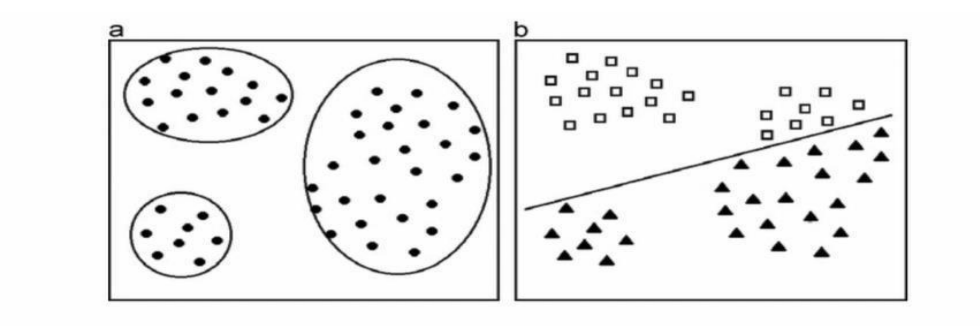
## 6.1. Kümeleme (Clustering)

Kümeleme, denetimsiz öğrenmenin bir yöntemidir ve birçok alanda kullanılan istatistiksel veri analizi için yaygın bir tekniktir. Denetimsiz öğrenme, veri kümesi ile çıktıların olmadığı bir öğrenme metodudur. Veri kümesindeki verileri yorumlayarak ortak noktaları bulmak ve bunları kümeleştirme işlemi yapılarak anlamlı bir veri elde edebilmektir. Sistem, öğreten olmadan öğrenmeye çalışır. Ham verileri organize verilere dönüştüren bir makine öğrenimi türüdür.

Denetimsiz öğrenmede sadece veriler vardır onlar hakkında bilgi verilmez. Bu verilerden sonuçlar çıkarılmaya çalışılır. En baştan veriler hakkında herhangi bir bilgi verilmediği için çıkartılan sonuçların kesinlikle doğru olduğu söylenemez. Veriyi değişkenler arasındaki ilişkilere dayalı olarak kümeleyerek çeşitli modeller, yapılar oluşturabiliriz.

Örneğin, bir alışveriş sitesinde alınan bir ürünün yanında kullanıcıların alabileceği diğer ürünlerin tavsiye olarak belirlenmesi. Ya da bir hizmet satın aldığında, o hizmetle etkileşimli diğer hizmetlerin müşterinin ilgi alanına girmesi.

Kümeleme işlevinden kimlik belirlenebilir mi? Özellikle alış veriş yaparken aldıklarımız, aynı anda kimliğimiz olabilir mi?




Kümeleme Sınıflandırma

Denetimsiz öğrenmede kümeleme söz konusudur. Kümeleme algoritmaları basitçe, veri kümesindeki elemanları kendi aralarında gruplamaya çalışır. Burada kaç grup olacağı veya en uygun küme sayısını algoritmanın kendisi belirler. Verilerin yakınlık, uzaklık, benzerlik gibi ölçütlere göre analiz edilerek sınıflara ayrılmasına kümeleme denir. Örnek; alışveriş yapılan bir marktte kasiyerin bir robot olduğun düşünün ve tüm ürünler birbirine karışmış olsun. Elma bulup, onu tanıyıp diğer elemanları yığın içerisinde topladğını düşünün. Seçme işlemi devam ederken yetenek kazanarak performansını atırabilir; hatalı seçtiği ürünler var ise ayıklayabilir. Böylece ürünlerin birbirlerine benzeme yakınlığı uzaklaşarak, ayrım yapma yeteneği artırılmış olur. Böylece sınılandırma da yapılmış olur.

Elmalar da kendi aralarında kümeleme yapılabilir mi? Aynı tipte verilerin değişik segmentlere bölünmüş halidir. Elinde örnekler var ama hangi veya kaç sınıfa ait olduğunu bilmiyor.

Sınıfları(kümeleri) kendisi inşaa ediyor. Örneğin elimizde sadece domatesler varsa, ve bunları kalitelerine göre ayırılırsa bu kümeleme işlemidir.

**Kümelemenin uygulama alanları:**
Tıp'da elde edilen görüntülemeler üzerindeki farklıları analiz edilerek değişik nitelikler çıkartılabilir.

Suç Yerlerinin Belirlenmesi: Bir şehirdeki belirli bölgelerde mevcut olan suçlarla ilgili veriler, suç kategorisi, suç alanı ve ikisi arasındaki ilişki, bir şehirdeki ya da bölgedeki suça eğilimli alanlara ilişkin kaliteli bilgiler verebilir.

Oyuncu istatistiklerini analiz etmek: Oyuncu istatistiklerini analiz etmek, spor dünyasının her zaman kritik bir unsuru olmuştur ve artan rekabetle birlikte, makine öğrenmenin burada oynayacağı kritik bir rol vardır.

Çağrı Kaydı Detay Analizi: Bir çağrı detay kaydı (CDR), telekom şirketleri tarafından bir müşterinin araması, SMS ve internet etkinliği sırasında elde edilen bilgilerdir. Bu bilgiler, müşteri demografisiyle birlikte kullanıldığında, müşterinin ihtiyaçları hakkında daha fazla bilgi sağlar.

**Müşteri Segmentasyonu:**
Collaborative Filtering: Davranışları birbirine benzeyen insanların yaptıklarına bakılarak kümedeki birinin ne yapacağını kestirmek.
Tehdit ve Sahtekarlık Yakalama: Sınıflandırmada tehditlerin özellikleri makineye öğretilir. Kümelemede ise kümelerin dışında kalan , herhangi bir kümeye girmeyen örnek bir tehdit unsuru olarak tanımlanabilir.
Eksik Verilerin Tanımlanması: Örneğin birinin maaş bilgisi eksikse segmentte benzer kişilerin maaş ortalamasına göre bir maaş hesaplanabilir.

**Pazar Segmentasyonu:**
Davranışsal Segmentasyon: Örneğin ücretsiz uygulamaları yükleyenler neyi seçiyiorlar?
Demografik Segmentasyon: Müşterilerin yaşı, cinsiyeti, eğitim düzeyi ile davranışlarının entegre edilmesi.
Psikolojik Segmentasyon: Müşterilerin hayal-beklentilerine göre farklı ürünler sunulabilir.
Coğrafi Segmentasyon: Ülkelere şehirlere göre vs.

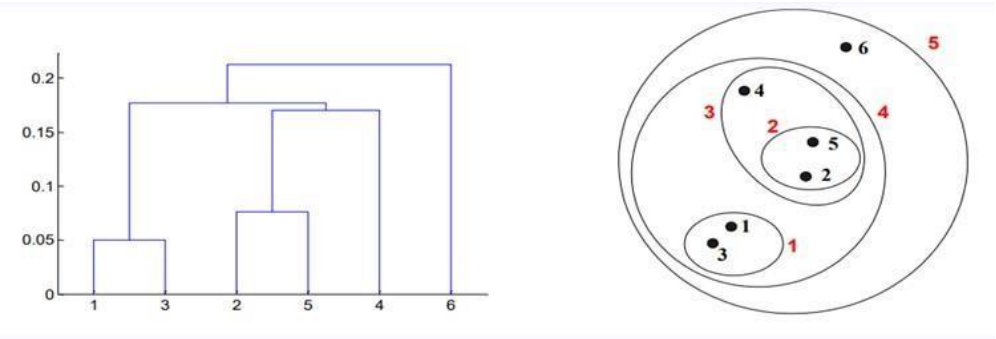
**Kümeleme Çeşitleri:**

- Hiyerarşik Kümeleme

- Gürültülü Uygulamaların Yoğunluğa Dayalı Konumsal Kümelenmesi (DBSCAN) (DBSCAN)
- K-means Kümeleme
- Ağırlık Ortalama Kaydırma Kümelemesi
- Gauss Karışım Modelleri (GMM) kullanarak Beklenti-Maksimizasyon (EM) Kümeleme

**Hiyerarşik Kümeleme:**

Hiyerarşik kümeleme algoritmaları 2 kategoriye ayrılır: yukarıdan aşağıya veya aşağıdan yukarıya. Aşağıdan yukarıya algoritmalar, her veri noktasını başlangıçta tek bir küme olarak ele alır ve ardından tüm kümeler tüm veri noktalarını içeren tek bir kümede birleştirilene kadar küme çiftlerini art arda birleştirir (veya toplar). Bu nedenle aşağıdan yukarıya hiyerarşik kümeleme, hiyerarşik kümelemeli kümeleme (hierarchical agglomerative clustering) veya HAC olarak adlandırılır. Bu küme hiyerarşisi bir ağaç (veya dendrogram) olarak temsil edilir. Ağacın kökü, tüm örnekleri toplayan benzersiz kümedir, yapraklar yalnızca bir örnek içeren kümelerdir. Algoritma adımlarına geçmeden önce bir örnek için aşağıdaki grafiğe bakın




- Her veri noktasını tek bir küme olarak ele alarak başlıyoruz, yani veri kümemizde X veri noktası varsa, o zaman X kümemiz var. Daha sonra iki küme arasındaki mesafeyi ölçen bir mesafe ölçüsü seçiyoruz. Örnek olarak, iki küme arasındaki mesafeyi, birinci kümedeki veri noktaları ile ikinci kümedeki veri noktaları arasındaki ortalama mesafe olarak tanımlayan ortalama bağlantıyı kullanacağız.
- Her yinelemede, iki kümeyi tek bir kümede birleştiriyoruz. Birleştirilecek iki küme, en küçük ortalama bağlantıya sahip olanlar olarak seçilir. Yani, seçtiğimiz uzaklık ölçütümüze göre, bu iki küme birbirleri arasındaki en küçük mesafeye sahiptir ve bu nedenle en benzer olanlardır ve birleştirilmeleri gerekir.
- Adım 2, ağacın köküne ulaşana kadar tekrar edilir, yani tüm veri noktalarını içeren tek bir kümeye sahibiz. Bu şekilde sonunda kaç tane küme istediğimizi seçebiliriz, sadece kümeleri birleştirmeyi ne zaman durduracağımızı seçerek, yani ağacı oluşturmayı bıraktığımızda!

Hiyerarşik kümeleme, küme sayısını belirlememizi gerektirmez ve hatta bir ağaç oluşturduğumuz için hangi küme sayısının en iyi görüneceğini seçebiliriz. Ek olarak, algoritma mesafe ölçüsü seçimine duyarlı değildir; hepsi eşit derecede iyi çalışma eğilimindeyken, diğer

kümeleme algoritmalarında uzaklık ölçüsü seçimi kritiktir. Hiyerarşik kümeleme yöntemlerinin özellikle iyi bir kullanım durumu, temeldeki verilerin hiyerarşik bir yapıya sahip olması ve hiyerarşiyi kurtarmak istemenizdir; diğer kümeleme algoritmaları bunu yapamaz. Hiyerarşik kümelemenin bu avantajları, K-Ortalamalarının ve GMM'nin doğrusal karmaşıklığından farklı olarak, O ($n^3$) zaman karmaşıklığına sahip olduğundan, daha düşük verimlilik pahasına gelir.

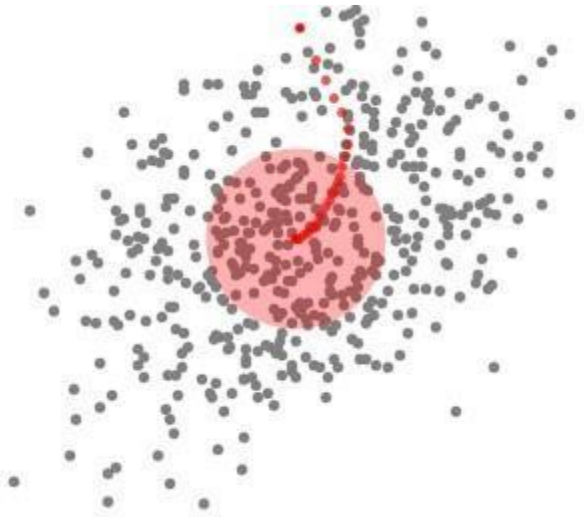
**K-Means Algoritması:**
K-means algoritmasında kullanılan örneklem, k adet kümeye bölünür. Algoritmanın özü birbirlerine benzerlik gösteren verilerin aynı küme içerisine alınmasına dayanır. Algoritmadaki benzerlik terimi, veriler arasıdaki uzaklığa göre belirlenmektedir. Uzaklığın az olması benzerliğin yüksek, çok olması ise düşük olduğu anlamına gelmektedir. K-means algoritmasının yapısı aşağıdaki gibidir;

1) K adet rastgele küme oluştur
2) Kare hata oranını hesapla
3) Verilerin kümelerin orta noktalarına olan uzaklıklarını bul
4) Her veri için en yakın kümeyi, o verinin kümesi olarak belirle
5) Yeni yerleşim düzenine göre hata oranını hesapla
6) Eğer önceki hata oranı ile şimdiki hata oranı eşit değilse 2,3,4,5 ve 6. adımları tekrarla
7) Eğer önceki hata oranı ile şimdiki hata oranı eşitse kümeleme işlemini sonlandır

Dirsek yöntemi, kümelere nasıl ihtiyaç duyacağımız konusunda kullanışlı olur. Dirsek noktasının belirlenmesi gerekir.

**Ağırlık Ortalama Kaydırma Kümelemesi:**
Ortalama kaydırma kümeleme, veri noktalarının yoğun alanlarını bulmaya çalışan kayan pencere tabanlı bir algoritmadır. Centroid tabanlı bir algoritmadır, yani amacın her bir grubun / sınıfın merkez noktalarını bulmaktır, bu da kayan pencere içindeki noktaların ortalaması olacak merkez noktaları için adayları güncelleyerek çalışır. Bu aday pencereler daha sonra, neredeyse kopyaları ortadan kaldırmak için bir işlem sonrası aşamasında filtrelenir ve nihai merkez noktaları ve bunlara karşılık gelen grupları oluşturur.

Sürgülü pencerelerin tümü ile uçtan uca tüm sürecin bir örneği aşağıda gösterilmiştir. Her siyah nokta, kayan bir pencerenin merkezini temsil eder ve her gri nokta bir veri noktasıdır.



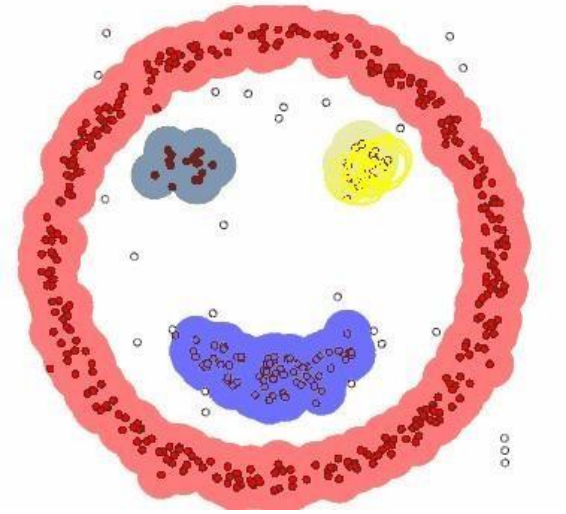
1) Ortalama-kaymayı açıklamak için, yukarıdaki resimde olduğu gibi iki boyutlu uzayda bir dizi noktayı ele alacağız. Bir C noktasında ortalanmış (rastgele seçilmiş) ve çekirdek olarak r yarıçapına sahip dairesel bir kayan pencere ile başlıyoruz. Ortalama kayma, bu çekirdeği yakınsamaya kadar her adımda yinelemeli olarak daha yüksek yoğunluklu bir bölgeye kaydırmayı içeren bir tepe tırmanma algoritmasıdır.

2) Her yinelemede, kayan pencere, merkez noktası pencere içindeki noktaların ortalamasına kaydırılarak (dolayısıyla adı) daha yüksek yoğunluklu bölgelere kaydırılır. Sürgülü pencere içindeki yoğunluk, içindeki noktaların sayısı ile orantılıdır. Doğal olarak, penceredeki noktaların ortalamasına geçerek, yavaş yavaş daha yüksek nokta yoğunluğuna sahip alanlara doğru hareket edecektir.

3) Bir kaymanın çekirdek içinde daha fazla noktayı barındırabileceği bir yön olmayana kadar ortalamaya göre kayan pencereyi kaydırmaya devam ediyoruz. Yukarıdaki grafiğe bakın; Artık yoğunluğu artırmayana kadar (yani penceredeki nokta sayısı) daireyi hareket ettirmeye devam ediyoruz.

4) 1'den 3'e kadar olan bu adım süreci, tüm noktalar bir pencerenin içinde kalana kadar birçok sürgülü pencerede yapılır. Birden çok sürgülü pencere örtüştüğünde, en çok noktayı içeren pencere korunur. Veri noktaları daha sonra bulundukları kayan pencereye göre kümelenir.

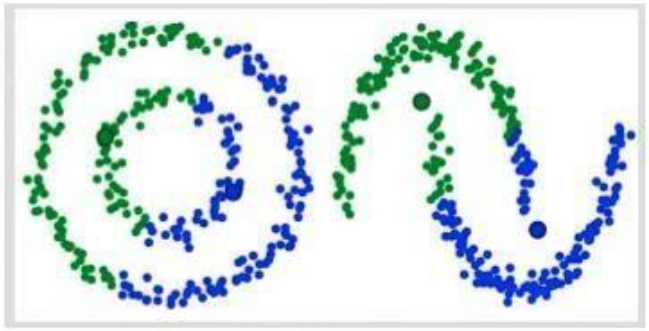
**Gürültülü Uygulamaların Yoğunluğa Dayalı Konumsal Kümelenmesi (DBSCAN):**
DBSCAN, ortalama kaymaya ekseninde, benzer ve yoğunluklu bölgeleri kümeleyen bir algoritmadır, ancak birkaç önemli avantajı vardır. Minimum bölge sayısında ve uzaklıkta maksimum yoğunluk bölgesi oluşturulması hedeflenir.



- DBSCAN, keyfi bir başlangıç veri noktasıyla başlar. Bu noktanın komşuluğu bir epsilon ε uzaklığı kullanılarak çıkarılır (ε mesafesi içindeki tüm noktalar komşuluk noktalarıdır).
- Bu mahalle içinde yeterli sayıda nokta (minPoints'e göre) varsa, kümeleme işlemi başlar ve mevcut veri noktası yeni kümedeki ilk nokta olur. Aksi takdirde, nokta gürültü olarak etiketlenecektir (daha sonra bu gürültülü nokta kümenin parçası haline gelebilir). Her iki durumda da bu nokta "ziyaret edildi" olarak işaretlenir.
- Yeni kümedeki bu ilk nokta için, ε mesafesi komşuluğundaki noktalar da aynı kümenin parçası olur. ε mahallesindeki tüm noktaları aynı kümeye ait yapma prosedürü, küme grubuna yeni eklenen tüm yeni noktalar için tekrarlanır.
- 2. ve 3. adımlardan oluşan bu süreç, kümedeki tüm noktalar belirlenene kadar, yani kümenin ε mahallesindeki tüm noktalar ziyaret edilip etiketlenene kadar tekrar edilir. Mevcut kümeyle işimiz bittiğinde, yeni bir ziyaret edilmemiş nokta alınır ve işlenir, bu da başka bir küme veya gürültü keşfine yol açar. Bu işlem, tüm noktalar ziyaret edildi olarak işaretlenene kadar tekrar eder.
- Bunun sonunda tüm noktalar ziyaret edildiğinden, her nokta bir kümeye ait veya gürültü olarak işaretlenecektir.

DBSCAN, diğer kümeleme algoritmalarına göre bazı büyük avantajlar sunar. İlk olarak, hiç bir küme gerektirmez. Ayrıca, veri noktası çok farklı olsa bile, aykırı değerleri basitçe bir kümeye atan ortalama kaymanın aksine, gürültü olarak tanımlar. Ek olarak, keyfi olarak boyutlandırılmış ve keyfi olarak şekillendirilmiş kümeleri oldukça iyi bulabilir. DBSCAN'ın ana dezavantajı, kümeler farklı yoğunlukta olduğunda diğerleri kadar iyi performans göstermemesidir. Bunun nedeni, komşu noktaların belirlenmesi için mesafe eşiği ε ve minPoints'in, yoğunluk değiştiğinde kümeden kümeye değişmesidir. Bu dezavantaj, çok yüksek boyutlu verilerde de ortaya çıkar, çünkü yine mesafe eşiğini ε tahmin etmek zorlaşır.

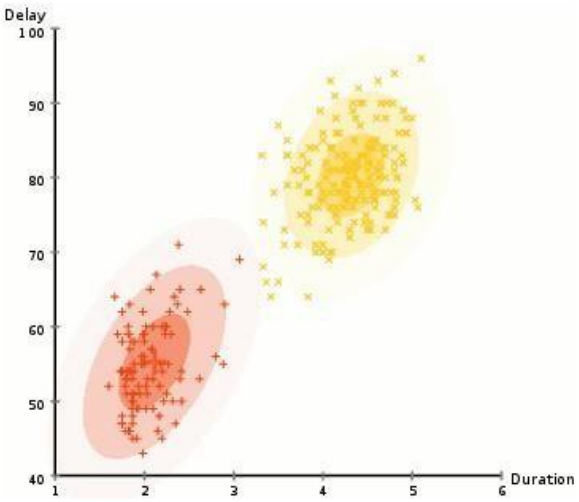
**Gauss Karışım Modelleri (GMM) kullanarak Beklenti-Maksimizasyon (EM) Kümeleme:**
K-Ortalamalarının en büyük dezavantajlarından biri, küme merkezi için ortalama değerin naif kullanımıdır. Aşağıdaki resme bakarak bunun bir şeyleri yapmanın en iyi yolu olmadığını anlayabiliriz. Sol tarafta, aynı ortalamaya merkezlenmiş farklı yarıçaplara sahip iki dairesel küme olduğu insan gözüne oldukça açık görünüyor. K-Ortalamalar bunun üstesinden gelemez çünkü kümelerin ortalama değerleri birbirine çok yakındır. K-Ortalamalar, yine ortalamanın küme merkezi olarak kullanılması sonucunda kümelerin dairesel olmadığı durumlarda başarısız olur.



Two failure cases for K-Means

Gauss Karışım Modelleri (GMM'ler) bize K-Ortalamalarından daha fazla esneklik sağlar. GMM'ler ile veri noktalarının Gauss olarak dağıtıldığını varsayıyoruz; bu, ortalamayı kullanarak döngüsel olduklarını söylemekten daha az kısıtlayıcı bir varsayımdır. Bu şekilde, kümelerin şeklini açıklamak için iki parametremiz var: ortalama ve standart sapma! İki boyutlu bir örnek alırsak, bu, kümelerin her türlü eliptik şekli alabileceği anlamına gelir (çünkü hem x hem de y yönlerinde standart bir sapmaya sahibiz). Böylece, her Gauss dağılımı tek bir kümeye atanır.

Her küme için Gauss'un parametrelerini bulmak için (örneğin ortalama ve standart sapma), Beklenti-Maksimizasyon (EM) adı verilen bir optimizasyon algoritması kullanacağız. Kümelere uydurulan Gaussian'ların bir örneği olarak aşağıdaki grafiğe bir göz atın. Daha sonra GMM'leri kullanarak Beklenti-Maksimizasyon kümeleme sürecine geçebiliriz.



EM Clustering using GMMs

1) Küme sayısını seçerek (K-Means'ın yaptığı gibi) ve her küme için Gauss dağılım parametrelerini rastgele başlatarak başlıyoruz. Verilere de hızlıca göz atarak ilk parametreler için iyi bir tahmin sağlamaya çalışılabilir. Yine de, yukarıdaki grafikte de görülebileceği gibi, bu% 100 gerekli değildir çünkü Gauss bize çok zayıf olarak başlarlar, ancak hızla optimize edilirler.
2) Her küme için bu Gauss dağılımları göz önüne alındığında, her veri noktasının belirli bir kümeye ait olma olasılığı hesaplanır. Bir nokta Gauss'un merkezine ne kadar yakınsa, o kümeye ait olma olasılığı o kadar artar. Gauss dağılımında verilerin çoğunun kümenin merkezine daha yakın olduğunu varsaydığımız için, bu sezgisel bir anlam ifade etmelidir.
3) Bu olasılıklara dayanarak, kümeler içindeki veri noktalarının olasılıklarını maksimize edecek şekilde Gauss dağılımları için yeni bir dizi parametre hesaplıyoruz. Bu yeni parametreleri, veri noktası konumlarının ağırlıklı toplamını kullanarak hesaplıyoruz; burada ağırlıklar, o belirli kümeye ait veri noktasının olasılıklarıdır. Bunu görsel olarak açıklamak için yukarıdaki grafiğe, özellikle örnek olarak sarı kümeye bakabiliriz. Dağıtım ilk yinelemede rastgele başlar, ancak sarı noktaların çoğunun bu dağılımın sağında olduğunu görebiliriz. Olasılıklara göre ağırlıklandırılmış bir toplamı hesapladığımızda, merkeze yakın bazı noktalar olmasına rağmen çoğu sağdadır. Dolayısıyla, doğal olarak dağılımın ortalaması bu noktalar kümesine daha da yakınlaşır. Ayrıca noktaların çoğunun “üstten-sağdan-aşağıya” olduğunu da görebiliriz. Bu nedenle standart sapma, olasılıkların ağırlıklandırdığı toplamı maksimize etmek için bu noktalara daha uygun bir elips oluşturmak üzere değişir.
4) Dağıtımların yinelemeden yinelemeye pek değişmediği yakınsamaya kadar 2. ve 3. adımlar yinelemeli olarak tekrarlanır.

GMM kullanmanın 2 önemli avantajı vardır. Birincisi, GMM'ler küme kovaryansı açısından KOrtalamalarına göre çok daha esnektir; standart sapma parametresi nedeniyle, kümeler

dairelerle sınırlı olmak yerine herhangi bir elips şeklini alabilir. K-Ortalamaları aslında her kümenin tüm boyutlardaki kovaryansının 0'a yaklaştığı özel bir GMM durumudur. İkinci olarak, GMM'ler olasılıkları kullandığından, veri noktası başına birden çok kümeye sahip olabilirler. Dolayısıyla, bir veri noktası üst üste binen iki kümenin ortasındaysa, sınıf 1'e yüzde X ve yüzde Y yüzde 2'ye ait olduğunu söyleyerek sınıfını tanımlayabiliriz. Yani GMM'ler karma üyeliği destekler.