketches) [65, 66] gibi yöntemler kullanılmaktadır.

- Önerdiğimiz yaklaşımın bir diğer avantajı kümeler ile ilgili bir hafıza geçmişine sahip olmasıdır. Literatürdeki çoğu çalışmanın aksine kümeleri silmek yerine aktif–pasif yapma dönüşümü kullanılır. Bu yapı akan verinin ruhuna uygun olarak farklı zamanlarda aynı bölgede oluşan kümelerin aynı küme etiketiyle etiketlenmesini sağlar.Makalenin geri kalanı şu şekilde sıralanmaktadır. 2. bölümde önerdiğimiz K-boyutlu ağaç ve uyarlanabilir yarıçap (KD-AR Stream) tabanlı gerçek zamanlı akan veri kümeleme yaklaşımı detaylı olarak açıklanmakta iken, 3. bölümde yapılan deneysel çalışmalar açıklanmaktadır. 4. bölümde tartışma bölümü yer alırken, sonuç ve gelecekte yapılması planlanan çalışmalar 5. ve son bölümde yer almaktadır.

Tablo 1. Akan veri kümeleme alanında geliştirilmiş algoritmalardan bazılarının karşılaştırılması
(Comparision of some of algorithms in the data stream clustering area) [19]

Algoritma	Yıl	Parametreler	Avantajı	Dezavantajı	Evrimsel mi?	Kümeleme türü
STREAM [23]	2001	Küme sayısı	Performansı	Küme sayısını sınırlama	Hayır	Parçalamalı
SPE-Cluster [24]	2001	Akan ve temel pencereler	Küme sayısı belirlemede ve zamana bağlı küme adaptasyonunda avantajlıdır	Dairesel olmayan kümeleri bulmakta problemli	Evet	Parçalamalı
CluStream [26]	2003	Küme sayısı ve zaman penceresi	Özetleme sayesinde kullanıcıya esneklik sağlar	Dairesel kümeleri tespit edemiyor, sapan veriden etkilenir, büyük verilerde problemli	Evet	Parçalamalı & Hiyerarşik
HPStream [27]	2004	Maksimum küme sayısı, ortalama boyut değeri	Çok boyutlu verileri destekler, yeni gelen verilerin ağırlığı daha fazla	Dairesel olmayan kümeleri bulmakta problemli	Evet	Parçalamalı & Hiyerarşik
DUCstream [40]	2005	Izgara hücrelerinin yoğunluk eşik değeri	Dairesel olmayan şekilleri bulabiliyor	Tüm verilerin ağırlığı aynı	Evet	Yoğunluk & Grid tabanlı
DenStream [60]	2006	Küme yarıçapı eşik değeri Veri ağırlığını azaltma eşik değeri Sapan veri eşik değeri Küme ağırlığı vs.	Farklı şekillerdeki kümeleri bulabilir Sapan verileri tespit edebilir	Sapan verileri bulma işlemi performansı düşürüyor Split ve merge işlemlerini yapmıyor	Evet	Yoğunluk tabanlı
D-Stream [39]	2007	Decay factor Izgara yoğunluk eşik değeri Veri ağırlığını azaltma oranı	Yoğunluğa bakarak kümeleri tespit edebiliyor, zamana bağlı olarak küme dönüşümü desteği	Algoritmanın zaman karmaşıklığı pek çok değişkenden etkilenir	Evet	Yoğunluk & Grid tabanlı
E-Stream [30]	2007	Maksimum küme sayısı, yarıçap faktörü, akan veri hızı, silme eşik değeri, aktif olma eşik değeri, birleştirme eşik değeri	Kümelerin zamana bağlı dönüşümünü destekler Zamanla küme bölme ve birleştirme işlemlerini gerçekleşir	Dairesel olmayan kümeleri bulmakta problemli, çok sayıda parametrenin tanımlanması gerekir	Evet	Hiyerarşik
DD-Stream [41]	2008	Izgara hücrelerinin yoğunluk eşik değeri, yoğunluk azaltma oranı gibi parametreler	Farklı şekillerdeki kümeleri bulabilir Kümeleme başarısı tatmin edici	Çok boyutluluğu destekleme	Evet	Yoğunluk & Grid tabanlı
ODAC [31]	2008	Kümeleri bölme eşik değeri	Kümeleri bölme ve birleştirme işleminde başarılı	Dairesel olmayan kümeleri bulmakta problemli	Evet	Hiyerarşik

POD-Clus [45]	2008	Küme bölme ve birleştirme eşik değerleri, veri ağırlığı azaltma oranı	Dairesel olmayan kümeleri tespit edebilir	İşlem performansı kötü	Evet	Model tabanlı
COMET-CORE [34]	2009	Korelasyon eşik değeri, küme bölme ve birleştirme eşik değerleri	Çok boyutluluğu destekler	Dairesel olmayan kümeleri bulmakta problemli	Evet	Hiyerarşik
HUE-Stream [33]	2011	Veri ağırlığını azaltma oranı ve E-Stream'in diğer parametreleri	Kümelerin zamana bağlı dönüşümlerini destekler, performansı E-Stream'e göre iyi	Dairesel olmayan kümeleri bulamıyor, çok sayıda parametre kullanır	Evet	Hiyerarşik
Rough set tabanlı [67]	2012	Sapan veri eşik değeri, eski verileri silme eşik değeri	Sapan verilere karşı dirençli	Dairesel olmayan kümeleri bulmakta problemli	Evet	Rough set tabanlı parçalamalı
HDDStream [50]	2012	Küme yarıçapı, küme ağırlığı, sapan veri eşik değeri, bozunum faktörü	Farklı şekildeki kümeleri tespit edebilir, çok boyutlu verileri destekler, sapan veri tespiti	Algoritmanın performansı düşük	Evet	Yoğunluk
LeaDen-Stream [51]	2013	MMLC ve MMLCweight, MMLC ve MMLCCenter, MMLC ve MMLCradius	Kümeleme başarısı makul Sapan verilere karşı dirençli, performansı tatmin edici	Çok boyutlu veriyi desteklemiyor	Evet	Yoğunluk
SE-Stream [32]	2013	Silme eşik değeri, yarıçap faktörü, veri akış hızı gibi çok sayıda parametre	E-Stream algoritmasına göre performansı daha iyi, algoritma optimize edilmiştir.	Çok sayıda parametrenin tanımlanması gerekir	Evet	Hiyerarşik
HSDStream [54]	2015	Pencere genişliği, yarıçap eşik değeri, sapan veri ve veri sayısı eşik değeri, varyans eşik değeri vs.	Sapan verilere karşı dirençli, çok boyutlu verileri destekler, performans tatmin edicidir	Çok sayıda verinin tanımlanması gerekir	Evet	Yoğunluk
SOC [46]	2015	Küme bölme ve birleştirme eşik değerleri	Sapan verileri tespit edebilir, kümeleme başarısı tatmin edici	Küme başarısı için belli sayıda verinin ulaşmış olması gerekir	Evet	Model tabanlı
pcStream [68]	2015	Sürüklenme eşik değeri, maksimum sürüklenme boyutu, varyans yüzdesi, model veri sayısı	Kümelerin çakışmasını tespit edebilmektedir.	Sapan verilerden etkilenmekte ve başarı oranı düşmektedir.	Evet	Model tabanlı
DCSTREAM [25]	2016	Küme bölme ve birleştirme eşik değerleri, sapan veri eşik değeri	STREAM algoritmasına göre çok daha başarılı, Çok boyutlu veriyi destekler	Dairesel olmayan kümeleri tespit etmek problemli	Evet	Parçalamalı
DBSTREAM [63]	2016	Ağırlık azaltma faktörü, yarıçap, α vs.	Mikro-kümeler arasında kalan verileri de kullanır buna göre gerekirse kümeler güncellenmektedir.	Mikro-kümeler arasındaki verileri de işlediği için zaman karmaşıklığı çok	Evet	Yoğunluk

FEAC-Stream [69]	2017	InitSize, k-means iterasyonu vs.	Küme sayısının ön tanımlı olmasına gerek yok	Eski oluşturulmuş kümeler silinir	Evet	Yoğunluk
DPStream [61]	2017	Yerel yarıçap, global yarıçap, işleme alınan veri boyutu vs.	Tamamıyla çevrimiçi ve farklı şekilleri destekler. Ayrıca çok hızlı çalışan bir algoritmadır.	Çok sayıda parametre var ve parametrelerin çok doğru seçilmesi gerekir.	Evet	Hiyerarşik ve Yoğunluk
CEDAS [58]	2017	Yarıçap, küme yaşlanma oranı ve minimum eşik değeri	Tamamıyla çevrimiçi çalışan ve farklı şekildeki kümeleri tespit edebilir.	Yarıçap için belirlenen değer başarıyı çok etkiliyor	Evet	Hiyerarşik
LLDStream [70]	2017	Yarıçap ve yoğunluk parametresi	Boyut indirgemeyi destekler, farklı şekilli kümeleri tespit edebilir, performansı iyi	Tamamen çevrimiçi değil ve seçilen parametreler sonucu etkiler	Evet	Yoğunluk
Xinxin vd. önerdiği algoritma [71]	2018	Yoğunluk eşik değeri, yoğunluk katsayısı, en yakın komşu değeri, paylaşım değeri	Farklı şekildeki kümeleri bulur, sapan verileri tespit eder, küme sayısını sınırlamaz	Optimum parametreleri belirlemek zor	Hayır	Yoğunluk & Grid tabanlı
SyncTree [72]	2018	Küme sayısı, kova boyutu, kesişim aralığı, çevrim dışı kümeleme değişkeni vs.	Veri özetleme şekli kendine özgü, performansı iyi	Çevrimiçi çevrimdışı evrelerden oluşması, çok sayıda parametre belirlenmeli	Evet	Hiyerarşi & Yoğunluk
evoStream [73]	2018	Küme sayısı, popülasyon boyutu, bozunum faktörü, yarıçap vs.	Kaynakları etkin kullanması, kümeleme başarısının iyi olması	En iyi parametreleri belirlemek güç, tamamen çevrimiçi değil	Evet	Yoğunluk
StreamSW [74]	2018	Sapan veri eşik değeri, yarıçap eşik değeri ve ağırlık eşik değeri	Yüksek performanslı ve başarılı kümeleme yapması	Çevrimiçi ve çevrimdışı evreler ve çok boyutlu verilerde performans	Evet	Yoğunluk
BOCEDS [75]	2019	İşlenecek veri miktarı, min. ve maks. yarıçap ve min. eşik değeri	Çevrimiçi çalışan, yerel en iyi yarıçapı belirleme, farklı küme şekillerini destekleme	Kullanılan hafıza biraz yüksek	Evet	Yoğunluk
ChronoClust [76]	2019	Minimum ağırlık eşik değeri, maksimum yarıçap eşik değeri, maksimum mikro-küme sayısı vs.	Zaman damgalı verileri işlemesi,	Çevrimiçi-çevrimdışı evrelerden oluşması, çok sayıda parametre kullanması	Evet	Yoğunluk

2. ÖNERİLEN YÖNTEM (PROPOSED METHOD)

2.1. K-Boyutlu Ağaç ve Uyarlanabilir Yarıçap (KD-AR Stream)Tabanlı Gerçek Zamanlı Akan Veri Kümeleme

(KD-Tree and Adaptive Radius (KD-AR Stream) Based Real-Time Data Stream Clustering)

Önerdiğimiz yaklaşım temel anlamda gücünü çok boyutluluğu destekleyen K-boyutlu ağaçtan almaktadır. Yeni bir kümenin tanımlanıp tanımlanmayacağına karar vermek için hiçbir kümeye ait olmayan veriler K-boyutlu ağaca yerleştirilir ve alan araştırması (rangesearch*)* işlemi gerçekleştirilir (K-boyutlu ağaç ve alan araştırması işlemi için Bölüm 2.2'ye bakınız). KD-AR Stream algoritmasının ana algoritması Şekil 1'de görüldüğü gibidir. Önerdiğimiz yaklaşım, E-Stream [30] algoritmasında üzerinde durulan kümeleme adaptasyonu işlemlerini desteklemektedir. E-Stream algoritmasında bir kümenin oluşması, var olan bir kümenin silinmesi, bir kümenin ikiye bölünmesi, birbirine yeterince yaklaşan iki kümenin birleştirilmesi ve bir kümenin yarıçapının veya sahip olduğu veri sayısının değişmesi gibi beş çeşit evrimsel değişim desteklenmektedir. Önerdiğimiz yaklaşımın sahip olduğu bu beş evrimsel işlemi kısaca şu şekilde açıklayabiliriz:

```
Algorithm: KD-AR Stream
while yeni bir veri geldiğinde do
    Yaşlandırma;
    YeniKümeOluştur;
    ÇakışanKümeleriBul;
    KümeBöl;
    EnYakınKümeyiBul;
    KümeMerkezleriniGüncelle;
    YarıçaplarıGüncelle;
    KümeleriAktifleştir;
end
```

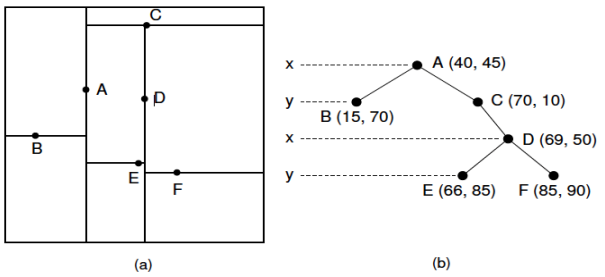
Şekil 1. KD-AR Stream algoritması (KD-AR Stream algorithm)

- **Yeni bir kümenin oluşması (Appearence):** Belirli yarıçap içinde ve diğer kümelerden yeterince uzakta belirli bir sayıdaki veri yeni bir küme olarak tanımlanır (Detaylı açıklama için Bölüm 2.5.1.'e bakınız).
- **Aktif bir kümenin pasif olması/Pasif bir kümenin tekrar aktif olması (Activation-Inactivation):** Aktif bir kümenin sahip olduğu veri sayısı, ömrünü tamamlayan verilerin silinmesiyle belirlenmiş eşik değerinin altına düşerse bu küme pasif küme olarak işaretlenir. Pasif durumdaki bu kümenin yeni veri alarak suretiyle yeniden eşik değerinin üzerine çıkarsa yeniden aktive edilir (Detaylı açıklama için Bölüm 2.5.6'ya bakınız).
- **Küme evrimi (Self-evolution):** Küme yeni veriler aldıkça veya yaşam süresini tamamlayan veriler silindikçe kümenin yarıçapı, sahip olduğu veri sayısı ve küme merkezinin bulunduğu koordinatlar gibi değerler değişir (Detaylı açıklama için Bölüm 2.5'e bakınız).
- **Küme birleştirme (Merge):** Eğer aktif iki kümeden birinin kabuk yarıçapı ile diğerinin çekirdek yarıçapının toplamı merkezleri arasındaki mesafeden fazla ise bu iki küme birleştirilir (Detaylı açıklama için Bölüm 2.5.4'e bakınız).
- **Küme bölme (Split):** Aktif bir kümede bulunan verilerden belirli sayıdaki bir kısmının oluşturduğu aday kümenin kabuk yarıçapı ile kümenin geri kalanının oluşturduğu kümenin kabuk yarıçapı kesişmiyorsa bu iki küme bölünür (Detaylı açıklama için Bölüm 2.5.5'e bakınız).

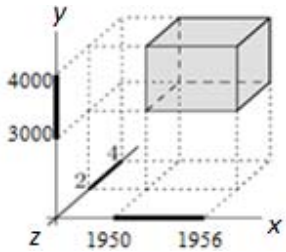
2.2. *K-Boyutlu Ağaç ve Alan Araştırması (RangeSearch) İşlemi*
(Kd-Tree and Range Search)

K-boyutlu ağaç Jon Bentley tarafından 1970'lerde önerilmiş olan çok boyutluluğu destekleyen bir dengeli ikili arama ağacı türüdür [77]. Veriler ağaca yerleştirilirken veya ağaçta arama yapılırken ağacın her seviyesi farklı bir niteliğe göre sıralanmaktadır. Şekil 2'de de görüldüğü gibi iki boyutlu veriler ağaca yerleştirilecekse, ilk önce x niteliğine göre sonra da y niteliğine göre yerleştirilmektedir ve tekrar x'e dönerek devam eder.

K-boyutlu ağaçlarda belirli bir aralıktaki verilere ulaşma çabası bir alan araştırması işlemidir. Örneğin Şekil 3'te görüldüğü gibi bir şirketin çalışanlarına ait verileri ele alalım. x çalışanların doğum yıllarını, y çalışanların maaşlarını ve z çalışanların sahip olduğu çocuk sayısını göstermek üzere 1950-1956 yılları arasında doğmuş, 2-4 çocuğa sahip ve 3000-4000 lira arası maaşa sahip çalışanlar koyu olarak gösterilen bölge olacaktır.



Şekil 2. K-boyutlu ağaç örneği (Kd-Tree example) [78]

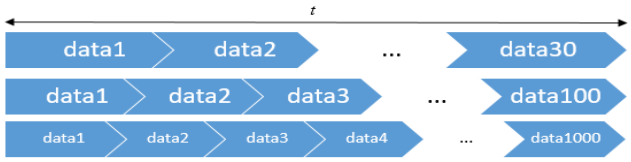


Şekil 3. Bir şirketin çalışanları için alan araştırması örneği (Range search example for employees of a company) [79]

2.3. Zaman Tabanlı Veri Özetleme
(Data Summarization Bazed on Time)

Klasik akan veri özetleme yaklaşımları, veriyi özetlerken ya belirli bir sayıdaki bir miktarını alır, ya da verileri ağırlıklandırarak mikro-küme olarak adlandırılan parçalar olarak ifade eder. Ancak çoğu uygulamada zaman tabanlı bir özetlemeye de ihtiyaç duyulmaktadır. Son 1 saniye veya son 1 saat içinde gelen verileri işleyecek bir özetleme yaklaşımına da ihtiyaç duyulmaktadır. Önerdiğimiz yaklaşım kullanıcının belirlediği zamana göre özetleme yapmaktadır. Dolayısıyla zamanın önemli olduğu uygulamalar için bu yaklaşım çok avantajlıdır. Ancak bu süre zarfında gelen veri miktarı sabit olmayabilir. Şekil 4'te de görüldüğü gibi söz konusu zaman zarfında 100 veri

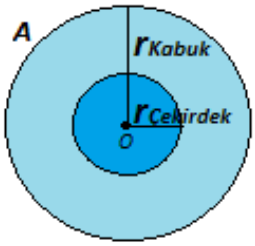
gelebileceği gibi 1000 veri de gelebilir. Hatta on binlerle ifade edilen veri de gelebilir. Bu durumda bu verileri işleme imkânı da düşmektedir. Bu problemi aşmak için, söz konusu bu verilerden en son gelen *TN* (ön tanımlı değişken) tanesi alınarak işlem yapılmaktadır.



Şekil 4. Zamana bağlı özetlemenin veri sayısı ile ilişkisi
(Time-based summarization and data quantity correlation)

2.4. Uyarlanabilir Yarıçap (Adaptive Radius)

Önerdiğimiz yaklaşımda, Şekil 5'te de görüldüğü gibi iki çeşit yarıçap bulunmaktadır. Bunlardan ilki kabuk yarıçaptır. Kabuk yarıçap kümenin sınırını belirler. Ayrıca küme birleştirme ve bölme gibi işlemlerde de kullanılmaktadır. Kümeye ait verilerden küme merkezine en uzak olan verinin uzaklığı kümenin kabuk yarıçapı olarak atanmaktadır.



Şekil 5. Yarıçap türleri (Radius types)

Çekirdek yarıçap, kümenin verilerinin yoğunlaştığı alan olarak düşünülebilir ve iki kümenin birleştirilip birleştirilmeyeceğine karar vermek için kullanılmaktadır. Çekirdek yarıçapın değeri niteliklerin standart sapmaları üzerinden hesaplanmaktadır. Bunun için ilk olarak kümeye ait her bir nitelik için standart sapmalar hesaplanır. Daha sonra bu standart sapmaların ortalaması alınır. Elde edilen değer kümenin çekirdek yarıçapı olarak atanır. *d* verilerin sahip olduğu nitelik sayısını, *N* kümenin sahip olduğu veri sayısını, μ *j* niteliğine ait ağırlık merkezi değerini ve X_i üzerinde işlem yapılan veri olmak üzere, kümenin çekirdek yarıçapı (σ) Eş. 1 ile hesaplanır.

$$\sigma = \frac{1}{d}\sum_{j}^{d}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(X_i^j - \mu_i^j\right)^2} \quad (1)$$

2.5. Temel İşlemler (Basic Operations)

2.5.1. Yeni bir küme oluşturma (Forming a new cluster)

Yeni küme oluşturmak için hiçbir kümeye ait olmayan veriler K-boyutlu ağaca yerleştirilir ve bu veriler üzerinde alan araştırması işlemi gerçekleştirilir. Amaç hiçbir kümeye ait olmayan *N* (kullanıcının belirlediği ön tanımlı değişken) tane veri belirlenmiş yarıçapta (*r* – ön tanımlı değişken) bulunuyorsa ve bu verilerin oluşturduğu aday kümenin merkezinin diğer kümelerin merkezlerine olan uzaklığı yeterli ise bu aday küme yeni bir küme olarak tanımlanır. Oluşturulan bu kümenin merkezi söz konusu verilerin oluşturduğu ağırlık merkezidir ve Eş. 2 ile hesaplanır. Burada *X* bir veriyi, X_i^j, *i*'inci verinin *j*'inci niteliğini, *N* kümenin sahip olduğu veri sayısı ve μ^j kümenin *j*'inci niteliğinin ağırlık merkezini ifade etmektedir.

$$\mu^j = \frac{1}{N}\sum_{i=1}^{N} X_i^j \quad (2)$$

2.5.2. Küme yarıçapının artırılması ve azaltılması
(Increasing and decreasing cluster radius)

Yeni bir veri geldiği zaman bu verinin en yakın olduğu küme tespit edilir. Verinin kümenin merkezine olan uzaklığı eğer kümenin kabuk yarıçapından az ise yani veri kümenin içinde ise veri ilgili kümeye atanır ve küme merkezi güncellenir. Verinin küme merkezine olan uzaklığı, kümenin kabuk yarıçapından fazla, maksimum yarıçaptan az ve kümenin kabuk yarıçapı ile yarıçap arttırma eşik değerinin toplamından az ise bu veri bu kümeye atanır.

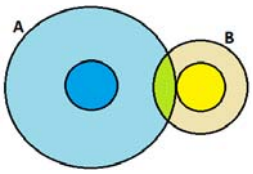
Bir kümenin kabuk yarıçapı belirlenmiş olan maksimum yarıçap değerine kadar arttırılabilir; ancak söz konusu artış aşama aşama gerçekleştirilmektedir. Bu artış miktarı her seferinde belirlenmiş yarıçap arttırma eşik değeri (*r threshold*) kadar yapılabilmektedir. Örneğin bir kümenin mevcut kabuk yarıçapının 10, kabuk yarıçap arttırma eşik değerinin 5 olduğunu ve kabuk yarıçapın ulaşabileceği maksimum değerin de 20 olduğunu varsayalım. Söz konusu kümenin merkezine uzaklığı 18 olan yeni bir verinin geldiğini varsayalım. Bu durumda veri ilgili kümeye eklenmeyecektir. Çünkü kümenin mevcut yarıçapı 10 olduğundan bu aşamada önce 15'e kadar arttırılabilir. Böyle bir yapının kullanılmasının nedeni kümelerin daha kararlı bir yapıda olmasını sağlamak ve sapan verilere karşı kümeleri daha dirençli hale getirmektir.

2.5.3. Veri yaşlandırma ve ömrünü tamamlayan verilerin silinmesi
(Data aging and deletion of expired data)

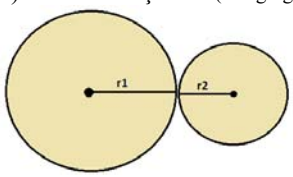
Önerdiğimiz yaklaşımda zaman tabanlı bir özetleme yaklaşımı kullanılmaktadır. Bu nedenle her bir verinin ömrü belirlenmiş olan süre kadardır. Ömrünü tamamlayan her veri silinir. Silinen veri hiçbir kümeye ait olmayan bir veri olabileceği gibi bir kümeye de ait olabilir. Eğer silinen veri bir kümeye aitse bu kümenin kabuk yarıçapı, çekirdek yarıçapı, merkez koordinatları ve sahip olduğu veri sayısı güncellenmektedir.

2.5.4. İki kümenin birleştirilmesi (Merging two clusters)

İki kümenin birleştirilip birleştirilmeyeceğine karar verirken merkezleri arasındaki Öklid uzaklığına bakılır. Şekil 7a'da da görüldüğü gibi eğer bir kümenin kabuk yarıçapı başka bir kümenin çekirdek yarıçapı ile kesişiyorsa bu iki küme birleştirilir [60]



a) Küme birleştirme (Merging of two clusters)



b) Küme bölme (Splitting of any cluster)

Şekil 7. Küme bölme ve birleştirme işlemleri
(Splitting and merging of clusters)

2.5.5. Bir kümenin ikiye bölünmesi (Splitting a cluster)

Bir kümenin bölünüp bölünmeyeceğine karar verirken öncelikle kümenin sahip olduğu veri sayısının minimum küme oluşturmak için gerekli olan sayının iki katına, kabuk yarıçapının iki katına ve çekirdek yarıçapının da en az *r* kadar olması şartı aranır. Bu şartların aranmasının nedeni performansı arttırmak için gereksiz işlem yükünü düşürmektir. Bölünüp bölünmeyeceğine karar verilecek olan kümeye ait veriler K-boyutlu ağaca yerleştirilir ve bu ağaç üzerinde alan araştırması işlemi yapılır. *r* yarıçap içerisinde *N* tane verinin oluşturduğu aday kümenin, geri kalan verilerin oluşturduğu kümenin merkezine olan uzaklığı bu iki kümenin kabuk yarıçaplarının toplamından büyükse bu küme bu şekilde ikiye bölünür. Şekil 7b bu işlemi göstermektedir.

2.5.6. Aktif bir kümenin pasif yapılması ve pasif bir kümenin yeniden aktive edilmesi
(Inactivation of an active cluster and reactivation of an inactive cluster)

Aktif bir kümenin sahip olduğu veri sayısı verilerin zamana bağlı olarak silinmesi nedeniyle küme oluşturmak için belirlenmiş olan eşik değerin altına düşerse bu küme pasif yapılır. Pasif durumdaki bir kümenin sahip olduğu veri sayısı yeni veri alması neticesinde eşik değerin üzerine tekrar çıkarsa bu küme yeniden aktive edilir.

2.6. Algoritma (Algorithm)

Ana algoritması Şekil 1'de verilen KD-AR Stream'de kullanılan notasyonlar ve açıklamaları Tablo 2'de verilmektedir. Önerdiğimiz yaklaşımda kullanılan alt algoritmaları kısaca şu şekilde sıralayabiliriz:

Sözde kodu aşağıda verilen **Yaşlandırma** algoritmasında, işlenen tüm veriler yaşlandırılmakta ve ömrünü tamamlayan veriler silinerek *silinenVeri* değişkenine eklenmektedir. Veri ömrü eşik değeri, uygulamanın türüne ve ihtiyaca göre belirlenmelidir. Örneğin bankacılık gibi bir uygulamada her verinin ömrü için 10 saniye veya 10 dakika gibi bir süre belirlenirken, bir salgın izleme uygulaması için 1 gün veya 1 hafta olarak belirlenebilir.

Tablo 2. KD-AR Stream algoritmasında kullanılan kısaltmalar ve anlamları
(The list of notations used in the KD-AR Stream algorithm)

NOTASYON	AÇIKLAMASI
C	Küme tablosu
IŞLENENVERI	Üzerinde işlem yapılan veriler
SILINENVERI	Silinen veriler
T	Veri ömrü eşik değeri
TN	Özet olarak alınmış olan veri miktarı çok ise ne kadarının alınacağını belirten eşik değer
R	Kümenin kabuk yarıçapı
R_THRESHOLD	Yarıçap arttırma eşik değeri
R_MAX	Yarıçap için maksimum değer
R_{TEMP}	Aday kümenin kabuk yarıçapı
D	Verilerin sahip olduğu nitelik sayısı
N	Küme oluşturmak için gerekli minimum veri sayısı
AĞAÇ / KBAĞACI	K-boyutlu ağaç
C_{TEMP}	Aday küme
MES	İki nokta arasındaki Öklid uzaklığı (Küme-veri veya küme-küme mesafesi)
$C_{(KYARIÇAP)}$	Kümenin kabuk yarıçapı
$C_{(CYARIÇAP)}$	Kümenin çekirdek yarıçapı (Niteliklerin standart sapmalarının ortalaması)
STD	Standart sapma
KÜMENO	Kümenin indeksi
MERKEZKOORDINAT	Kümenin ağırlık merkezi diğer bir deyişle kümenin merkez koordinatları
C_IAKTIF	C_i kümesinin aktif olma durumu
V	Kök düğüm
V_{SOL}	Sol alt ağaç
$V_{SAĞ}$	Sağ alt ağaç
DERINLIK	Ağacık derinliği

Algorithm 1: Yaşlandırma

Input: C, işlenenVeri, silinenVeri, t, TN
Output: C, işlenenVeri, silinenVeri
if *işlenenVeriBoyutu* > *1* **then**
  foreach $Veri_i \in işlenenVeri$ **do**
    if $Veri_i$'nin yaşı $t_i > t$ **then**
      if $Veri_i$ *herhangi bir kümeye aitse* **then**
        silinenVeri$_i$ ← Veri$_i$;
      end
      Veri$_i$'yi sil;
    end
  end
  if *işlenenVeriBoyutu* > TN **then**
    işlenenVeri ← işlenenVeri'nin son TN tanesini al;
  end
end

Sözde kodu aşağıda verilen **YeniKümeOluştur** algoritması hiçbir kümeye ait olmayan veriler üzerinden gerekiyorsa yeni küme tanımlar.

Algorithm 2: YeniKümeOluştur

Input: C, işlenenVeri, r, N
Output: C, işlenenVeri
Herhangi bir kümeye ait olmayan bütün $Veri$'yi al;
ağaç ← KBAğacıOluştur($Veri$);
if $Veri$*'nin boyutu* > N **then**
  foreach $Veri_i \in Veri$ **do**
    C_{temp} ← AlanArama(ağaç, $Veri_i$, r);
    if C_{temp}*'in boyutu* > N **then**
      C_{temp}'i yeni bir küme olarak tanımla;
    end
  end
end

ÇakışanKümeleriBul algoritması Bölüm 2.5.4'te açıklandığı gibidir ve sözde kodu aşağıdaki gibidir.

Algorithm 3: ÇakışanKümeleriBul

Input: C, işlenenVeri, d
Output: C, işlenenVeri
foreach *Küme* $C_i \in C$ **do**
  foreach *Küme* $C_j \in C$ **do**
    if C_i *ve* C_j *aktif kümeler ise ve* $i \neq j$ **then**
      $mesafe$ ← mesafeBul(C$_i$, C$_j$, d);
      if $mesafe < (C_{i(KYaricap)} + C_{j(CYaricap)})$ *or* $mesafe < (C_{i(CYaricap)} + C_{j(KYaricap)})$ **then**
        birleştir(C$_i$, C$_j$);
      end
    end
  end
end

KümeBöl algoritması Bölüm 2.5.5'te açıklandığı gibi kümeleri böler. Sözde kodu aşağıdaki gibidir.

Algorithm 4: KümeBöl

Input: C, işlenenVeri, r, N, d
Output: C, işlenenVeri
foreach *aktif küme* $C_i \in C$ **do**
  if C_i*'nin veri sayısı*$> 2*N$ *ve* $C_{i(KYaricap)} >= 2*r$ *ve* $C_{i(CYaricap)} >= r$ **then**
    ağaç ← KBAğacıOluştur(C_i*'nin verileri*);
    foreach X *verisi* $\in C_i$ **do**
      C_{temp} ← AlanArama(ağaç, X, r);
      if C_{temp}*'in veri sayısı* $>= N$ **then**
        mesafe ← mesafeBul(C_{temp}, C_i-C_{temp}, d);
        r_{temp} ← yarıçapBul(C_{temp});
        r_{Ci} ← yarıçapBul(C_i-C_{temp});
        if $mesafe > r_{temp} + r_{Ci}$ **then**
          C_i'yi böl;
        end
      end
    end
  end
end

EnYakınKümeyiBul algoritması yeni gelen veya hiçbir kümeye ait olmayan veriler arasında kümelerin durumunun değişmesi nedeniyle ilgili kümelere dâhil edilmek için yeterli yakınlığa ulaşan var ise bu verileri ilgili kümelere eklemektedir. Sözde kodu aşağıdaki gibidir.

Algorithm 5: EnYakınKümeyiBul

Input: C, işlenenVeri, d, r_threshold, r_max
Output: C, işlenenVeri
foreach *hiçbir kümeye ait olmayan* $işlenenVeri_i \in işlenenVeri$ **do**
  $mesafe$ ← max;
  foreach C_j *aktif kümesi* $\in C$ **do**
    mes ← mesafeBul(C$_j$, işlenenVeri$_i$, d);
    if $mes < mesafe$ *ve* $mes < C_{j(KYaricap)} + r\_threshold$ *ve* $mes < r\_max$ **then**
      mesafe ← mes;
      kümeNo ← j;
    end
  end
  işlenenVeri$_i$'nin küme numarası ← kümeNo;
end

KümeMerkezleriniGüncelle kümelerin merkez koordinatlarını günceller. Ancak bu güncelleme işlemini kümenin aktiflik durumuna bağlı olarak iki şekilde yapmaktadır. Eğer küme aktif bir küme ise bu kümenin merkez koordinatlarını kümenin aktif verileri üzerinden hesaplar. Ama eğer küme pasif bir küme ise bu durumda kümenin merkez koordinatlarını varsa kümenin aktif verileri ve kümenin silinmiş verileri üzerinden hesaplar (Eksik olan sayıda veriyi silinmişlerden tamamlar). KümeMerkezleriniGüncelle fonksiyonu üzerinde durduğumuz yaklaşımı aşağıdaki gibi gerçekleştirmektedir.

Algorithm 6: KümeMerkezleriniGüncelle

Input: C, işlenenVeri,silinenVeri, d, N, r
Output: C, işlenenVeri
foreach C_i *kümesi* $\in C$ **do**
  C_i kümesinin veri sayısı ← size(işlenenVeri'de bulunan C_i kümesine ait veri sayısı);
  if $C_i Aktif = true$ **then**
    foreach *nitelik* $d_j \in d$ **do**
      $C_i merkezKoordinat^j$ ← ortalama(işlenenVeri$_i{}^j$);
    end
  end
  else
    C_i kümesinin eksik verilerini kendisine ait $silinenVeri$'den tamamla;
    foreach *nitelik* $d_j \in d$ **do**
      $C_i merkezKoordinat^j$ ← ortalama(işlenenVeri$_i{}^j$);
    end
  end
end

YarıçaplarıGüncelle tüm kümeler için çekirdek yarıçap ve kabuk yarıçap değerlerini hesaplar. Eğer küme aktif bir küme değilse kümenin kabuk yarıçapını *r/2* olarak atar. Bu değerin atanmasının nedeni kümeyi sapan verilerden korumak içindir. Kümelerin yarıçap değerleri aşağıdaki sözde kodda görüldüğü şekilde hesaplanmaktadır.

KümeleriAktifleştir her kümenin sahip olduğu veri sayısına bakarak kümelerin aktif mi yoksa pasif mi olacağına karar vermektedir. KümeleriAktifleştir fonksiyonunun sözde kodu aşağıda görüldüğü gibidir.

KBAğacıOluştur algoritması aldığı verileri bir K-boyutlu ağaca yerleştirir. KBAğacıOluştur özyinelemeli bir yapıda

çalışarak verileri sağ ve sol alt ağaçlara yerleştirir. Bu şekilde çalışarak oluşturduğu ağacı döndürür. KBAğacıOluştur algoritmasının sözde kodu aşağıdaki gibidir.

Algorithm 7: YarıçaplarıGüncelle

```
Input: C, işlenenVeri, r_max, r, d
Output: C, işlenenVeri
foreach Ci kümesi ∈ C do
    mesafe ← min;
    foreach Ci kümesine ait Veri_i ∈ işlenenVeri do
        mes ← mesafeBul(Ci,Veri_i);
        if mes > mesafe then
            if Ci aktif bir küme ise then
                if mes >= r ve mes <= r_max then
                    C_i(KYaricap) ← mes;
                    C_i(CYaricap) ← ortalama(std( Ci kümesine ait
                      veriler));
                end
                else if mes <= r then
                    C_i(KYaricap) ← r;
                    C_i(CYaricap) ← ortalama(std( Ci kümesine ait
                      veriler));
                end
            end
            else
                C_i(KYaricap) ← r/2;
            end
        end
    end
end
```

Algorithm 8: KümeleriAktifleştir

```
Input: C, işlenenVeri, d, N
Output: C, işlenenVeri
foreach Ci kümesi ∈ C do
    if Ci kümesinin veri sayısı >= N then
        CiAktif ← true;
    end
    else
        CiAktif ← false;
    end
end
```

Algorithm 9: KBAğacıOluştur

```
Input: Veri
Output: ağaç
if Veri sayısı = 1 then
    return Veri;
end
else if derinlik çift ise then
    Veri'yi X medyana göre dikey şekilde Veri1 ve Veri2 olarak iki alt
      kümeye ayır;
end
else
    Veri'yi Y medyana göre yatay şekilde Veri1 ve Veri2 olarak iki alt
      kümeye ayır;
end
v düğümünü oluştur;
v_sol ← KBAğacıOluştur(Veri1, derinlik+1);
v_sag ← KBAğacıOluştur(Veri2, derinlik+1);
v_sol'u v'nin sol çocuğu, v_sag'ı da v'nin sağ çocuğu olarak ata;
return v;
```

AlanArama algoritması K-boyutlu ağaçta alan araştırması yapar. Ağaçta kullanıcının belirlediği *r* yarıçapta minimum *N* tane veri var mı diye araştırır ve sözde kodu aşağıdaki gibidir.

2.7. Çalışma Zamanı Karmaşıklığı (Run-Time Complexity)

n, tamponlanmış veri sayısı, *d* verilerin sahip olduğu nitelik sayısı ve *m* küme sayısı olmak üzere algoritmaların karmaşıklığı Tablo 3'te görüldüğü gibidir. KD-AR Stream algoritmasının genel karmaşıklığı beklendiğinden çok daha iyidir. Çünkü her algoritmanın çalışması belirli şartlara bağlıdır.

Algorithm 10: AlanArama

```
Input: Düğüm v, r
Output: Alandaki verilerin indeksleri
if v bir yaprak ise then
    Eğer veri r yarıçap alanında ise veriyi rapor et;
end
else
    Sol alt ağacı değerlendir;
    if Sol alt ağaç tamamen r yarıçap içerisinde ise then
        Sol alt ağacı rapor et;
    end
    else if sol alt ağaç r ile kesişiyorsa then
        AlanArama(sol alt ağac, r);
    end
    Sağ alt ağacı değerlendir;
    if Sağ alt ağaç tamamen r yarıçap içerisinde ise then
        Sağ alt ağacı rapor et;
    end
    else if Sağ alt ağaç r ile kesişiyorsa then
        AlanArama(sağ alt ağac, r);
    end
end
```

3. DENEYSEL ÇALIŞMA (EXPERİMENTAL RESULTS)

3.1. Deneysel Tasarım (Experimental Design)

Geliştirdiğimiz yöntem, yöntemimizle çok sayıda ortak özelliğe sahip olmaları ve başarılı sonuçlar vermeleri nedeniyle CEDAS [58], DPStream [61] ve SE-Stream [32] algoritmaları ile karşılaştırılmıştır. SE-Stream, E-Stream [30] algoritmasının evrimsel yapısını destekleyen, öznitelik indirgeme özelliğine sahip bir versiyonudur. CEDAS algoritması mikro-küme yapısını kullanan çizge tabanlı bir akan veri kümeleme yaklaşımıdır. CEDAS algoritmasının en büyük avantajı dairesel olmayan kümeleri de tespit edebilmesidir. CEDAS algoritması da SE-Stream algoritması gibi gerçek zamanlı çalışan bir akan veri kümeleme algoritmasıdır. DPStream algoritması Leading Tree (LT) [80] tabanlı ve çevrimiçi-çevrimdışı evrelerden oluşan hiyerarşik ve yoğunluk tabanlı bir akan veri kümeleme yaklaşımıdır. DPStream dairesel olmayan kümeleri de tespit etme yeteneğine sahiptir.

Karşılaştırma işlemi hem kümeleme başarısı, hem de zaman karmaşıklığı açısından gerçekleştirilmiştir. Tüm karşılaştırma işlemleri Intel® Core™ i5-4460S CPU 2.90 GHz işlemcili ve 8 GB RAM kapasiteli ve Windows 10 yüklü bir bilgisayarda, Matlab 2017a versiyonu kullanılarak gerçekleştirilmiştir.

KD-AR Stream algoritmasında kullanıcının tanımlaması gereken 6 tane değişken bulunmaktadır. Bunlardan *N*, küme oluşturmak için gereken minimum veri sayısını ifade ederken, *t* her bir verinin ömrünü ifade etmektedir. Belirlenmiş *t* sürede gelen veri miktarı çok fazla olursa en son gelen *TN* tane veri özet olarak alınır. *r* değişkeni, yeni küme tanımlarken minimum *N* tane verinin bulunması gereken alanın yarıçapını ifade etmektedir. *r_threshold* değişkeni küme yarıçapını arttırma/azaltma eşik değerini ifade etmekteyken, *r_max* ise küme yarıçapının ulaşabileceği maksimum değeri ifade etmektedir.

UCI'nin KDD veri seti neredeyse tüm algoritmalarda ortak olduğu için ilgili çalışmalarda kullanılan parametreler seçilmiştir [32, 58, 61]. Bununla beraber, ilgili çalışmalarda kullanılmamış olan veri setleri için en iyi sonuç veren parametreler ızgara arama (grid search) ile tespit edilerek kullanılmıştır. Tüm algoritmalar için en iyi sonuç veren parametreler tespit edilirken, en yüksek doğruluk (accuracy) sonucunu üreten parametreler en iyi parametreler olarak seçilmiştir. Algoritmaları test ederken kullanılan parametreler Tablo 4, 5, 6 ve 7'de verilmiştir.

Tablo 3. Fonksiyonların karmaşıklığı (Complexity of functions)

Name of Function	Complexity
Yaşlandırma	$O(n)$
YeniKümeOluştur	$O(n \log n)$
ÇakışanKümeleriBul	$O(m^2)$
KümeBöl	$O(mn \log n + n\sqrt{n})$
EnYakınKümeyiBul	$O(mn)$
KümeMerkezleriniGüncelle	$O(md)$
YarıçaplarıGüncelle	$O(mn)$
KümeleriAktifleştir	$O(m)$
KBAğacıOluştur	$O(n \log n)$
AlanArama	$O(\sqrt{n})$

Tablo 4. KD-AR Stream algoritmasında kullanılan parametreler (Test parameters of KD-AR Stream algorithm)

Veri setleri						
Parametreler	KDD	Fisher Iris	Breast Cancer	MrData	ExclaStar	IdealData
N	90	5	3	25	14	3
t (s)	0,5	0,3	0,3	0,1	3	0,05
TN	160	90	200	200	50	30
r	1,4	1	7,5	15	4,5	4
r_threshold	4	0,55	2,5	2,5	1,5	3
r_max	6,55	1,55	10,35	26	6	9

Tablo 5. SE-Stream algoritmasında kullanılan parametreler (Test parameters of SE-Stream algorithm) [32]

Veri setleri						
Parametreler	KDD	Fisher Iris	Breast Cancer	MrData	ExclaStar	IdealData
α	5	5	5	5	5	5
max_cluster	10	3	4	4	3	5
stream speed	100	100	1000	1000	100	100
decay_rate	0,1	0,1	0,1	0,1	0,1	0,9
radius_factor	3	1	5	7	2	8
remove_threshold	0,1	0,1	0,01	0,01	0,1	0,9
merge_threshold	1,25	0,5	2	1,5	1,5	3
active_threshold	5	5	5	5	5	5
number of projected dimension (l)	10	no projection	3	no projection	no projection	no projection

Tablo 6. CEDAS algoritmasında kullanılan parametreler (Test parameters of CEDAS algorithm) [58]

Veri setleri						
Parametreler	KDD	Fisher Iris	Breast Cancer	MrData	ExclaStar	IdealData
radius	2,5	1	8	22	4,8	10
Decay	200	200	200	200	200	200
MinThreshold	1	1	1	1	1	1

3.2. Kullanılan Veri Setleri (Used Datasets)

Deneysel çalışmada UCI[1]'nin KDD CUP'99, Fisher Iris, Breast Cancer ve zaman damgalı Occupancy veri setleri, DPStream algoritmasında kullanılan ExclaStar ve MrData ile bu çalışmada önerdiğimiz, yöntemin tüm özelliklerini ölçen kendi ürettiğimiz IdealData veri setleri kullanılmıştır. Kullanılan veri setlerinin özellikleri Tablo 8'de kısaca açıklanmıştır.

3.3. Kümeleme Başarısını Ölçme (Cluster Validation)

Kümeleme başarısını ölçmek için Saflık (Purity), Doğruluk (Accuracy), F-Skor (F-Score) ve Silhouette İndeks parametreleri kullanılmıştır. Doğruluk ve F-Skor parametreleri KD-AR Stream algoritmasının evrimsel yapısını daha iyi test etmek adına kullanılmıştır. Algoritmaların performanslarını karşılaştırmak için de algoritmalar harcadıkları süre açısından karşılaştırılmıştır.

Saflık kümelerin kendi içerisinde ne kadar saf olduğunu tespit etmeye çalışan bir test türüdür. *K,* küme sayısı ve $|C_i^d|$, $|C_i|$ kümesindeki baskın küme etiketi olmak üzere ortalama Saflık Eş. 3 ile hesaplanır.

$$Saflık = \frac{\sum_{i=1}^{K} \frac{|C_i^d|}{|C_i|}}{K} x100 \quad (3)$$

Doğruluk testi modelin ürettiği küme etiketlerini gerçek küme etiketleri ile karşılaştırarak modelin başarısını ölçen bir test yaklaşımıdır. a_i, i olarak etiketlenmiş ve gerçekte de i kümesine ait verilerin sayısı ve *n* toplam veri sayısı olmak üzere ortalama Doğruluk Eş. 4 ile hesaplanır.

$$Doğruluk = \frac{\sum_{i=1}^{k} a_i}{n} x100 \quad (4)$$

F-Skor (F-Score - F-Measure), Kesinlik (Precision) ve Duyarlılık (Recall) değerlerinin harmonik ortalamasıdır. $|R|$ ilgili dokümanların sayısı, $|A|$ *R*'ye ait olduğu tahmin edilen dokümanların sayısı ve $|R_a|$, *R*'ye ait olduğu tahmin edilenlerden gerçekte de *R*'ye ait olan dokümanların sayısı olmak üzere Kesinlik ve Duyarlılık değerleri Eş. 5 ve Eş. 6 ile hesaplanırken F-Skor da Eş. 7 ile hesaplanır.

$$Kesinlik = \frac{|R_a|}{A} \quad (5)$$

$$Duyarlılık = \frac{|R_a|}{|R|} \quad (6)$$

$$F - Skor = \frac{2.Precision.Recall}{Precision+Recall} x100 \quad (7)$$

Silhouette İndeks, kümeleme başarısını farklı kümelere ait verilerin birbirinden ne kadar uzak ve aynı kümeye ait verilerin birbirine ne kadar yakın olduğuna bakarak ölçer.

Tablo 7. DPStream algoritmasında kullanılan parametreler (Test parameters of DPStream algorithm) [61]

Veri setleri						
Parametreler	KDD	Fisher Iris	Breast Cancer	MrData	ExclaStar	IdealData
percent	2	5	2	2	5	5
LocalR	1,6	1,2	10	4,8	4,8	5
GlobalR	0,05	0,6	0,2	0,3	0,1	0,5
bufferSize	50	25	30	50	20	40
SketchIndex	0,90	0,70	0,75	0,90	0,75	0,8
startBufferSize	1000	25	100	1000	180	40
HalfLife	5	5	5	5	5	5
RemovePop	0,5	0,5	0,5	0,5	0,5	0,5

Tablo 8. Kullanılan veri setleri (Datasets used)

Veri seti	Türü	Örnek sayısı	Nitelik sayısı	Sınıf sayısı	Açıklama
KDD	Gerçek	50.000	38	23	Ağ saldırı verisi
Fisher Iris	Gerçek	150	4	3	İris bitkisinin türleri
Breast Cancer	Gerçek	699	9	2	Meme kanseri verisi
MrData	Sentetik	42470	2	4	Sapan veri içeren bir veri seti
ExclaStar	Sentetik	755	2	3	Evrimsel değişimi test etmek için uygun
IdealData	Sentetik	211	2	5	Kendi ürettiğimiz veri seti
Occupancy	Gerçek	8143	7	2	Oda doluluğu ile ilgili zaman damgalı veri seti

[1] *the UC Irvine Machine Learning Repository - https://archive.ics.uci.edu/ml/index.php*

Elde edilen sonuç [-1, 1] aralığında olur. -1 kümeleme başarısının minimum ve +1 kümeleme başarısının maksimum olduğunu gösterir. *a*, *i* verisinin ait olduğu kümenin diğer elemanlarına olan uzaklıklarının ortalaması ve *b* de *i* verisinin ait olmadığı en yakın kümenin elemanlarına olan uzaklıkların ortalaması olmak üzere S_i, ortalama Silhouette İndeks Eş. 8 ile hesaplanır.

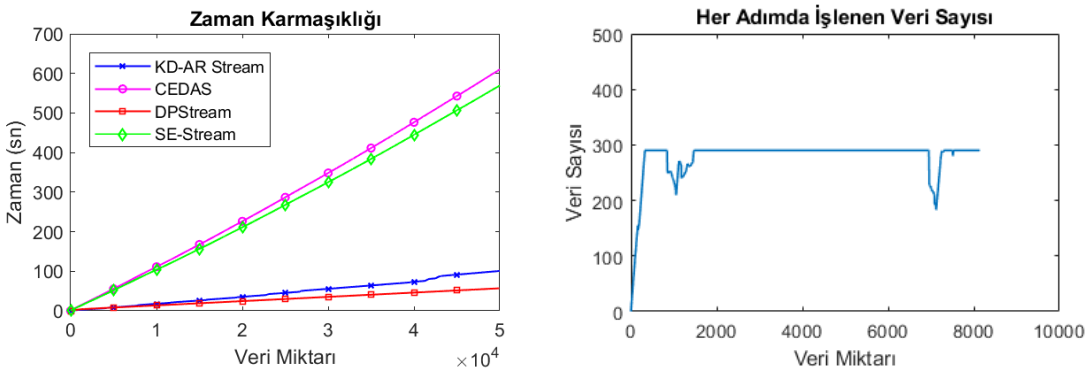
$$S_i = \frac{b-a}{\text{enb}(a,b)} \tag{8}$$

4. TARTIŞMA (DISCUSSION)

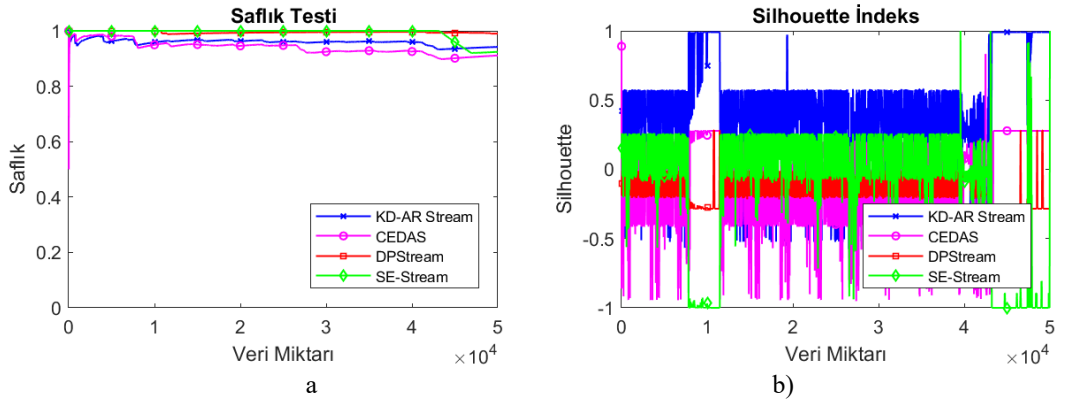
Algoritmaların başarılarını karşılaştırmak için KDD veri setinin 50000 verilik bir kısmı kullanılmış ve Şekil 9a, 10 ve 11'de görülen sonuçlar elde edilmiştir. Grafiklerden de görülebileceği gibi KD-AR Stream'in, Saflık testinde DPStream ve SE-Stream algoritmaları dışında bütün testlerde en iyi sonucu ürettiği görülmektedir. DPStream ve SE-Stream algoritmaları neredeyse tüm verileri tek bir kümeye atadığından Saflık değerleri yüksek çıkmaktadır. Bu nedenle tek bir test yöntemi üzerinden değerlendirme yapmak doğru değildir. Şekil 10b'de bulunan Silhouette İndeks sonuçları karşılaştırıldığında KD-AR Stream algoritmasının çok daha iyi sonuç verdiği görülmektedir. Bu da KD-AR Stream algoritmasının küme etiketinden bağımsız olarak kümeleri daha iyi ayrıştırdığını gösterir.

Şekil 11a ve 11b'de görülen Doğruluk ve F-Skor grafiklerine baktığımız zaman KD-AR Stream algoritması dışındaki diğer algoritmalarda bazı bölgelerde çok önemli düşüşler söz konusudur. Bu düşüşlerin nedeni diğer algoritmaların bu bölümde oluşması gereken kümeleri tespit edememesinden kaynaklanmaktadır. Zaten KDD veri seti özelinde baktığımız zaman 6 adet küme bulunmaktadır. KD-AR Stream bunların 5 tanesini tespit ederken diğer algoritmalar 3-4 tanesini tespit edebilmektedir.

Tablo 3'te de görüldüğü gibi KümeBöl gibi bazı fonksiyonların karmaşıklığı yüksektir. Ancak bu fonksiyonun çalışması belirli şartların oluşmasına (örneğin 2N tane veriye sahip olma ve minimum çekirdek yarıçapının r kadar olması gibi) bağlı olduğu için her adımda çalışmaz. Bu nedenle düşünüldüğü gibi performansı çok fazla düşürmemektedir. Zaten algoritmaların zaman karmaşıklığını karşılaştıran Şekil 9a'da da görüldüğü gibi KD-AR Stream oldukça makul bir sürede verileri işlemektedir.



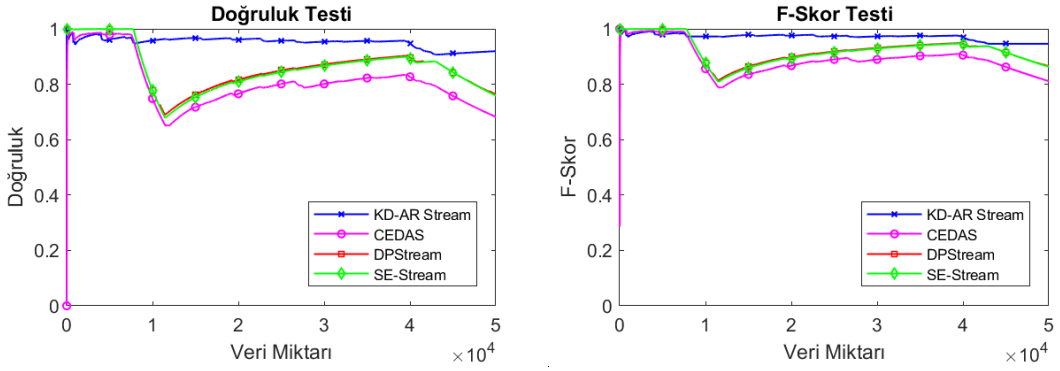
Şekil 9. Algoritmaların zaman karmaşıklıkları ve KD-AR Stream'in her adımda işlediği veri sayısı
(Performance comparison of algorithms and Buffered data size limitation example of KD-AR Stream on Occupancy dataset)



Şekil 10. Algoritmaların KDD veri seti üzerinde test edilmesiyle elde edilen Saflık ve Silhouette indeks değerlerinin karşılaştırılması
(Comparison of algorithms in terms of Purity and Silhouette index values on KDD dataset)

KD-AR Stream algoritması zaman tabanlı bir özetleme yaptığından belirlenmiş süre zarfında gelen veri sayısı beklenenden çok daha fazla olabilir. Dolayısıyla bu verileri işlemek çok zor olacaktır. Böyle bir durumda kullanıcının tanımladığı kadarlık kısmı özet olarak alınmaktadır. Şekil 9b'de her adımda işlenen veri miktarını gösteren grafikte de görüldüğü gibi veri miktarı çok fazla olduğunda algoritma en son gelen TN tanelik kısmını almaktadır. Grafikte de görüldüğü gibi t zamanda gelen veri bazen makul iken bazen eşik değerini aşmıştır. KD-AR Stream sapan verilere karşı dirençli bir algoritmadır. Tablo 9'da da görüldüğü gibi, %10'u sapan veri olan MrData veri setini %98'lik doğrulukla kümelemektedir. Bu da önerdiğimiz yaklaşımın sapan verilerin yaklaşık %98'ini doğru bir şekilde tespit edebildiğini göstermektedir.

Algoritma geliştirirken kullanıcının tanımlaması gereken değişken sayısının mümkün olduğunca az olması istenir. Tablo 4'te de görüldüğü gibi KD-AR Stream algoritmasında 6 değişkenin tanımlanması gerekmektedir. SE-Stream'in 8, CEDAS algoritmasının 3 ve DPStream algoritmasının 8 adet değişken kullandığı göz önüne alındığında KD-AR



a) (b)

Şekil 11. Algoritmaların KDD veri seti üzerinde test edilmesiyle elde edilen Doğruluk ve F-skor değerlerinin karşılaştırılması
(Comparison of algorithms in terms of Purity and Silhouette index values on KDD dataset)

Tablo 9. Algoritma başarılarının tek tabloda karşılaştırılması (Comparison of algorithms' successes in a single table)

		Saflık	Doğruluk	F-Skor	Silhouette İndeks
KD-AR Stream	KDD	95,86	95,14	97,07	0,52
	Fisher Iris	96,88	96,26	95,84	0,39
	Breast Cancer	90,05	90,05	91,90	0,56
	MrData	92,46	98,56	98,07	0,55
	ExclaStar	99,87	99,87	99,88	0,56
	IdealData	100	100	100	0,50
SE-Stream	KDD	99,21	85,80	92,15	-0,14
	Fisher Iris	9821	98,21	97,82	0,55
	Breast Cancer	95,19	58,39	73,14	-0,03
	MrData	91,55	38,78	9,78	-0,23
	ExclaStar	100	100	100	0,57
	IdealData	91,52	87,22	83,10	0,53
CEDAS	KDD	93,92	80,67	89,05	-0,04
	Fisher Iris	93,38	85,36	88,86	0,28
	Breast Cancer	96,85	57,86	73,15	-0,23
	MrData	89,68	39,37	55,73	-0,25
	ExclaStar	92,13	91,81	93,17	0,01
	IdealData	88,93	75,64	72,22	0,52
DPStream	KDD	99,51	86,19	92,34	-0,15
	Fisher Iris	97,67	72,04	80,37	0,12
	Breast Cancer	100	59,13	74,12	0,05
	MrData	92,49	39,37	55,87	-0,24
	ExclaStar	83,39	66,62	68,37	0,14
	IdealData	98,21	77,05	73,63	0,68

Stream algoritmasının makul sayıda değişken kullandığı söylenebilir.

KD-AR Stream evrimsel değişimi destekleyen bir algoritmadır. Bu nedenle zamana bağlı olarak küme yapılarında oluşan değişimleri başarıyla gerçekleştirebilmektedir. Bu özellik kümeleme başarısını arttırmaktadır. Tablo 9'da da görüldüğü gibi farklı testlerin büyük çoğunluğunda KD-AR Stream diğer algoritmalardan daha iyi sonuç vermektedir.

5. SONUÇLAR VE TARTIŞMALAR
(RESULTS AND DISCUSSIONS)

Bu çalışmada veriyi tamamen çevrimiçi işleyen, uyarlanabilir yarıçap özelliğine sahip ve zaman tabanlı bir özetleme yapan KD-AR Stream algoritması önerilmektedir. Yapılan deneysel çalışmalar, diğer algoritmalarla kıyaslandığında, KD-AR Stream'in makul bir sürede daha yüksek kümeleme başarısı gösterdiğini ortaya koymaktadır. Tamamen çevrimiçi çalışması, zaman tabanlı özetleme yapması, evrimsel bir algoritma olması ve yarıçap gibi parametrelerin uyarlanabilir olması KD-AR Stream algoritmasının öne çıkan özellikleri olarak göze çarpmaktadır. Ayrıca zamanla sahip olduğu verileri kaybeden kümeleri silmek yerine pasif duruma getirmesi ve daha sonra bu tür kümelerin yeni veri almak suretiyle tekrar yeterli veriye sahip olması durumunda yeniden aktive etmesi KD-AR Stream'in, akan verinin değişken yapısını destekleyen önemli bir diğer özelliği olarak ön plana çıkmaktadır.

Gelecekte KD-AR Stream algoritmasının performansını daha da arttıracak ve parametrelerin otomatik seçilebileceği çalışmaların yapılması planlanmaktadır.

KAYNAKLAR (REFERENCES)

1. Oussous, A., et al., Big Data technologies: A survey. Journal of King Saud University - Computer and Information Sciences, 2018. 30(4): p. 431-448.
2. Martín, A., A.B.A. Julián, and F. Cos-Gayón, Analysis of Twitter messages using big data tools to evaluate and locate the activity in the city of Valencia (Spain). Cities, 2019. 86: p. 37-50.
3. Görmüş, S., H. Aydın, and G. Ulutaş, Nesnelerin interneti teknolojisi için güvenlik: Var olan mekanizmalar, protokoller ve yaşanılan zorlukların araştırılması. Journal of the Faculty of Engineering and Architecture of Gazi University, 33(4), 1247-1272, 2018.
4. AlNuaimi, N., et al., Streaming feature selection algorithms for big data: A survey. Applied Computing and Informatics, 2019.
5. Kanmaz, M. and M.A. Aydin, Kablosuz Sensör Ağlarda Konumlandırma Yöntemleri ve K-means++ Kümeleme Yöntemi ile Yeni Yaklaşım. Journal of the Faculty of Engineering and Architecture of Gazi University, 34(2), 975-986, 2019.
6. Antonellis, P., C. Makris, and N. Tsirakis, Algorithms for clustering clickstream data. Information Processing Letters, 2009. 109(8): p. 381-385.
7. Yin, C., L. Xia, and J. Wang. Application of an Improved Data Stream Clustering Algorithm in Intrusion Detection System. in Advanced Multimedia and Ubiquitous Engineering. 2017. Singapore: Springer Singapore.
8. Yin, C., L. Xia, and J. Wang. Data Stream Clustering Algorithm Based on Bucket Density for Intrusion Detection. in Advances in Computer Science and Ubiquitous Computing. 2018. Singapore: Springer Singapore.
9. Li, Z.Q., A New Data Stream Clustering Approach about Intrusion Detection. Advanced Materials Research, 2014. 926-930: p. 2898-2901.
10. Hendricks, D., Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. Pattern Recognition Letters, 2017. 97: p. 21-28.
11. Aggarwal, C.C., Data Streams: An Overview and Scientific Applications, in Scientific Data Mining and Knowledge Discovery: Principles and Foundations, M.M. Gaber, Editor. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 377-397.
12. King, R.C., et al., Application of data fusion techniques and technologies for wearable health monitoring. Medical Engineering & Physics, 2017. 42: p. 1-12.
13. Gravina, R., et al., Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. Information Fusion, 2017. 35: p. 68-80.
14. Manzi, A., P. Dario, and F. Cavallo, A Human Activity Recognition System Based on Dynamic Clustering of Skeleton Data. Sensors (Basel, Switzerland), 2017. 17(5): p. 1100.
15. Diaz-Rozo, J., C. Bielza, and P. Larrañaga, Clustering of Data Streams with Dynamic Gaussian Mixture Models. An IoT Application in Industrial Processes. IEEE Internet of Things Journal, 2018: p. 1-1.
16. Tasnim, S., et al. Semantic-Aware Clustering-based Approach of Trajectory Data Stream Mining. in 2018 International Conference on Computing, Networking and Communications (ICNC). 2018.
17. Ankleshwaria, T.B. and J.S. Dhobi, Mining Data Streams: A Survey. International Journal of Advance Research in Computer Science and Management Studies, 2014. 2(2): p. 379-386.
18. Ikonomovska, E., S. Loskovska, and D. Gjorgjevik, A survey of stream data mining, in Eighth International Conference with International Participation – ETAI 2007. 2007: Ohrid, Republic ofMacedonia.
19. Şenol, A. and H. Karacan, A Survey on Data Stream Clustering Techniques. European Journal of Science and Technology, 2018(13): p. 17-30.
20. Aggarwal, C.C., Data Streams: Models and Algorithms. 1 ed. Advances in Database Systems. 2007: Springer US.
21. Bifet, A. and R. Kirkby, Data stream mining a practical approach. 2009.

22. Hancer, E., Diferansiyel Gelişim Tabanlı Çoklu Bulanık Kernel Kümeleme. Journal of the Faculty of Engineering and Architecture of Gazi University, 34(3), 1281-1293, 2019.

23. O'Callaghan, L., et al. Streaming-data algorithms for high-quality clustering. in Proceedings 1st International Conference on Data Engineering. 2002. San Jose, CA, USA, USA: IEEE.

24. Keogh, E., et al. An online algorithm for segmenting time series. in Proceedings 2001 IEEE International Conference on Data Mining 2001. San Jose, CA, USA, USA: IEEE.

25. Khalilian, M., N. Mustapha, and N. Sulaiman, Data stream clustering by divide and conquer approach based on vector model. Journal of Big Data, 2016. 3(1): p. 1.

26. Aggarwal, C.C., et al., A framework for clustering evolving data streams, in Proceedings of the 29th international conference on Very large data bases - Volume 29. 2003, VLDB Endowment: Berlin, Germany. p. 81-92.

27. Charu, C.A., et al., A framework for projected clustering of high dimensional data streams, in Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. 2004, VLDB Endowment: Toronto, Canada. p. 852-863.

28. Zhang, T., R. Ramakrishnan, and M. Livny, BIRCH: an efficient data clustering method for very large databases. SIGMOD Rec., 1996. 25(2): p. 103-114.

29. Karypis, G., E.-H. Han, and V. Kumar, Chameleon: Hierarchical Clustering Using Dynamic Modeling. Computer, 1999. 32(8): p. 68-75.

30. Udommanetanakit, K., T. Rakthanmanon, and K. Waiyamai. E-Stream: Evolution-Based Technique for Stream Clustering. 2007. Berlin, Heidelberg: Springer Berlin Heidelberg.

31. Rodrigues, P.P., J. Gama, and J. Pedroso, Hierarchical Clustering of Time-Series Data Streams. IEEE Transactions on Knowledge and Data Engineering, 2008. 20(5): p. 615-627.

32. Chairukwattana, R., et al. Efficient evolution-based clustering of high dimensional data streams with dimension projection. in 2013 International Computer Science and Engineering Conference (ICSEC). 2013.

33. Meesuksabai, W., T. Kangkachit, and K. Waiyamai. HUE-Stream: Evolution-Based Clustering Technique for Heterogeneous Data Streams with Uncertainty. 2011. Berlin, Heidelberg: Springer Berlin Heidelberg.

34. Yeh, M.Y., B.R. Dai, and M.S. Chen, Clustering over Multiple Evolving Streams by Events and Correlations. IEEE Transactions on Knowledge and Data Engineering, 2007. 19(10): p. 1349-1362.

35. Kranen, P., et al., The ClusTree: indexing micro-clusters for anytime stream mining. Knowledge and Information Systems, 2011. 29(2): p. 249-272.

36. Wang, W., J. Yang, and R.R. Muntz, STING: A Statistical Information Grid Approach to Spatial Data Mining, in Proceedings of the 23rd International Conference on Very Large Data Bases. 1997, Morgan Kaufmann Publishers Inc. p. 186-195.

37. Sheikholeslami, G., S. Chatterjee, and A. Zhang, WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. The VLDB Journal, 2000. 8(3): p. 289-304.

38. Agrawal, R., et al., Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec., 1998. 27(2): p. 94-105.

39. Tu, L. and Y. Chen, Stream data clustering based on grid density and attraction. ACM Trans. Knowl. Discov. Data, 2009. 3(3): p. 1-27.

40. Gao, J., et al. An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection. 2005. Berlin, Heidelberg: Springer Berlin Heidelberg.

41. Jia, C., C. Tan, and A. Yong. A Grid and Density-Based Clustering Algorithm for Processing Data Stream. in 2008 Second International Conference on Genetic and Evolutionary Computing. 2008.

42. Wan, L., et al., Density-based clustering of data streams at multiple resolutions. ACM Trans. Knowl. Discov. Data, 2009. 3(3): p. 1-28.

43. Dempster, A., N.M. Laird, and D.B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, in Paper presented at the Royal Statistical Society at a meeting organized by the Research Section. 1976.

44. Dang, X.H., et al. An EM-Based Algorithm for Clustering Data Streams in Sliding Windows. 2009. Berlin, Heidelberg: Springer Berlin Heidelberg.

45. Chaovalit, P. and A. Gangopadhyay, A method for clustering transient data streams, in Proceedings of the 2009 ACM symposium on Applied Computing. 2009, ACM: Honolulu, Hawaii. p. 1518-1519.

46. Choromanski, K., S. Kumar, and X. Liu, Fast Online Clustering with Randomized Skeleton Sets. CoRR, 2015. abs/1506.03425.

47. Ester, M., et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. 1996, AAAI Press: Portland, Oregon. p. 226-231.

48. Ankerst, M., et al., OPTICS: ordering points to identify the clustering structure. SIGMOD Rec., 1999. 28(2): p. 49-60.

49. Hinneburg, A. and D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining. 1998, AAAI Press: New York, NY. p. 58-65.

50. Ntoutsi, I., et al. Density-based Projected Clustering over High Dimensional Data Streams. in SIAM International Conference on Data Mining. 2012.

51. Amini, A. and T.Y. Wah, LeaDen-Stream: A Leader Density-Based Clustering Algorithm over Evolving Data Stream. Journal of Computer and Communications, 2013. 1: p. 26-31.

52. Hyde, R. and P. Angelov. A new online clustering approach for data in arbitrary shaped clusters. in 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF). 2015.

53. Mousavi, M. and A. Abu Bakar, Improved density based algorithm for data stream clustering. Jurnal Teknologi, 2015. 77(18): p. 73-77.

54. Ahmed, I., I. Ahmed, and W. Shahzad, Scaling up for high dimensional and high speed data streams: HSDStream. CoRR, 2015. abs/1510.03375.

55. Liu, L.x., et al. rDenStream, A Clustering Algorithm over an Evolving Data Stream. in 2009 International Conference on Information Engineering and Computer Science. 2009.

56. Cao, F., et al., Density-Based Clustering over an Evolving Data Stream with Noise, in Proceedings of the 2006 SIAM International Conference on Data Mining. 2006, Society for Industrial and Applied Mathematics. p. 328-339.

57. Ren, J. and R. Ma. Density-Based Data Streams Clustering over Sliding Windows. in 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery. 2009.

58. Hyde, R., P. Angelov, and A.R. MacKenzie, Fully online clustering of evolving data streams into arbitrarily shaped clusters. Information Sciences, 2017. 382-383: p. 96-114.

59. Chaoji, V., et al. SPARCL: Efficient and Effective Shape-Based Clustering. in 2008 Eighth IEEE International Conference on Data Mining. 2008.

60. Cao, F., et al., Density-Based Clustering over an Evolving Data Stream with Noise, in Proceedings of the 2006 SIAM International Conference on Data Mining. p. 328-339.

61. Xu, J., et al., Fat node leading tree for data stream clustering with density peaks. Knowledge-Based Systems, 2017. 120: p. 99-117.

62. Badiozamany, S., K. Orsborn, and T. Risch, Framework for real-time clustering over sliding windows, in Proceedings of the 28th International Conference on Scientific and Statistical Database Management. 2016, ACM: Budapest, Hungary. p. 1-13.

63. Hahsler, M. and M. Bolaños, Clustering Data Streams Based on Shared Density between Micro-Clusters. IEEE Transactions on Knowledge and Data Engineering, 2016. 28(6): p. 1449-1461.

64. Guha, S., R. Rastogi, and K. Shim, Cure: an efficient clustering algorithm for large databases. Information Systems, 2001. 26(1): p. 35-58.

65. Aggarwal, C., Y. Zhao, and P. Yu, On Clustering Graph Streams, in Proceedings of the 2010 SIAM International Conference on Data Mining. 2010, Society for Industrial and Applied Mathematics. p. 478-489.

66. Chen, J., P. Chen, and X.g. Sheng, A Sketch-based Clustering Algorithm for Uncertain Data Streams. JNW, 2013. 8: p. 1536-1542.

67. Yogita, D.T. A Novel Rough Set Based Clustering Approach for Streaming Data. in Second International Conference on Soft Computing for Problem Solving (Scrods 2012). 2012. New Delhi: Springers, New Delhi.

68. Mirsky, Y., et al. pcStream: A Stream Clustering Algorithm for Dynamically Detecting and Managing Temporal Contexts. in Advances in Knowledge Discovery and Data Mining. 2015. Cham: Springer International Publishing.

69. Silva, J.d.A., et al., An evolutionary algorithm for clustering data streams with a variable number of clusters. Expert Syst. Appl., 2017. 67(C): p. 228-238.

70. Laohakiat, S., S. Phimoltares, and C. Lursinsap, A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction. Information Sciences, 2017. 381: p. 104-123.

71. Shao, X., M. Zhang, and J. Meng. Data Stream Clustering and Outlier Detection Algorithm Based on Shared Nearest Neighbor Density. in 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS). 2018.

72. Shao, J., et al., Synchronization-based clustering on evolving data stream. Information Sciences, 2018.

73. Carnein, M. and H. Trautmann, evoStream – Evolutionary Stream Clustering Utilizing Idle Times. Big Data Research, 2018. 14: p. 101-111.

74. Reddy, K.S.S. and C.S. Bindu, StreamSW: A Density-based Approach for Clustering Data Streams over Sliding Windows. Measurement, 2018.

75. Ahmed, M., Buffer-based Online Clustering for Evolving Data Stream. Information Sciences, 2019.

76. Putri, G.H., et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data. Knowledge-Based Systems, 2019.

77. Bentley, J.L., Multidimensional binary search trees used for associative searching. Commun. ACM, 1975. 18(9): p. 509-517.

78. Ye, Y. Spatial data structure: the K-D tree. 10 May 2018]; Spatial data structure: the K-D tree]. Available from: http://homes.sice.indiana.edu/yye/lab/teaching/spring2014-C343/moretrees.php.

79. Kreveld, M.v. and W.v. Toll. Computational Geometry - Lecture 7: Range searching and kd-trees. 2018 12 January 2018]; Lecture Notes]. Available from: http://www.cs.uu.nl/docs/vakken/ga/slides5a.pdf.

80. Rodriguez, A. and A. Laio, Clustering by fast search and find of density peaks. Science, 2014. 344(6191): p. 1492.