



T.C  
SAKARYA ÜNİVERSİTESİ  
**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**  
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

# BSM 310 – YAPAY ZEKA

## **Grup üyeleri:**

Zeynep Burcu BAŞTUĞ b171210043

Beyza KARACA b161210367

Canan TOKYAY b161210061

Ece Nur ARSLAN b171210061

Hilal YILDIZ b171210019

**Sakarya**

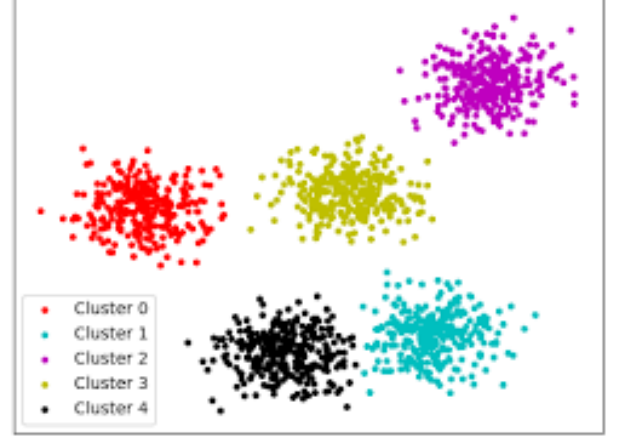
**2020**

## K-Ortalama Kümeleme Algoritması (K-Means)

İlk olarak 1967’de James MacQueen tarafından kullanılmıştır.Kümeleme, benzer özellikler gösteren verilerin gruplara ayrılması demektir. Çok sade bir algoritma olduğu için çok fazla kullanılır. Kullanım alanları :

-Belge sınıflandırma: Belgeleri etiketlerine, konularına, içeriklerine göre sınıflandırma işlemidir.

-Suç Yerleri Tespitinde: Suç kategorileri,şehirlerdeki suç bölgeleri,suç eğilim oranları ve suça eğilimli insan tipleri gibi kümelemeler için kullanılır.



-Müşteri Segmentasyonu: Pazarlamacılar için müşteri tanıma kısmında, müşterilerin alım seçeneklerini belirlemede, ilgi alanlarının tespitinde kullanılır.

\*Özellikle markalara ait kampanya düzenleneceği zaman bilgi analizi için kullanılmaktadır.

Bu ve burda yazılmayan bir çok alanda K-ortalama kümeleme algoritması kullanılmaktadır.

Bu algoritma gözetimsiz öğrenme ve bölümlmeli kümeleme algoritmasıdır.

Gözetimsiz algoritma, denetleme yapılmadan uygulanan bir modeldir.Etiketlenmemiş verilerle ilgilenir.Temel prensip olarak model, bilgiyi kendi çalışmalarıyla keşfeder.

Avantajları:

- Verilerde her türlü bilinmeyen kalıplar bulunabilir.
- Kategorizasyon için gerekli olabilecek özellikleri bulabilmeye yardımcı olur.
- Gerçek zamanlı olarak gerçekleşir.
- Etiketlenmemiş verilerle uğraşmak manuel olarak zordur.Bu zorluğu basitleştirir.

Bölümlmeli kümeleme ise, her verinin sadece bir kümeye ait olmasını ifade eder.

## K-Means Algoritmasının Amacı Nedir?

K-Means algoritmasının amacı, kümeler arası benzerliğin en az düzeyde, küme içi benzerliğin en üst düzeyde olmasıdır. Küme benzerliği, kümenin merkezi kabul edilen birim ile kümedeki diğer birimler arası uzaklığın ortalamasıdır.

## K-Means Algoritma Nasıl Çalışır?


Temel olarak 4 adımdan oluşur:

1. Seçilen merkez sayısı kadar küme merkezinin belirlenmesi
2. Merkez dışında kalan örneklerin sınıflandırılması
3. Eski merkezlerin yeni seçilen merkezlere kaydırılması
4. Bu adımların, kararlı hale gelene kadar tekrarı

## Bilgileri Okuma


```
def ReadData(fileName):  
  
    # Read the file, splitting by lines  
    f = open(fileName, 'r');  
    lines = f.read().splitlines();  
    f.close();  
  
    items = [];  
  
    for i in range(1, len(lines)):  
        line = lines[i].split(',');  
        itemFeatures = [];  
  
        for j in range(len(line)-1):  
            v = float(line[j]); # Convert feature value to float  
            itemFeatures.append(v); # Add feature value to dict  
  
        items.append(itemFeatures);  
  
    shuffle(items);  
  
    return items;
```

## Min-Max Bulma




```
def FindColMinMax(items):  
    n = len(items[0]);  
    minima = [sys.maxint for i in range(n)];  
    maxima = [-sys.maxint -1 for i in range(n)];  
  
    for item in items:  
        for f in range(len(item)):  
            if (item[f] < minima[f]):  
                minima[f] = item[f];  
  
            if (item[f] > maxima[f]):  
                maxima[f] = item[f];  
  
    return minima,maxima;
```

## Kümelerde Karşılık Gelen Kısımlara Atama




```
def InitializeMeans(items, k, cMin, cMax):  
  
    # Initialize means to random numbers between  
    # the min and max of each column/feature  
    f = len(items[0]); # number of features  
    means = [[0 for i in range(f)] for j in range(k)];  
  
    for mean in means:  
        for i in range(len(mean)):  
  
            # Set value to a random float  
            # (adding +-1 to avoid a wide placement of a mean)  
            mean[i] = uniform(cMin[i]+1, cMax[i]-1);  
  
    return means;
```

## Benzerlik Metriği




```
def EuclideanDistance(x, y):  
    S = 0; # The sum of the squared differences of the elements  
    for i in range(len(x)):  
        S += math.pow(x[i]-y[i], 2);  
  
    return math.sqrt(S); #The square root of the sum
```

## Merkezlerin Yenilenmesi




```
def UpdateMean(n,mean,item):  
    for i in range(len(mean)):  
        m = mean[i];  
        m = (m*(n-1)+item[i])/float(n);  
        mean[i] = round(m, 3);  
  
    return mean;
```

## En Yakın Merkeze Atama



```
def Classify(means,item):  
  
    # Classify item to the mean with minimum distance  
    minimum = sys.maxint;  
    index = -1;  
  
    for i in range(len(means)):  
  
        # Find distance from item to mean  
        dis = EuclideanDistance(item, means[i]);  
  
        if (dis < minimum):  
            minimum = dis;  
            index = i;  
  
    return index;
```

## K girdisi, ögeler,max yineleme sayısı alınır kümeler döndürülür



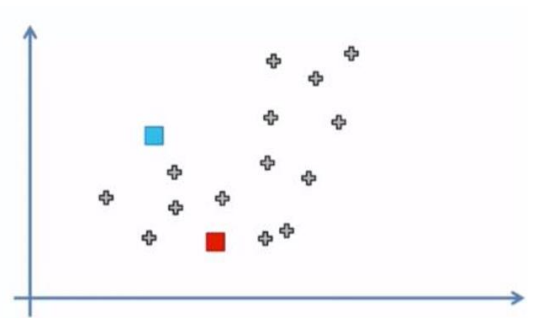
```
def CalculateMeans(k,items,maxIterations=100000):  
  
    # Find the minima and maxima for columns  
    cMin, cMax = FindColMinMax(items);  
  
    # Initialize means at random points  
    means = InitializeMeans(items,k,cMin,cMax);  
  
    # Initialize clusters, the array to hold  
    # the number of items in a class  
    clusterSizes= [0 for i in range(len(means))];  
  
    # An array to hold the cluster an item is in  
    belongsTo = [0 for i in range(len(items))];  
  
    # Calculate means  
    for e in range(maxIterations):  
  
        # If no change of cluster occurs, halt  
        noChange = True;  
        for i in range(len(items)):  
  
            item = items[i];  
  
            # Classify item into a cluster and update the  
            # corresponding means.  
            index = Classify(means,item);  
  
            clusterSizes[index] += 1;  
            cSize = clusterSizes[index];  
            means[index] = UpdateMean(cSize,means[index],item);  
  
            # Item changed cluster  
            if(index != belongsTo[i]):  
                noChange = False;  
  
            belongsTo[i] = index;  
  
        # Nothing changed, return  
        if (noChange):  
            break;  
  
    return means;
```

## Kümeleri Bulma

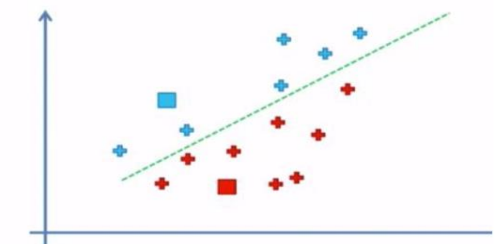
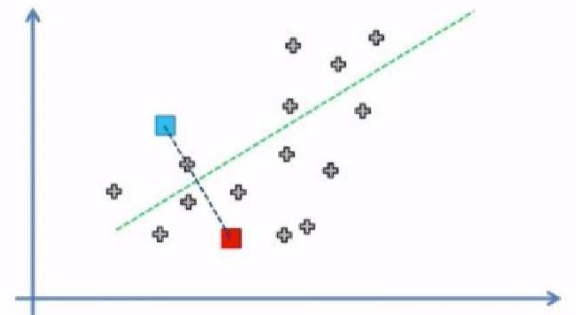
```
def FindClusters(means, items):  
    clusters = [[] for i in range(len(means))]; # Init clusters  
  
    for item in items:  
        # Classify item into a cluster  
        index = Classify(means, item);  
  
        # Add item to cluster  
        clusters[index].append(item);  
  
    return clusters;
```

## Anlatım Modeli

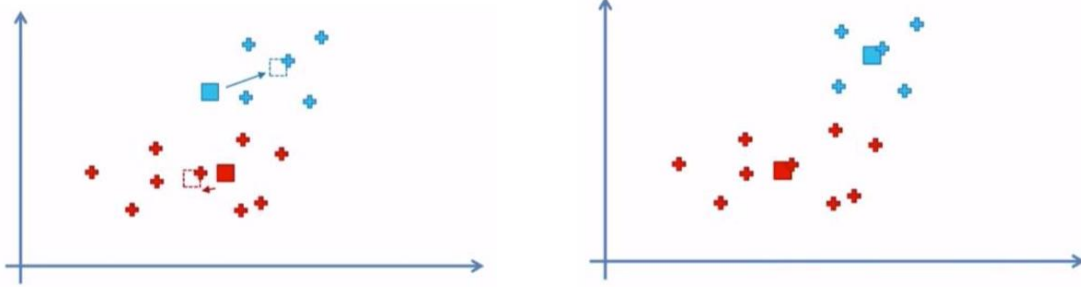
Yandaki gibi bir veri setinde kullanıcıdan K değeri olarak 2 aldığımızı varsayalım. Bu durumda algoritma bu uzay içerisinde rastgele iki nokta belirleyecektir.



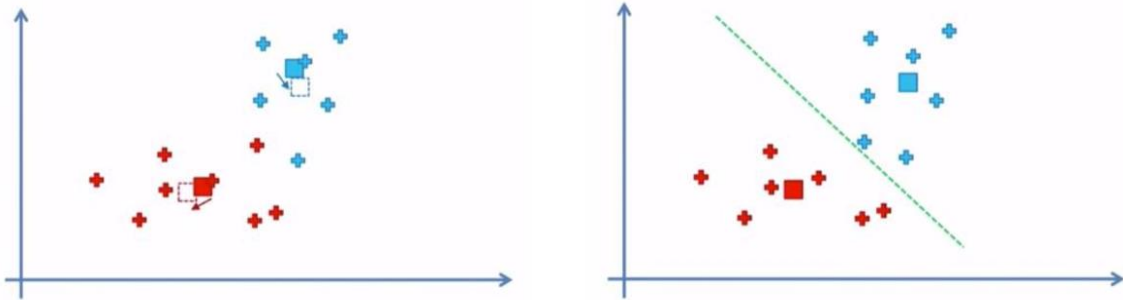
Sonrasında her veri noktası kendisine en yakın olan merkez noktasına bağlı kümeye atanmalıdır. Bu örnekte merkez noktalarının birbiriyle uzaklıklarının orta dikmesinin hangi tarafında kaldığına göre veriler ikiye ayrılmış ve kendilerine en yakın olan merkez noktasının bulunduğu kümeye dahil edilmişlerdir.



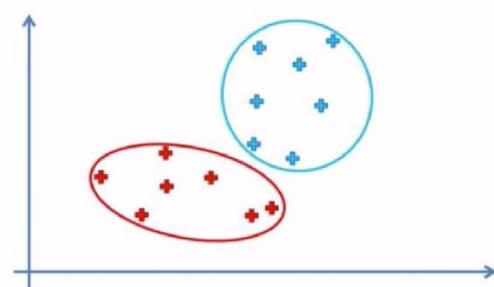
Veriler ilk kez kümelendirildikten sonra her kümenin kendi içinde ağırlık noktası bulunur. Ağırlık noktası kümelerin yoğunluklu olarak bulunduğu noktadır. Bu noktalar kümelerin yeni merkez noktaları olur.



Algoritma bu şekilde kendisini tekrar ederek merkez noktaları kararlı hale gelene kadar onları bulundukları kümenin ağırlık merkezine taşıyıp kümelemeyi yeniler.



Merkez noktaları artık değişmemeye başladığında yani kararlı hale (stable state) geldiklerinde veriler en optimum şekilde kümelenebilir demektir.





## Algoritmaya Rassal Başlangıçla Başlamanın Dezavantajları

- Veriler istenilen şekilde dağılmayabilir.
- Kümeler istenilen şekilde oluşmayabilir.
- Tekrar eden adım sayısı uzayabilir.
- Kararlı hale gelme durumunda bile istenilen şekilde kümeleme oluşmayabilir.

## K-Means++

Başlangıç merkez noktalarının rastgele seçilmesinden kaynaklanan bu hataya çözüm olarak geliştirilen bazı algoritmalar vardır. K-Means++ da bu algoritmalarından biridir.

Adım 1: Rastgele seçilen merkez noktalardan en yakınına her noktadan uzaklık hesaplanır. (Buna  $D(x)$  diyelim)

Adım 2: Yeni noktalar mesafenin karesini olasılık olarak  $(D(x)^2)$  bulunur.

Algoritma tıpkı K-Means gibi başlar. Rastgele olarak merkez noktaları dağıtılır. Sonra her data pointten kendisine en yakın merkez noktasına bir mesafe hesaplanması yapılır. Bu her data point için tek tek gerçekleştirilir. Data pointin merkeze uzaklığına  $D(x)$  diyoruz. Daha sonra merkez noktaları veri noktalarından  $D(x)^2$  değeri en yüksek olanlara atanır. Yani merkez noktalarına en uzak veri noktaları yeni merkez noktaları olur. K-Means++ gibi başka iyileştirme algoritmaları da mevcuttur.

## K Sayısına karar Vermek

Çok sayıda küme olması analizi zorlaştıran bir durumdur. Optimal küme sayısına karar verebilmek için bir veri bilimcinin yorumlaması daha iyidir. Fakat bunun nasıl yorumlanacağı, bu başarının nasıl ölçüleceği ve en optimum K değerinin nasıl seçileceği ile ilgili çözümler mevcuttur. Bunlardan biri de WCSS hesaplaması yapmaktır.

## WCSS Hesaplaması

Her bir kümenin veri noktalarının merkez noktalarına uzaklığının karelerinin toplamı toplanır ve buna WCSS değeri denir. Mesafenin karesi bir kriter olarak

verilir, mesafe uzadıkça karesi daha hızlı uzar. İstenen, uzaktaki noktaları çok daha hızlı tespit etmektir.

Wcss değeri küçük olmalıdır. Fakat Wcss değerini küçültmek isterken k değeri arttırılacaktır. Bu durum her verinin bir merkez olması(K veri adedi kadar Wcss=0) ile sonlanır. Aynı zamanda bu yapay zekada istenmeyen ezberleme şeklidir.

```
wcss = []
kume_sayisi_listesi = range(1, 11)
for i in kume_sayisi_listesi :
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

## Dirsek Metodu Grafiği

```
plt.plot(kume_sayisi_listesi, wcss)
plt.title('Küme Sayısı Belirlemek için Dirsek Yöntemi')
plt.xlabel('Küme Sayısı')
plt.ylabel('WCSS')
plt.show()
```



Grafikte ani eğimin olduğu kısım bulunarak, k sayısı belirlenir.

## Kaynaklar

Yüksek Lisans Tezi KÜMELEME ANALİZİNDE KÜME SAYISININ BELİRLENMESİ  
ÜZERİNE BİR ÇALIŞMA Azize Celile GÜNAY ATBAŞ

Python ile Makine Öğrenmesi Sadi Evren Şeker Udemy

Ekrem Hatipoglu Machine Learning — Clustering ( Kümeleme )— K-MeansAlgorithm —Hierarchical Clustering ( Hiyerarşik Kümeleme )— Part 13

Veri Madenciliğinde Kümeleme Teknikleri Üzerine Bir Çalışma: K-Means ve K-Medoids Kümeleme Algoritmalarının Karşılaştırılması Güncel SARIMAN Süleyman Demirel Üniversitesi, Mühendislik Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü

<https://www.veribilimiokulu.com/kumeleme-notlari-4-k-ortalamalar-python-uygulama/>

<https://www.geeksforgeeks.org/k-means-clustering-introduction/>

<https://medium.com/deep-learning-turkiye/k-means-algoritmas%C4%B1-b460620dd02a>