İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ MADENCİLİĞİ VE DEMETLEME

YÜKSEK LİSANS TEZİ
Ahmet Cüneyd TANTUĞ

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Programı : BİLGİSAYAR MÜHENDİSLİĞİ

MAYIS 2002

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

VERİ MADENCİLİĞİ VE DEMETLEME

YÜKSEK LİSANS TEZİ

Ahmet Cüneyd TANTUĞ
504001550

Tezin Enstitüye Verildiği Tarih : 13 Mayıs 2002
Tezin Savunulduğu Tarih : 30 Mayıs 2002

Tez Danışmanı : **Prof. Dr. Eşref ADALI**
Diğer Jüri Üyeleri : **Doç. Dr. Coşkun SÖNMEZ (İ.T.Ü.)**
Doç. Dr. Ethem ALPAYDIN (B.Ü.)

MAYIS 2002

ÖNSÖZ

Bu tez çalışmam süresince sonuca gitmemde teknik anlamda yardımlarını esirgemeyen hocam Sayın Prof. Dr. Eşref ADALI'ya, Garanti Teknoloji Veri Madenciliği Bölüm Müdürü Sayın Dr. Ömer F. ALIŞ'a, idari ve maddi destekler konusunda çok katkıları bulunan Garanti Teknoloji Genel Müdür Yardımcısı Sayın Yük. Müh. Rüştü KARACA'ya, bana ömrümün ilk gününden beri konuda destek olan anneme, babama ve son olarak da benden desteğini esirgemeyen herkese sonsuz teşekkürlerimi sunarım.

Mayıs 2002 A. Cüneyd TANTUĞ

KISALTMALAR

LQV : Learning Vector Quantization

SOM : Self Organizing Map

MST : Minimum Spanning Tree

SSP : Shortest Spanning Path

FCM : Fuzzy C-Means

INSPEC : Information Service in Physics, Electrotechnology and Control

MBR : Memory Based Reasoning

SQL : Structured Query Language

TABLO LİSTESİ

ŞEKİL LİSTESİ

Sayfa No

ÖZET

Günümüzün hızla gelişen iş dünyası içerisinde her gün değeri artan, yöneticilerin ileriye dönük olarak doğru kararlar almasında ışık tutan karar destek sistemlerinden birisi olan veri ambarı ve veri madenciliği kavramlarının ayrıntılı olarak ele alındığı bu tez çalışmasında ayrıca uygulama anlamında ortaya çıkan sorunlar da mercek altına alınmış ve teknik anlamda bir takım iyileştirmeler önerilmiştir.

Günlük işleri yerine getiren canlı sistemlerde biriken bilgilerin arka planda belirli bir mantık içerisinde istif edildiği sistemlere veri ambarı adı verilmektedir. Canlı sistemlerden periyodik olarak bilgiler belirli bir şablonda toplanmakta ve veri ambarı içerisinde belirli bir anlam bütünlüğü içerisinde yığılmaktadırlar. Bu sebepten dolayı güncel olmayan veriler içeren veri ambarı içindeki bilgiler kullanılarak çeşitli araştırmalar ve incelemeler yapılır. İstatiksel raporlar oluşturulması, çok boyutlu analizler, veri madenciliği bu işlemlerden sadece birkaç tanesidir.

Veri ambarı üzerinde gerçekleştirilen veri madenciliği işleminde amaç verinin iç yapısındaki ilişkileri ortaya çıkarmak, kümelenmeleri ve bu kümelerin yapıları bulmak, varolan verilerden yola çıkarak çeşitli öngörülerde bulunmak, kısaca verinin iç yapısını çözmektir. Veri madenciliğinin çeşitli yöntemleri bulunmaktadır. Bunlar sepet analizi, sınıflandırma, demetleme, ilişkisel analiz, yapay sinir ağları, karar ağaçları, OLAP ve benzeri araçlardır. Bu yöntemlerin hiçbiri, her veri madenciliği işleminde tek başına kesin olarak yeterli değildir. Bu yöntemlerin çoğu kez birlikte kullanılması gereklidir.

Demetleme işlemi sadece veri madenciliğinde kullanılmamaktadır. Görüntü işleme, imge tanıma, sıkıştırma, pazarlama ve daha birçok değişik kolda da uygulama alanı bulmuştur. Demetleme en genel olarak büyük ölçekli heterojen yapıya sahip verileri homojen demetlere ayırma işlemidir. Sonuçta demet içi benzerlik yüksek, demetler arası benzerlik de düşük olmalıdır.

Demetleme yöntemini gerçeklemek için birçok algoritma bulunmaktadır. Bunlardan en yaygın kullanılanlarından bir tanesi hiyerarşik demetlemedir. Bu yapıda her veri bir demet olarak kabul edilerek işe başlanır ve her adımda birbirine en yakın demetler birleştirilerek, istenilen sayıda ya da tek bir büyük demet elde edilinceye

kadar devam edilir. Bir diğer sık kullanılan demetleme yöntemi olan bölmelemeli yöntemlerden K-Means yönteminde ise öncelikle K adet merkez rastgele olarak belirlenir daha sonra da her bir verinin bu k adet merkeze uzaklığı hesap edilerek en yakın olduğu merkeze atanır. Bu şekilde her bir veri bir merkeze atandıktan sonra ikinci adıma geçmek üzere bir merkeze yani bir demete dahil tüm verilerin ortalaması alınarak yeni demet merkezi hesap edilir. Algoritma bu şekilde bir kaç adım devam ettikten sonra sonuca gitmekte ve k adet demeti oluşturmaktadır.

Yukarıda kısaca tanıtılan demetleme yönteminin çok büyük ölçekli veriler üzerinde gerçeklenmesi durumunda zaman ve bellek karmaşıklığı çok artmaktadır. İşlemlerin performansını yükseltmek amacı ile bir takım iyileştirmelerin yapılması gerekmektedir. Performansı yükseltmek ve bellek karmaşıklığını azaltmak hedeflenerek yeni bir veri yapısı tasarlanmıştır. Ancak bu kriteri sağlamak için her incelemede değişen verileri saklamak için ayrı bir yapı kullanmak yerine esnek bir veri yapısı tasarlanmalıdır. Veri yapısı olarak bilinen yöntemlerle oluşturulan en esnek veri yapısında ise verimlilik çok düşük olmaktadır. Sadece 280 MB'lık bir veriyi demetlemek için 1,3 GB'lık bellek alanı işaretçiler ve diğer yardımcı alanlar ile boş yere kaplanmıştır. Algoritma tarafından kullanılmayan, sadece esnekliği korumak amaçlı işaretçilerin elendiği, verinin bellekteki yerleşiminin çok daha kompakt ve optimum seviyede olduğu bir çerçeve veri yapısı geliştirilmiştir. Bu şekilde bellek kullanım verimi arttırılmış ve bellek karmaşıklığı azaltılmıştır.

Ayrıca tasarlanan kompakt veri yapısının daha da kullanışlı olması amaçlanmış ve blok mekanizması ile destek sağlanmıştır. Bu mekanizma ile veriler kompakt bir şekilde bloklar halinde bulunmaktadır ve bellek içerisinde dağınık bir şekilde yerleşebilmektedirler. Böylelikle bellek kullanımı optimize edilemeye çalışılmış ve parçalı (fragmentated) bellek durumlarında dahi olası en uygun yerleşimin sağlanması amaçlanmıştır.

Veri ambarı içindeki verilerin tiplerinin çeşitliliğinden ötürü veri madenciliği algoritmalarının da bu tiplere destek vermesi gerekmektedir. Çoğu demetleme algoritması, veri madenciliğinin ortaya çıkmasından çok daha önceleri geliştirildiği için sadece sayısal veriler üzerinde çalışmakta ve çok büyük örüntü kümelerinde çok yavaş çalışmaktadır. Yapılan bu tez çalışmasında, günümüzün olası tüm veri tipleri için algoritmalar yeni genişletmeler getirilmiştir. Karakter katarı alanların

demetlenebilmesi amacı ile iki karakter katarı arası benzerliği belirleyen, bulanık mantık ilkesini kullanan bir karakter katarı karşılaştırma yöntemi geliştirilmiştir. Ek olarak veri ambarı kayıtlarında çok sık rastlanılan kategorik veri tipi tanımlaması yapılmış ve bu veri tipinin demetlemeye dahil olabilmesi için kategorik veri tipleri üzerinde bir benzerlik tanımlaması getirilmiştir.

Tezin dayandığı ilkelerin gerçeklendiği bir uygulama geliştirilmiş ve bu uygulama kapsamında çekirdek modüller gerçeklenmiştir. Ön çalışması yapılmış ve çekirdek kısımları gerçeklenmiş bu yazılım ile temel veri madenciliği işlemlerinin gerçeklenebileceği bir mimari çatı oluşturulmuştur.

DATA MINING AND CLUSTERING

SUMMARY

One of the most popular decision support system that helps managers to take future decisions more clearly is dataware house and data mining. These topics are widely analysed in this thesis, moreover with the application problems and some modifications has been supposed.

Dataware house acts as the main back office repository of the operational systems that has been used to perform daily operations. Data has been periodically collected from various discrete operational systems and pushed into the warehouse by preserving the consistency of the datawarehouse. A number of analysis and investigations has been done on the datawarehouse that contains data which is not update. Some of these operations are statistical reporting, multi-dimensional analysis and data mining.

The main aim of the data mining operations that takes place on the data warehouse is finding out the internal structure of the data, sensing the clusters and cluster properties, doing predictions based on the existing data; shortly finding underlying information of the data. There is various data mining techniques: market basket analysis, classification, clustering, link analysis, artificial neural networks, decision trees, OLAP and etc. However, none of these techniques is adequent for performing all data mining tasks. Collaboration of these techniques is needed to reach the solution.

Clustering is widely used not only in data mining but also in various subjects like image processing, pattern recognition, compressing, marketing and many other fields. The most general definiton of clustering is partitioning large heterogenous data into homogenous clusters. As a result, internal cluster similarity should be high although similarities between clusters are low.

One can find a lot of methods to cluster any data, but there is two major ones. The first one is called hierarchical clustering. In the begining of this technique, all individual records that will be clustered become individual cluster centers. After that step, the closest two records are joined and forms one single cluster.This process continues until the desired number of clusters emerge. Another common used

algorithm in clustering is K-Means algorithm that needs to know the number of desired clusters K before the analysis. This algorithm starts clustering process by choosing K cluster centers from the data randomly and then calculates the distances between all points and all cluster centers. Each point is assigned to the nearest cluster. After all points assigned to a proper cluster, each cluster center is re-calculated by using the averages of the cluster members. After a number of step, K-Means converges to a stable state and forms the desired clusters.

The space complexity and time complexity of clustering algorithms decribed above become very high when the data that will be clustered is too large. In that circumstances, the performance of the clustering algorithms should be improved. To minimize the space complexity of the clustering algoritms, a new approach is introduced to store the data that will be clustered in main memory. A new and flexible data structure is developed to maintain the compactness of data. The common data structures that satisfy the flexibility condition, requires 1.3 GB memory space just for the pointers and sub-fields necessary for data structures although the size of the data to be clustered is nearly 280 MB. The memory space used for pointers and sub-fields are not necessary for the clustering process and that makes the efficiency of memory usage 17%. A new data structure for storing data is produced by eliminating all unnecessary fields and pointers and stores data in a maximum compact way. In this way, the efficiency of memory usage becomes nearly 100% and the time complexity of algorithms decreases significantly. This is done by building a complete template record.

The block mechanism and block management routines are added to this compact data storing structure and makes our structure more useful. In this mechanism, the data are grouped in blocks of records and these blocks can be placed anywhere in the main memory. The main target was optimizing the memory usage and we got better results eventhough the main memory is hardly fragmentated.

The support of data mining for various types of data is needed for operating on data warehouses that there exists records that has various types of attributes. The most of the clustering algorithms is founded early before data mining emerges so most of them operates only on numeric values. Some extensions that supports every type of data that can be found today are added to the algorithms. A fuzzy string similarity

method that calculates the similarity between two strings is introduced for clustering string attributes. In addition, the categorical data type definition is done and a similarity measure between categoric attributes is composed to involve categorical data to the clustering.

As a result, a modular application that includes the main points mentioned above is developed and the core modules are coded. An architecture for performing fundamental data mining techniques is built with this software.

1. VERİ AMBARI KAVRAMI

1.1. Veri Ambarı Tanımı

Veri ambarı, bir işletmenin ya da kurumun çeşitli birimleri tarafından canlı sistemler aracılığı ile toplanan bilgilerin, ileride değerlendirmeye alınabilecek olanlarının arka planda yer alan bir sistemde birleştirilmesinden oluşan büyük ölçekli bir veri deposudur.

Günümüzün ticari işletmelerinde bilgi sistemlerinin iki ayrı başlık altında toplandığı söylenebilir:

- Canlı Sistemler

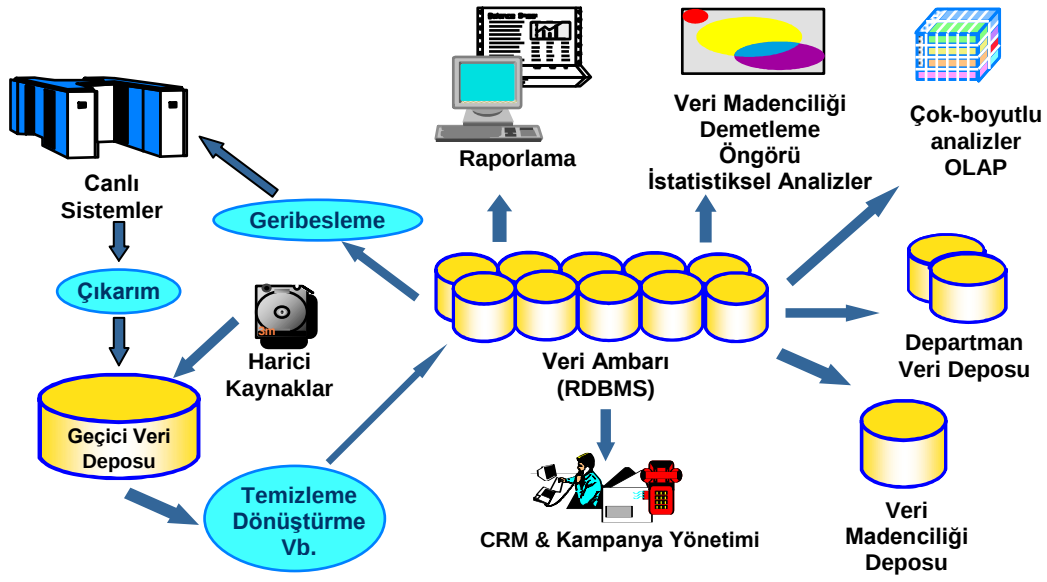
  Bu sistemlerde güncel veriler bulunur. Günlük yapılan işleri gerçeklemek ve sonuçları saklamak için tasarlanan bu sistemlere örnek olarak marketlerde ya da mağazalarda stok takibi, üyelerin borçları, satış işlemleri ve ödeme kayıtları gibi bilgilerin işlendiği ve tutulduğu bilgi sistemleri verilebilir. Bu tür canlı sistemlerde erişilebilirlik esastır yani veriye en kısa sürede ulaşmak, işlemleri en kısa sürede sona erdirmek hedeflenir. Örneğin süpermarkette bir satış işleminin mümkün olduğu kadar çabuk bitirilmesi istenir. Bu nedenden dolayı canlı sistemler çevrimiçi çalışma prensibi ile tasarlanırlar.

- Karar Destek Sistemleri

  İşletmelerde yer alan ikinci tür bilgi sistemleri ise karar destek sistemleridir. Bu sistemlerde yer alan bilgiler çeşitli incelemelerden ve araştırmalardan geçirilerek yöneticilerin ileride işletmenin kârını ya da verimliliğini arttırması, gelecekte izlenecek şirket politikalarının belirlenmesi ve benzeri yönetimsel kararların alınmasını kolaylaştırır ve bu kararların daha doğru verilmesine yardımcı olurlar. Bu sistemlerde verilerin erişimi birinci kriter değildir. Herhangi bir veriye herhangi bir anda çabuk erişmek gerekmez. Karar destek sistemlerinde öncelik performanstadır. Bu sistemlerde veriler, canlı sistemlere oranla çok daha büyük boyutlardadır. Dolayısı ile bu verilerin incelenmesi ve bu incelemelerden sonuçlar çıkartılması, sistem kaynaklarını aşırı kullanmakta ve uzun süre almaktadır. Bu yüzden karar destek

sistemlerinde, yapılacak incelemelerin ve araştırmaların performansını arttırmak için bir takım önlemler alınmış ve iyileştirmeler yapılmıştır.

Veri ambarı, bir karar destek sistemi olarak nitelendirilebilir. Veri ambarı esasında günlük işlemlerin gerçeklendiği canlı sistemlerin arka planında bulunmaktadır. Canlı sistemlerde oluşan veriler periyodik olarak veri ambarına aktarılırlar. Bu periyodun seçimi tamamen veri ambarını kullanan işletmenin ihtiyaçları doğrultusunda belirlenir ve 1 gün, 1 hafta 1 ay gibi çok değişken olabilir. Dolayısı ile veri ambarı çevrimdışı olarak çalışmaktadır. Yani veri ambarı içerisindeki kayıtlar, güncel olmayabilir, çoğunlukla da güncel değildir. Veri ambarına belirli aralıklarla verileri gönderen canlı sistemlerin bir tane olma zorunluluğu da yoktur. Örneğin bir işletmenin içerisindeki farklı bölümler ve farklı günlük işlemleri gerçeklemek üzere tasarlanmış birbirlerinden habersiz ve bağımsız çalışan değişik canlı sistemlerdeki veriler, veri ambarındaki bir tek yapı içerisinden erişilebilecek bir şekilde toplanıp veri ambarına aktarılabilir. Bu aktarım süreci içerisinde veriler üzerinde veri ambarında önceden bulunan kayıtları aktarmama, kayıtlar içerisindeki bazı bilgileri değiştirmek, silmek ve benzeri bir takım işlemler de gerçeklenebilir. Bu ön işleme sonrasında değişik kaynaklardan yani değişik canlı sistemlerden toplanan veriler, anlamsal bütünlüğü sağlayacak şekilde veri ambarına yerleştirilirler. Tipik bir veri ambarı mimarisi şu şekildedir:



Şekil 1.1 – Veri Ambarı Mimarisi

Şekilden de görülebileceği gibi veri ambarına aktarılan veriler sadece işletmeye ait canlı sistemlerden gelmemektedir. Dışarıda işletmeye ait olmayan kaynaklardan da veri alımı mümkün olmaktadır. Ayrıca gene şekilde “geri besleme” olarak gösterilen

adım ile de veri ambarı ve canlı sistemler arasında bir ilişki kurulmuştur. Bu bağlantı ile canlı sistemler veri ambarından gelen istekleri karşılayabilmektedirler.

Bu şekilde veri ambarına aktarılan veriler üzerinde daha sonra gerçeklenen incelemeler ve araştırmalar neticesinde yöneticiler ve iş uzmanlarının daha rahat kararlar alabilecekleri ve yapılacak işlere ışık tutan bir takım sonuçlar üretilir. Bu sonuçlar istatistiksel, olasılıksal raporlar şeklinde olabileceği gibi, bu büyük veri yığını içerisinde yer alan en tecrübeli yöneticilerin bile göremeyeceği veri ilişkilerinin belirlenmesi şeklinde de olabilir.

Veri ambarında yer alan bilgiler, bu bilgilerin kullanılacağı alanlara göre ayrı alt depolara dağıtılabilirler. Çoğunlukla işletme içindeki departmanların kullanımına göre bölmelenen veri ambarlarında alt depolar (data-mart) oluşturulur.

1.2. Veri Ambarı - Veri Tabanı Arasındaki Farklar

Veri tabanı içerisindeki bilgiler genelde anlık bilgilerdir. Yani o an için güncelliğini koruyan ancak belirli bir süre sonunda güncelliğini kaybedecek olan bilgilerdir. Ancak veri ambarı içerisindeki veriler genelde yığılarak birikirler ve verilerin geçerliliği çok daha uzun süre olmaktadır. Veri ambarı içerisinde ne kadar çok kayıt olursa yapılan incelemelerin sonucu da o kadar doğru olacaktır. Oysa veri tabanı içerisindeki kayıt adedinin çok fazla sayıda olması durumda, bu veri tabanını kullanan canlı sistemlerin performansları düşecek dolayısı ile verilere erişim çok yavaşlayacaktır ki bu, canlı sistemlerde en istenmeyen durumdur.

1.3. Veri Ambarları Üzerinde Gerçeklenen İşlemler

Şekil 1.1'de yer aldığı gibi veri ambarı üzerinde bir çok işlem gerçeklenebilmektedir. Bunlardan başlıcaları veri madenciliği, çok boyutlu analizler (OLAP), müşteri ilişkileri yönetimi (MİY – CRM), kampanya yönetimi, istatistiksel analizler ve raporlamadır. OLAP ile veri ambarı içerisinde yer alan kayıtlar yöneticiler tarafından istenilen boyutlarda ve biçimlerde raporlar haline getirilebilir ve çok boyutlu analizler yapılabilmektedir. Veri madenciliği ile, ilerideki bölümlerde daha da ayrıntılı olarak anlatılacağı gibi, verinin doğasında yatan kümelenmeleri, kayıtlar arasındaki ilişkileri bulmak, karar verme sürecinde yer alan soruların cevaplarını çıkarmak olası hale getirilmiştir. Aynı zamanda veri ambarı içerisinde yer alan kayıtlar üzerinde istatistiksel yaklaşımlarla raporlar oluşturmak ve istatistiksel sonuçlara varmak da mümkündür.

2. VERİ MADENCİLİĞİ

2.1. Tanımı

2.2. Amaçları

Pratik anlamdaki veri madenciliği genellikle sınırlı sayıda tekniğin, gene sınırlı sayıda koşullarda uygulanması ile gerçeklenir. Güncel, ekonomik ve iş hayatındaki bir çok sorun aşağıdaki altı genelleme ile tanımlanabilir:

Sınıflandırma (Classification)
Kestirme (Estimation)
Öngörü (Prediction)
Benzerlik Kümelemesi (Affinity Grouping)
Demetleme (Clustering)
Tanımlama (Description)
Tüm bu analizler için eşit oranda verimlilik sağlayacak kullanışlı bir veri madenciliği aracı ya da tekniği bulunmamaktadır. İstenilen amaca ve analizin türüne göre değişik teknikler kullanılmaktadır.

2.2.1. Sınıflandırma (Classification)

En çok gerçeklenen veri madenciliği işlerinden birisi olan *sınıflandırma* işlemi, insan düşünce yapısına en yatkın veri madenciliği işlemlerinden bir tanesidir. İnsanoğlu dünya üzerideki maddeleri daha iyi anlamak ve başkalarına anlatmak için hemen hemen herşeyi sürekli sınıflandırmakta, kategorilere ayırmakta ve derecelendirmektedir. Maddeleri elementlere, köpekleri türlere, ülkeleri şehirlere, şehirleri semtlere vb. şekilde kategorize etmektedir.

Veri madenciliğinde geçerli olan sınıflandırma işleminde ise amaç, yeni karşılaşılan bir girdinin özelliklerinin incelenip, bu girdinin daha önce tanımlanmış olan sınıflardan hangisine atanacağına karar vermektir. Genellikle bizim ilgilendiğimiz kapsamdaki girdiler veri tabanı kayıtları olmakta ve "bir sınıfa atama" işlemi de veri tabanında bu kayda ilişkin olan alanlardan sınıf kodu ya da benzer bir alanın uygun şekilde doldurulması şeklinde gerçeklenmektedir.

Sınıflandırma işlemine örnek olarak verilebilecek bazı işlemler aşağıda sıralanmıştır:

- Kredi başvurularını düşük, orta ve yüksek riskli olarak sınıflandırma
- Sigorta şirketlerini dolandırmak için işlenen sahte suçları seçme

- Yeni gelen bir öğrencinin hangi sınıfta eğitim görmesi gerektiğine karar verme

Tüm bu örneklerde sınırlı sayıda sınıf bulunur ve beklenen de her bir kaydın bu sınıflardan birisine atanmış olmasıdır. Sınıflandırma için Karar Ağaçları (Bölüm 2.3.5), Bellek Tabanlı Sistemler (Bölüm 2.3.2) yöntemleri en uygun olan tekniklerdir. Ayrıca bazı durumlarda İlişkisel Analiz (Bölüm 2.3.4) yöntemi de faydalı sonuçlar üretmektedir.

2.2.2. Kestirme (Estimation)

Bir önceki konuda anlatılan sınıflandırma işlemi ayrık sonuçlar üretmektedir, yani bir kayıt ya belirli bir sınıfa aittir ya da değildir. Ancak kestirme işleminde ise sürekli değerler alabilen sonuçlar üretilmektedir. Kestirme işleminde, giriş verisine göre bilinmeyen sürekli bir değişken (örneğin gelir düzeyi, boy ya da kredi kartı bakiyesi gibi) için bir değere ulaşılır.

Uygulamada değerlendirme çoğunlukla sınıflandırma işlemlerini gerçeklemek için kullanılır. Örneğin bir kredi kartı şirketi, hesap ekstrelerinde yer alan reklam alanını bir kayak takımı üreticisine kiralamak istediği zaman, kredi kartı şirketinin ekstre göndereceği tüm müşterilerini, kayak yapanlar ve yapmayanlar şeklinde ayıracak sınıflandırma modelini kurması gerekmektedir. Bir başka yaklaşım ise ekstre gönderilecek her müşterinin kayak yapma ihtimalinin hesaplanmasıdır. Bu ihtimal 0 ile 1 arasında herhangi bir değer alabilir. Şu aşamada sınıflandırma sorunu, uygun bir eşik değer seçimine indirgenmiştir. Eşik değere eşit ya da daha fazla "kayak yapma ihtimali"ne sahip müşteriler "kayak yapanlar", bu eşik değerin altında bir değerde "kayak yapma ihtimali"ne sahip müşteriler ise "kayak yapmayanlar" olarak sınıflandırılmış olurlar.

Değerlendire işleminin sınıflandırma işleminden bir diğer önemli avantajı da her bir kaydın sahip olduğu değerlendirmeye göre tüm kayıtların sıralanabilir hale gelmesidir. Az önceki örnek üzerinden devam etmek gerekirse, kayak takımı üreticisinin 500 bin ekstreye reklam verebilecek sınırlı bir bütçesi olması durumunda eğer 1,5 milyon kredi kartı müşterisi "kayak yapanlar" olarak sınıflandırılmış ise seçilecek en basit yöntem bu 1,5 milyon kişi içerisinden rastgele müşterileri seçmek ve reklam içeren ekstreleri seçilen bu müşterilere göndermek olacaktır. Ancak eğer kredi kartı şirketinin her müşteriye ait bir kayak yapma ihtimali değerlendirmesi varsa, reklam içeren ekstreler, kayak yapma ihtimali en yüksek olan 500 bin müşteriye gönderilerek en uygun çözüm sağlanmış olur.

Değerlendirme işleminin başka örnekleri aşağıda bulunmaktadır:

- Bir ailedeki çocuk sayısının tahmini
- Ailenin toplam gelirinin hesaplanması
- Müşterilerin bağlılık süresinin hesaplanması
- Müşterilerin yeni banka hesap tiplerini seçme olasılığının bulunması

Değerlendirme işlemi için kullanılabilecek tek yöntem Yapay Sinir Ağları (Bölüm 2.3.6) yöntemidir.

2.2.3. Öngörü (Prediction)

Öngörü işlemi sınıflandırma veya kestirme işlemlerine çok benzer bir işlemdir. Ancak öngörü işleminde sınıflandırma, gelecek için tahmin edilen belirli bir davranışa ya da belirli bir değere göre yapılır. Öngörü işleminde yapılan sınıflandırmanın doğru olup olmadığını test etmenin tek yöntemi, bekle ve gör prensibidir.

Sınıflandırma ve değerlendirme işlemlerinde kullanılan hemen hemen her yöntem küçük adaptasyonlarla öngörü işlemi için de kullanılabilir hale gelirler. Bunun için, eski verilerden, gelecek için tahminde bulunulacak verinin daha önceden bilindiği bir eğitim kümesi hazırlamak gerekmektedir. Bu eski veriler, şu anda gözlenen davranışları açıklayacak bir modelin kurulması için kullanılacaktır. Bu modelin herhangi bir veri kümesine uygulanması sonucu ortaya çıkan sonuç ise gelecekteki davranışlar için bir öngörü niteliğinde olacaktır.

Örneğin bir mağazadan yapılan alışverişlerde hangi malların daha çok bir arada alındığının ortaya çıkarılması amaçlayan Sepet Analizi Tekniği, küçük değişiklerle gelecekte hangi malların bir arada satılacağını kestirmeyi ve bu amaca uygun olarak satışı daha da arttıracak hazırlıkların belirlenmesini sağlayacak şekilde uyarlanabilir.

Veri madenciliği kapsamında yer alan öngörü işlemleri için bir takım örnekler aşağıda listelenmiştir:

- Hangi müşterilerin 6 aylık bir zaman dilimi sonrasında ayrılacağı
- Hangi cep telefonu abonelerinin özel servislerden (WAP, GPRS gibi) faydalanmak isteyeceği
- Deprem tahmini

Sepet analizi tekniği (Bölüm 2.3.1), bellek tabanlı teknikler (Bölüm 2.3.2), karar ağaçları (Bölüm 2.3.5) ve yapay sinir ağları (Bölüm 2.3.6) yöntemlerinin tümü öngörü işlemi için uygundur. Öngörü işlemi için kullanılacak olan yöntemin seçimi

tamamen giriş verisinin özelliklerine, öngörülecek değerin tipine ve yapılan öngörünün açıklanabilirlik özelliğinin ne derece önemli olduğuna doğrudan bağlıdır.

2.2.4. Benzerlik Kümelemesi (Affinity Grouping)

Benzerlik gruplaması ya da sepet analizi, hangi ürünlerin hangi ürünlerle daha çok satıldığının belirlenmesi amacı ile kullanılır. En yaygın örneği süpermarketlerdeki alışveriş kayıtlarının incelenerek hangi satışlarda hangi ürünlerin daha çok beraber satıldıklarının saptanmasıdır. Perakende satış yapan mağazalar, benzerlik gruplaması işlemini daha çok ürünlerin raflardaki ya da kataloglardaki yerleştirilmesinin, çok satılan ürünlerin birlikte gözükecek şekilde uygun olarak düzenlenmesi amacı ile kullanmaktadırlar. Aynı zamanda benzerlik gruplaması ile çapraz satış fırsatlarının belirlenmesi ya da hizmetlerde ve ürünlerde satışı arttırıcı paketlerin, gruplandırmaların ve promosyonların yapılması sağlanabilmektedir.

Aynı zamanda benzerlik eşlemesi, verilerden kurallar çıkarmanın en basit yöntemidir. Örneğin kedi maması ile kedi kumu yeteri kadar sayıda birlikte satılıyorlar ise aşağıdaki 2 ilişkisel kuralı türetebiliriz:

- Kedi maması alan bir müşterinin, kedi kumu da alma olasılığı P_1'dir.

- Kedi kumu alan bir müşterinin, kedi maması da alma olasılığı P_2'dir.

İlişkisel kurallar bölüm 2.3.1. de daha ayrıntılı olarak ele alınacaktır.

2.2.5. Gruplandırma

Gruplandırma işlemi, heterojen yapıya sahip bir popülasyonu, daha homojen birkaç altgruba ya da kümeye bölütleme işlemidir. Sınıflandırma ile gruplandırmayı birbirinden ayıran en önemli fark, gruplandırma işleminin, sınıflandırma işleminde olduğu gibi önceden belirlenmiş bir takım sınıflara göre bölütleme yapmamasıdır. Hatırlanacağı gibi sınıflandırmada yeni gelen her bir kayıt, önceden sınıflandırılmış bir takım sınıflar üzerinde yapılan bir eğitim neticesinde ortaya çıkan bir modele göre önceden belirlenmiş olan bir sınıfa atanmaktadır. Gruplandırma ya da demetleme işleminde ise önceden tanımlanmış sınıflar ya da örnek sınıflar bulunmamaktadır. Kayıtların gruplandırılması işlemi, kayıtların birbirlerine olan benzerliklerine göre yapılmaktadır. Oluşan sınıfların hangi anlamları taşıdığının belirlenmesi tamamen analizi yapana kalmıştır. Örneğin hastaların kayıtlarından oluşan verilerin gruplanması sonucunda semptomlardan oluşan kümeler, değişik hastalıklara işaret edebilir.

Demetleme işlemi çoğunlukla bir başka veri madenciliği işlemi için bir ilk işlem olarak kullanılır. Örneğin gruplama işlemi bir pazar payı araştırması için bir ilk işlem olarak uygulanabilir. "Ne tip promosyonlar müşteriler tarafından rağbet görür?" sorusunun cevabını bulmayı kolaylaştırmak için herkes için tek bir model uygulamaktan vazgeçip, müşteriler alışveriş alışkanlıklarına göre gruplandırılırsa, her küme için "Bu kümedeki müşteriler hangi tip promosyonlara rağbet eder?" sorusunun cevabı çok daha kolay verilir.

Gruplama ya da demetleme işlemi bu tezin ana konusu olduğu için ilerleyen bölümlerde çok daha ayrıntılı olarak ele alınacaktır.

2.2.6. Tanımlama (Description)

Veri madenciliği uygulamalarının amaçlarından birisi de, karmaşık bir veri tabanında neler olup bittiğini, verileri ilk elden oluşturan insanlar, ürünler ve işlemler bazında daha iyi anlamamızı sağlayacak açıklamalar bulmaktır. Bir davranış için verilen iyi bir tanımlama çoğu kez bu davranış için bir de açıklama getirmelidir. En azından iyi bir tanımlama, açıklamayı bulmak için nereden başlanması gerektiğini göstermelidir.

Veri madenciliğinde kullanılan tekniklerin bazıları, örneğin sepet analizi tekniği, tamamen tanımlayıcıdır. Ancak yapay sinir ağları gibi bazıları da tanımlamanın bir adım sonrası için hiç bir bilgi vermemektedir.

2.3. Veri Madenciliği Teknikleri

2.3.1. Sepet Analizi

Sepet analizi tekniği, adında da anlaşılacağı üzere, çok basit olarak hangi ürünlerin hangi ürünlerle satıldığını, hangi ürünlerin promosyona girmesi gerektiğini ve benzeri bilgileri ortaya çıkarır. Her ne kadar yoğunlukla satış alanında kullanılsa da, sepet analizi tekniği bunun dışında bir çok alanda daha kullanılmaktadır:

- Kredi kartları ile yapılan alışverişlerin işlenmesi ve müşterilerin yapacakları potansiyel harcama kalemlerinin bulunması
- Cep telefonundaki opsiyonel hizmetlerin (GPRS, WAP, TeleSekreter vb.) müşteriler tarafından tercih edilmelerine göre kârı arttırmak için hangi ürünlerin birlikte kampanyaya girmesi gerektiğini belirlemek
- Sigortacılıkta ortaya çıkan değişik tarzdaki suçların dolandırıcılık olup olmadığının belirlenmesi ve yapılacak soruşturmaya ışık tutulması

- Hastaların sağlık kayıtlarından, önerilen çeşitli tedavi kombinasyonlarından doğan komplikasyonların görülmesi

Sepet analizi çoğunlukla ticari anlam taşıyan verilerin var olduğu ancak bu veri üzerinde hangi örüntülerin (pattern) aranılacağının bilinmediği durumlarda bir başlangıç noktası olarak kullanılır. Bu veri içerisindeki bazı kalıplar sayesinde kazancı arttırmak üzere bir takım aksiyonlara gidilebilir.

Örneğin yurtdışında yapılan sepet analizi tekniklerine göre Perşembe günleri bira ve çocuk bezi satışlarının çok fazla sayıda olduğu görülmüştür (Berry ve Linoff 1997). Bunun temel nedeni olarak ise evli çiftlerin hafta sonunu evde geçirmek istemeleri ve bu süre içerisinde de rahatlarını bozmamak için gerekli olması muhtemel bira ve çocuk bezini hafta sonu gelmeden almak istemeleri olarak gösterilmiştir.

Ancak sepet analizi, geleceğe dönük tahminlerde pek başarılı değildir. Sepet analizinin temeldeki mantığında yer alan teknikler istatistik ve olasılıktan gelmektedir.

Ticari anlam taşıyan veriler üzerinde belirli bir ürün kombinasyonunun kaç defa geçtiğinin bulunması işlemi tek başına yeterli değildir. Bu kombinasyonu işletme açısından anlamlı hale getirecek olan kilit nokta, bu kombinasyonu oluşturan kuralı bulmaktır.

Kural tanımı iki kısımdan oluşur: koşul kısmı ve sonuç kısmı (Berry ve Linoff 1997).

Eğer *KOŞUL* doğru ise, *SONUÇ* da doğrudur.

Örneğin “konferans görüşme hizmeti satın alan bir müşteri, çağrı bekletme hizmeti de satın almıştır” kuralı kısaca şu şekilde gösterilir:

Eğer konferans görüşme ise çağrı bekletme

Pratikte işletme açısından eyleme dönüştürülebilecek kuralların sadece bir adet sonuç kısmı vardır. Yani

Eğer Çocuk bezi ve Perşembe ise Bira satılabilir kuralı

Eğer Perşembe ise Çocuk Bezi ve Bira kuralından daha çok faydalıdır. Çünkü sadece günün Perşembe olmasından dolayı birisine çocuk bezi ve bira satmaya çalışmak anlamsız olacaktır. Aksine eğer günlerden Perşembe ise ve müşteri çocuk bezi almış ise bu müşterinin bira alma olasılığı çok yüksek demektir. Bunun için işletme bira satışlarını arttırmak için Perşembe günleri çocuk bezi ürünleri ile biraları beraber satmak üzere promosyona girebilir. Dolayısı ile ikinci kural işletme açısından çok daha anlamlı ve eyleme dönüştürülebilecek bir yapıya sahiptir.

Sepet analizi tekniğinin çalışma mantığı aşağıda anlatılmıştır:

Örnek olarak bir marketteki satış noktasında kaydedilen aşağıdaki alışveriş bilgilerini ele alalım.

Tablo 2.1 - Market Satış Noktasındaki Örnek Satış Kayıtları

Müşteri No	Satın Alınan Ürünler
1	kola, soda
2	süt, kola, camsil
3	kola, deterjan
4	kola, deterjan, soda
5	camsil, soda

Bu tablodaki bilgiler kullanılarak bir birliktelik (co-occurence) tablosu oluşturulur:

Tablo 2.2 - Satış Kayıtlarından Oluşturulmuş Birliktelik Tablosu

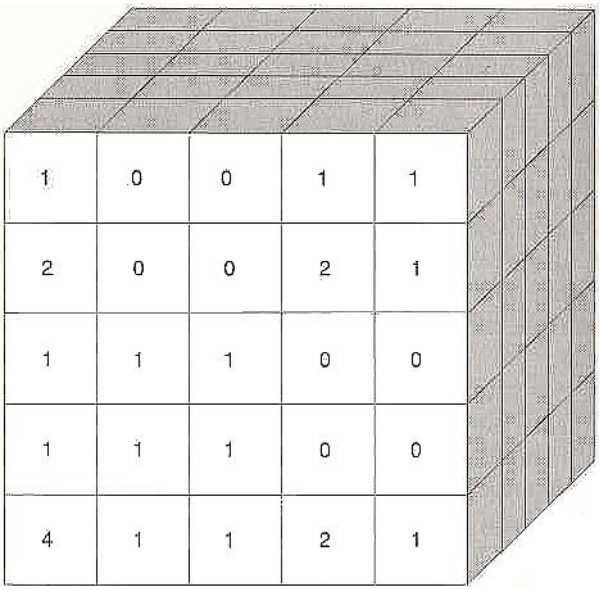
	Kola	Camsil	Süt	Soda	Deterjan
Kola	4	1	1	2	1
Camsil	1	2	1	1	0
Süt	1	1	1	0	0
Soda	2	1	0	3	1
Deterjan	1	0	0	1	2

Bu birliktelik tablosu ilk bakışta şunları göstermektedir:

- Kola ve sodanın birlikte satılma olasılığı diğer tüm ikili ürün gruplarından daha fazladır.
- Deterjan hiçbir zaman süt ya da camsil ile beraber satılmamıştır.
- Soda veya deterjan ile süt hiçbir zaman satılmamıştır.

Örnek veri kümesindeki 5 kayıttan 2'sinde soda ile kola birlikte satılmıştır. Bu iki kayıt, ilgili kuralı destekleyen kayıtlardır. Yani bu kuralın doğru olma ihtimali %40'dır. Ayrıca soda içeren tüm kayıtlarda kolanın da alınmış olması çok yüksek derecede güvenilirlik sağlamaktadır. Bu yüzden "eğer soda ise kola" kuralının güvenilirliği %100'dür. Bu kuralın tersi için ise "eğer kola ise soda" güvenilirlik değerimiz o kadar yüksek değildir. Tablodan da anlaşılacağı üzere bu kuralın güvenilirliği %50'dir.

Bu örnek çalışma sadece hangi iki ürünün birlikte daha çok satıldığını bulma amaçlıydı. Oysa ikiden fazla ürünün birliktelik analizini yapmak kolay değildir. Örneğin eğer 3 ürünün birliktelik analizi yapılmak istenseydi, birliktelik tablosu 2 boyutlu olmaktan çıkıp küp halini alacaktır:



Şekil 2.1 - 3 Boyutlu Birliktelik Tablosu

Toplam 5 ürünün bulunduğu bu basit analizde dahi doldurulması gereken toplam 125 adet kutu vardır. Bu sayı, küp yapısının simetriklik özelliği kullanılarak bir miktar azaltılabilse dahi gene de n adet ürün için n^3 mertebelerinde bir analiz gerektirecektir.

n adet ürünün bulunduğu bir veri ambarında ürünlerin n'li kombinasyonlarını bulmak, n^n boyutunda analiz yapmayı gerektirmektedir.

Örneğin bir süpermarkette en az 10.000, çoğunlukla ise 20.000 ila 30.000 kalem ürün bulunur. Bu ürünlerin 2'li kombinasyonlarının toplam olasılığı 50 milyon, 3'lü kombinasyonlarının olasılığı ise 100 milyara yakındır. Ayrıca süpermarketlerin satış kayıtları da çok büyük ölçektedir. Orta ölçekli (10-20 şubeli) bir süpermarketin yılda on milyonlarla ölçülen satış kayıtları bulunmaktadır. Günümüzün gelişmiş bilgisayarlarında dahi bu büyüklükteki verileri işleyip 3'lü ürün kombinasyonları çıkarmak çok pahalıya mal olmaktadır. Bu verilerden 5'li ya da daha fazla kombinasyonların çıkarılması ise çok büyük şirketlerin dahi altından kalkamayacağı bir maliyet getirmektedir.

Sepet analizinin başarılı olduğu noktalar:

- Kolay ve anlaşılır sonuçlar üretir.
- Gözetimsiz veri madenciliği yöntemidir.
- Değişik boyutlardaki veriler üzerinde çalışabilir.

- Her ne kadar kayıtların sayısı ve kombinasyon seçimine göre işlem adedi artsa da sepet analizi için her adımda gerekli olan hesaplamalar diğer yöntemlere göre (genetik algoritmalar, yapay sinir ağları vb.) çok daha basittir.

Sepet analizinin başarısız olduğu noktalar:

- Sorunun boyutu büyüdükçe, gerekli hesaplamalar üstel olarak artmaktadır.
- Sepet analizinde kullanılacak doğru ürünlerin seçimi. Ürün gruplandırma (süt ürünleri, unlu mamüller vb.) biraz bilgi kaybı getirse de analizin boyutlarını küçültebilir.
- Kayıtlarda çok az rastlanan ürünleri yok sayar. Sepet analizi tekniği, en doğru sonucu, tüm ürünlerin kayıtlar içinde yaklaşık aynı frekansta göründüğü durumlarda üretmektedir.

2.3.2. Bellek Tabanlı Yöntemler

İnsanlar kararlarını genellikle daha önce yaşadıkları deneyimlere göre verirler. Örneğin doktorlar bir hastayı incelerken, elde ettiği bulguları daha önce tedavi ettiği benzer hastalığa yakalanmış hastalar üzerindeki deneyimlerini kullanırlar. Aynı şekilde bir sigorta eksperi de bir olayın sahtekarlık olup olmadığı bulmak için daha önceki sahterkarlıklarla olan benzerlikleri göz önünde bulundurmaktadır. Gözetimli bir veri madenciliği tekniği olan bellek tabanlı yöntemler de benzer şekilde deneyimleri kullanmaktadır. Bu yöntemlerde, bilinen kayıtların bulunduğu bir ver itabanı oluşturulur ve sistem yeni gelen bir kayda komşu olan diğer kayıtları belirler ve bu kayıtları kullanarak tahminde bulunur ya da bir sınıflandırma işlemi gerçekler. Bellek yaklaşımlı yöntemlerin en önemli özelliği veriyi olduğu gibi kullanabilme yeteneğidir. Diğer veri madenciliği tekniklerinin aksine bellek tabanlı yöntemler, kayıtların formatı yerine sadece iki işlemin varlığı ile ilgilenir: iki kayıt arasındaki uzaklığı belirleyen bir uzaklık fonksiyonu ve komşu kayıtları işleyerek bir sonuç üreten bir kombinasyon foksiyonu.

Bellek tabanlı yöntemler bir çok alanda kullanılmaktadırlar:

- Sahtekarlık Tesbitinde

  Karşılaşılan her yeni sahtekarlık davası, önceki sahtekarlık davaları ile mutlaka benzerlik gösterecektir. Bellek tabanlı yöntemler bu benzerlikleri bulur ve ilerideki işlemler için bu bulguları işaretler.

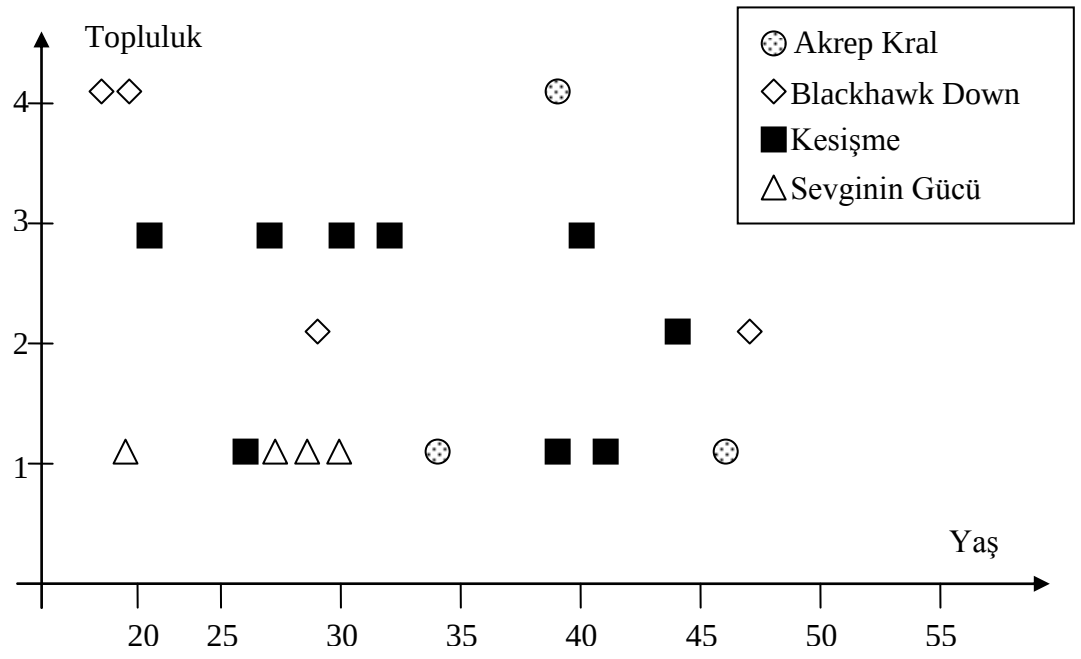
- Müşteri Tepkisi Tahmini

Belirli bir pazarlama faaliyeti ya da benzer bir faaliyete cevap verecek yeni bir müşteri, çok yüksek bir olasılıkla bu faaliyete daha önceden cevap veren müşterilere benzer davranışlar gösterecektir. İşte bellek tabanlı yöntemler de bu davranışları tespit etmekte kullanılırlar.

- Klinik İşlemlerde

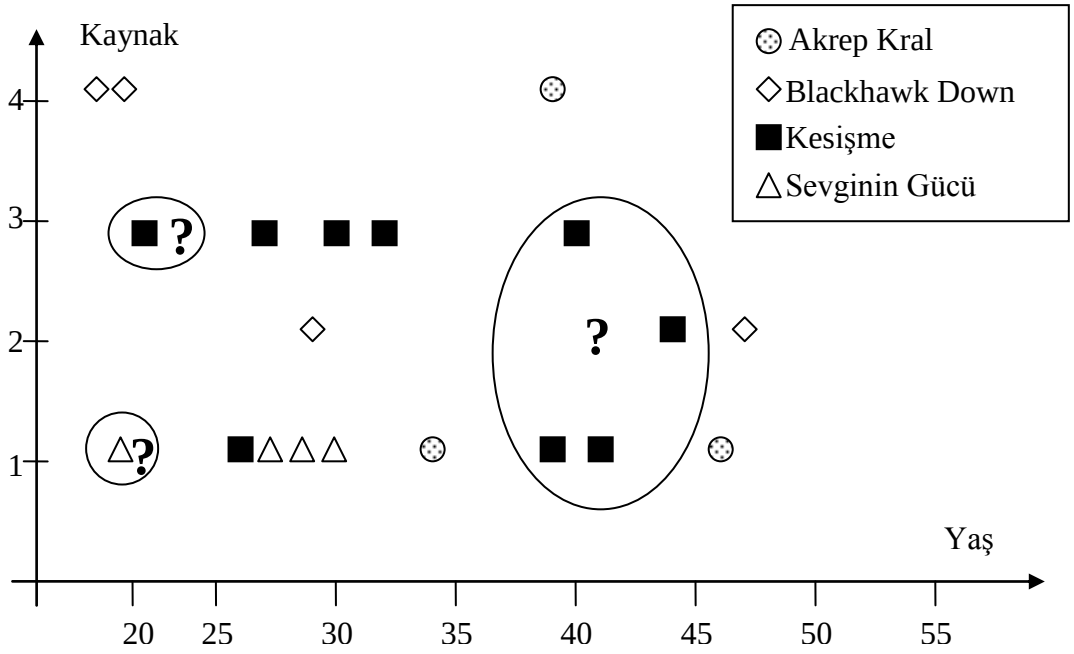
  Hastalara verilebilecek en etkili tedavi yöntemi, benzer rahatsızlıkları daha önceden yaşamış hastalar üzerinde başarılı sonuçlar vermiş olan tedavi yöntemleridir. Bellek tabanlı yöntemler, en etkili tedavi yöntemini bulabilmektedir.

Bellek tabanlı yöntemlerin çalışma mantığı için şöyle bir örnek verilebilir. Örneğin elimizde hangi yaş gruplarının sinemada hangi filmleri izlediğine dair bir veri tabanı bulunsun. Bu bilgileri görselleştirmenin en iyi yolu grafiksel olarak ifade etmektir. Aşağıda örnek verileri içeren bir grafik bulunmaktadır:



Şekil 2.3 – Bellek Tabanlı Yöntemlerin Çalışmasında Kullanılan Örnek Veriler

Burada yer alan bilgiler 4 ayrı topluluktan toplanmış ve hangi yaştaki insanların hangi filmlere gittiği bilgisi biraraya getirilmiştir. Bu bilgiler ışığında, bu topluluklardan herhangi birisinden gelen yeni bir izleyicinin hangi filmi izleme olasılığının daha yüksek olacağını bulmak için bu yeni gelen verilere, en yakın olan komşuları bulunur. Yüksek olasılıkla bu izleyiciler, bu komşu filimlerden bir tanesini izlemeye gidecektir. Bu durum şekil 2.4'de gösterilmiştir.



Şekil 2.4 – Bellek Tabanlı Yöntemlerin Çalışma Prensibi

Şekil 2.4'te de gösterildiği gibi, yeni gelen üç adet izleyici şekilde ? işaretleri ile ifade edilmiştir. Bu yeni gelenlere en yakın komşular bulunmuş ve bu en yakın komşular da halka içerisine dahil edilmişlerdir. Bu komşulardan yola çıkılarak yeni gelen izleyicinin hangi filime gitme olasılığının yüksek olduğuna ilişkin bir tahminde bulunulabilir.

Bellek tabanlı yöntemlerin güçlü olduğu noktalar şunlardır:

- Kolayca anlaşılabilir sonuçlar üretir.
- Rastgele seçilen, hatta ilgisiz dahi olabilen verilere uygulanabilir.
- Analiz alanlarının çok olduğu durumlarda dahi efektif olarak çalışabilir.
- Eğitim kümesinin oluşturulması basittir.

Bellek tabanlı yöntemlerin zayıf olduğu noktalar da bulunmaktadır:

- Sınıflandırma ve kestirim işlemleri için kullanıldığında işlem maliyeti yüksektir.
- Eğitim kümesi için büyük miktarlarda yere ihtiyaç vardır.
- Üretilen sonuçlar seçilen uzaklık fonksiyonuna, kombinasyon fonksiyonuna ve komşu adedine doğrudan bağlıdır.

2.3.3. Demetleme

İnsanoğlu kritik kararlar almadan önce genellikle bir adım geri atar ve "büyük resmi" görmeye çalışır. Ancak zaman zaman bu büyük resim anlaşılmayacak kadar çok karmaşıktır. Büyük bir veri tabanı çok miktarda boyut, yani alan içerebilir ve çok karmaşık bir yapıya sahip olduğundan en iyi yönetilen veri madenciliği teknikleri bile bu veri yığını içerisinden anlamlı sonuçlar üretemeyebilir.

Bizlerin çok karmaşık ve büyük sorunları çözmekte izlediğimiz yöntem genellikle büyük sorunu daha küçük ve tek başına daha rahat çözülebilecek alt sorunlara bölmek ve her bir alt sorunu çözdükten sonra çözümleri birleştirerek sonuca gitmek şeklindedir. Ancak bazı durumlarda veriler öyle dağılmışlardır ki nereden bölüneceği hangi şekilde alt gruplara ayrılabileceğini kestirmek mümkün değildir. Bu yüzden otomatik demet bulma yöntemleri geliştirilmiştir.

Veri madenciliği yöntemlerinden bir tanesi olan demetleme sadece veri madenciliğinde değil örüntü tanıma, imge işleme ve benzeri birçok değişik alanda daha kullanılmaktadır. Demetleme ile ilgili ayrıntılı bir çalışma yapmış ve tezin 3. bölümünde demetleme ile ilgili ayrıntılı bilgiler anlatılmıştır.

2.3.4. İlişkisel Analiz

İş dünyası, insanların, mekanların ve bütün herşeyin birbirleri ile bağlantı kurduğu bir ilişkiler dünyasıdır. Havayolu şirketleri, kargo şirketleri ve benzeri firmalar şehirleri birbirine bağlar. Haberleşme şirketleri ile diğer müşteriler birbirleri ile telefon ve benzeri şekillerde bağlantı kurarlar. Her kredi kartı müşterisi belirli mağazaları ve lokantaları tercih eder. Benzer şekilde ilişkiler her alanda bolca bulunmaktadır ve bu ilişkiler bir çok veri madenciliği tekniğinin kullanamayacağı kadar zengin bilgi içermektedir.

İlişkisel analiz, matematiğin bir alt alanı olan graf teorisi tabanlıdır. Yukarıda anlatılanlara rağmen ilişkisel analiz her tür sorunu çözemez ya da her türlü veri üzerinde uygulanamaz.

İlişkisel analizi gerçeklemek üzere birkaç tane yazılım bulunmaktadır ve bu yazılımların çoğu hukuksal alanda özelleşmişlerdir. En basit ilişkisel analiz aracı olarak ise ilişkisel veri tabanları üzerinde kullanılan SQL gösterilebilir.

İlişkisel analiz yönteminin uygulanmasında ortaya çıkan bir başka sorun ise maddeler arasındaki ilişkilerin ortaya çıkarılmasıdır. Telefon çağrılarının analizinde ilişkiler açıktır: bir kişi bir başkasını arar ve burada çağrının kendisi ilişkidir. Ancak her durumda ilişkileri bulmak bu kadar kolay değildir, ilişkilerin bulunması için

otomatikleşmiş bazı işlemlerin gerçeklenmesi gerekebilir. FBI tarafından birimler arası ilişkilerin ortaya çıkarılmasında MBR'ye benzer bir yöntem kullanılmaktadır.

2.3.5. Karar Ağaçları ve Kural Türetme

Karar ağaçları yöntemi en güçlü ve en yaygın sınıflandırma ve öngörü araçlarından birisidir. Ağaç yapılı yöntemlerin sık kullanılmasının nedeni ise yapay sinir ağlarının tersine ağaç yapılarının kuralları ifade edebilmesinden kaynaklanmaktadır. Kurallar insanların okuyup anlayabileceği herhangi bir dile çevrilebileceği gibi, bir veri tabanında belirli bir kategoriye düşen kayıtların getirilmesi için bir SQL cümleciği haline de getirilebilir.

Bazı uygulamalarda, sınıflandırmanın ya da öngörünün doğruluğu, önemli olan tek şeydir, örneğin doğrudan posta ilanları ile iş yapan bir firma, hangi müşterilerin kendilerine gönderilen ilanlara olumlu yanıt vereceğini öngören bir model sahibi olduğunda bu modelin nasıl veya neden çalıştığını sorgulamaz. North Illinois üniversitesi tarafından geliştirilen bir model de ise hem yapay sinir ağları hem de karar ağaçları bir arada kullanılmışlardır.

Sık oynanan bazı oyunlarda olduğu gibi karar ağaçları da bir dizi soru sorup bunların cevapları doğrultusunda hareket ederek en kısa sürede sonuca gider. Karar ağaçları, sorduğu bir soruya gelen cevap ile soracağı diğer soruları belirler. Eğer sorular iyi seçilmiş olursa, yeni gelen bir kaydın sınıflandırılması işlemi, en az sayıda soru sorarak gerçeklenebilir.

Sorulacak sorular ve bu sorulara gelebilecek cevapların yönlendirdiği başka soruların bulunduğu bir ağaç yapısı olarak adlandırılan karar ağaçları ile değerlendirme yaparken yeni gelen bir kayıt, ağacın kökünden girer. Kökte test edilen bu yeni kayıt bu test sonucuna göre bir alt düğüme gönderilir. Bu süreç, yeni kayıt herhangi bir yaprak düğüme gelene kadar devam eder. Ağacın belirli bir yaprağına gelen bütün yeni kayıtlar aynı şekilde sınıflandırılırlar. Kökten her bir yaprağa giden sadece tek bir yol vardır. Bu yol, kayıtları sınıflandırmak için kullanılan bir kuralı tanımlamaktadır. Bazı yapraklar aynı sınıflandırmayı yapabilirler fakat her bir yaprak bu sınıflandırmayı farklı nedenlere dayanarak yaparlar.

Karar ağaçlarının güçlü olduğu noktalar şunlardır:

- Karar ağaçları kolay anlaşılır sonuçlar üretir.
- Çok sayıda işlem yapılmasına gerek duymadan sınıflandırma işlemini gerçekleyebilmektedir.

- Hem sayısal hem de kategorik veriler üzerinde işlem yapabilmektedir.
- Karar ağaçları hangi alanların sınıflandırma ve kestirme için daha önemli olduğunu açık şekilde belirtmektedir.

Karar ağaçlarının zayıf olduğu noktalar ise şöyledir:

- Karar ağaçlarının tahmin için kullanıldığı durumlarda tahmin edilecek değişkenin sürekli değerler alması durumunda uygun sonuçlar üretilememektedir.

2.3.6. Yapay Sinir Ağları

Yapay sinir ağlarının popülaritesi, veri madenciliği ve karar destek sistemlerinde önceden kanıtlanmış başarılarından gelmektedir. Yapay sinir ağları, sınıflandırma, demetleme ve kestirim amaçları ile kolaylıkla kullanılabilecek genel amaçlı ve güçlü araçlardır. Ekonomik alanlardan tıbbi konulara, değerli müşterilerin belirlenmesi için yapılan demetleme işlemlerinden kredi kartlarında sahtekarlıkların belirlenmesine kadar çok geniş bir alanda uygulama alanı bulmuştur.

Yapay sinir ağlarını, insanların deneyimlerinden bir takım bilgiler çıkartması gibi kendisine verilen örneklerden bir takım bilgiler çıkartma yeteneğine sahiptir. Yapay sinir ağları öncelikle sonuçları bilinen belirli bir veri kümesi üzerinde öğrenme algoritmaları çalıştırılarak eğitilir. Bu eğitim neticesinde yapay sinir ağının içerisindeki bir takım ağırlıklar belirlenir. Bu ağırlıklar kullanılarak yeni gelen veriler işlenir ve bir sonuç üretilir. Yapay sinir ağlarının en olumsuz tarafı ise bu ağırlıkların neden ilgili değeri aldığının bilinmemesidir. Ya da çıkan sonucun neden geçerli bir sonuç olduğunu bize açıklayamaz. Yapay sinir ağlarını kullanmak için en iyi yaklaşım onları içi bilinmeyen bir şekilde çalışan kara kutular olarak düşünmek olacaktır.

Yapay sinir ağlarının veri madenciliği açısından kuvvetli yönleri şunlardır:

- Çok geniş spektrumdaki sorunların çözümünde kullanılabilirler.
- Çok karmaşık durumlarda dahi iyi sonuçlar üretmektedirler.
- Hem sayısal hem de kategorik veriler üzerinde işlem yapabilirler.

Bütün bu olumlu özelliklerine rağmen yapay sinir ağlarının olumsuz yönleri de vardır:

- 0 ile 1 arasında giriş verileri olması zorunludur.
- Ürettikleri sonuçların açıklamasını yapamazlar.
- Varılan sonucun olası en iyi sonuç olduğunun garantisi yoktur.

2.3.7. Genetik Algoritmalar

Genetik algoritmalar da yapay sinir ağları ve bellek tabanlı yöntemler gibi biyolojik işlemlerden kaynağını almıştır. Yüzyıllar boyu süren adaptasyonlar ve doğal seleksiyon ile çevre koşullarına en fazla uyum sağlayanlar hayatta kalmışlardır. Genetik algoritmaların da benzer bir çalışma biçimi vardır. Geçtiğimiz yıllar boyunca genetik algoritmalar yapay sinir ağlarının eğitilmesi, bellek tabanlı yöntemlerde kombinasyon fonksiyonunun oluşturulması gibi işlerde kullanılmışlardır.

Genetik algoritmalar açıklanabilir sonuçlar üretirler. Çok değişik tiplerdeki verileri işleme özelliğine sahip olan genetik algoritmalar optimizasyon amacı ile kullanılabilirler. Ayrıca genetik algoritmalar yapay sinir ağları ile ortaklaşa çalışarak başarılı sonuçlar üretmektedirler.

Tüm bu güzel yönlerine rağmen genetik algoritmaların kullanılmalarında bazı sıkıntılar da yaşanmaktadır. Bunlardan en belirgin olanı karmaşık sorunların genetik kodlanmasının çok zor olmasıdır. Ayrıca optimal sonucun üretildiğine dair bir garanti de bulunmamaktadır. Son olarak genetik algoritmaların çalıştırılması çok ağır bir işlem yükü getirmektedir.

3. DEMETLEME

3.1. Tanımı

Büyük ölçekli verileri homojen demetlere ayırma işlemi veri madenciliğinin en temel işlemlerinden birisidir. Gözetimsiz sınıflandırma işlemi, verileri özetleme, heterojen yapıya sahip büyük veri yığınlarını daha kolay yönetilebilir ve işlenebilir daha küçük homojen alt kümelere ayırma işlemleri gibi birçok iş için öbeklere ayırma işlemi en temel işlemlerden bir tanesidir. Demetleme (clustering) yöntemi ise öbeklere ayırma işleminin en yaygın olarak kullanılan yöntemidir. Demetleme yöntemi, verilen belirli bir örüntü kümesini alt kümelere (clusters) böler. Bu bölme işlemi sonucunda, aynı demete giren kayıtlar arasındaki benzerlik diğer demetlerdeki kayıtlara göre çok daha fazla olur. Burada geçen "benzerlik" kavramı, çeşitli tanımlara sahip olabilir. En yaygın olarak kullanılan benzerlik kriteri "Euclid Uzaklığı"dır. (Huang 1997)

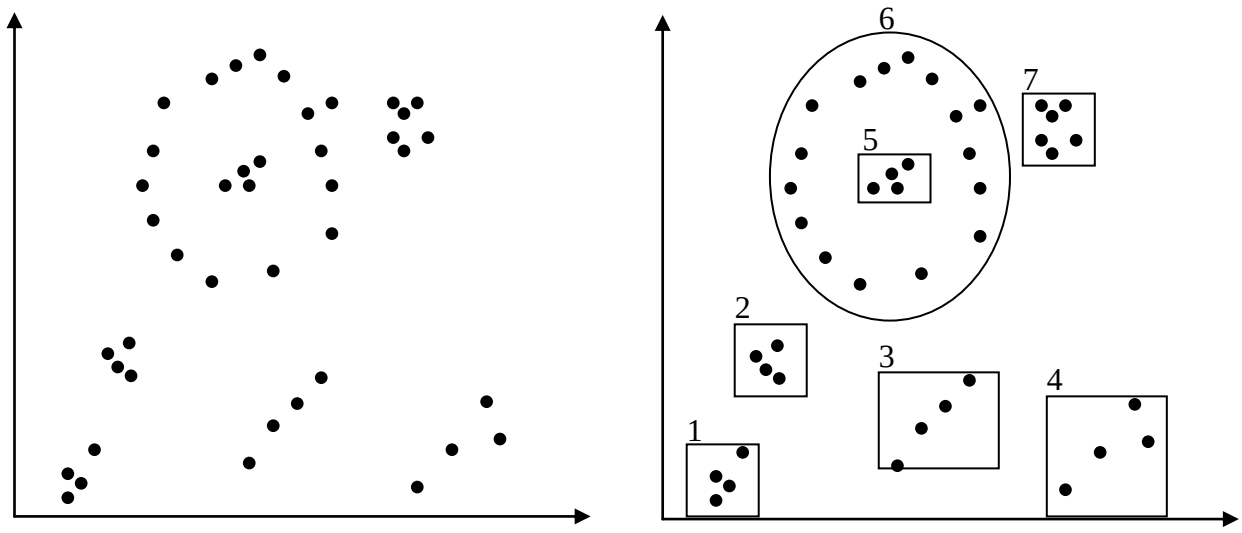
Veri madenciliğinin en belirgin özelliği, veri madenciliğinde işlenen verilerin boyutlarının çok büyük olmasıdır. Çoğu zaman işlenen veriler giga-sekizli (gigabyte) ve hatta tera-sekizli (terabyte) boyutunda olmaktadır. İşlenmesi gereken verilerin bu denli çok olması, veri madenciliğinde kullanılan tekniklerin, **ölçeklenebilir** olması zorunluluğunu ortaya koymaktadır. Ancak veri madenciliğinde kullanılan tekniklerin çoğu, geniş ölçekli veri kümeleri üzerinde gerçeklendiklerinde, ölçeklenebilirlik özelliğini yeteri kadar sağlayamamaktadırlar. Bunun en açık nedeni ise bu tekniklerin çoğunun veri madenciliğinden çok daha farklı amaçlar için geliştirilmiş olmalarıdır. Bu amaçlar doğrultusunda çalışıldığında, veri madenciliğine oranla çok daha küçük ölçekli veriler yer aldığından, bu teknikler büyük ölçekli veri kümeleri üzerinde yeteri kadar başarılı olamamışlardır (Huang 1997). Ölçeklenebilir veri madenciliği algoritmaları geliştirilmesi ise şu sıralar veri madenciliğinde yer alan önemli bir araştırma konusu olmuştur (Shafer 1996).

Demetleme işleminin kullanıldığı alanlar şu şekildedir:

- Örüntü tanıma
- Konuşma tanıma
- Parmak izi tanıma

- Coğrafya: Konumsal veriden yararlanılarak bölgeler arasındaki benzerliklere göre bölgeleri gruplandırma
- Biyoloji: Genlerin analizi ve genleri işlevlerine göre gruplandırma.
- Ekonomi: Müşterileri gruplandırma ve pazar araştırması
- www: Belge sınıflandırma.
- Diğer veri madenciliği algoritmaları için bir önişleme.
- Pazarlama: Müşterileri sınıflandırarak, bu bilgi pazar büyültme
- Şehir planlaması: Konumlarına, değerlerine ve türlerine göre binaları gruplandırma.
- Deprem araştırmaları: Sürekli fay hatları belirlenerek deprem merkezlerinin gruplandırılması

Gözetimsiz sınıflandırma ya da demetleme (clustering, unsupervised classification) ile gözetimli sınıflandırma (supervised classification) arasındaki farkı anlamak çok önemlidir. Gözetimli sınıflandırma yönteminde bize verilen veriler önceden sınıflandırılmış (pre-classified) örüntülerdir. Burada sorun, yeni gelecek ve henüz hangi bir sınıfta olduğu bilinmeyen bir verinin var olan sınıflardan uygun olanına yerleştirilmesidir. Genellikle başlangıçta sağlanan sınıflara ayrılmış veri, bir eğitim örüntüsü olarak kullanılır ve bu eğitim sonucunda daha sonra, yeni gelecek bir verinin hangi sınıfa girmesi gerektiğine karar verecek ölçüde, sınıflar ile ilgili özellikler çıkartılır. Demetleme işleminde ise sorun, başlangıçta verilen ve henüz sınıflandırılmamış bir küme veriyi anlamlı alt kümeler oluşturacak şekilde öbeklemektir. Burada kümeleme işlemi tamamen gelen verinin özelliklerine göre yapılmaktadır (data driven) yani kümelere ayırma işlemi sadece eldeki veriler kullanılarak yapılır. (Jain ve diğ. 1996)



Şekil 3.1 – Veri Demetleme Örneği

Basit bir demetleme örneği yukarıda verilmiştir. Bu örnekte soldaki grafikte yer alan veriler giriş olarak verilmiş ve demetleme işlemi sonucunda ortaya çıkan 7 adet demet (cluster) sağda gösterilmiştir.

3.2. Demetlemenin İşleminin Temel Adımları

Genel bir kümeleme işleminde gerçeklenmesi gereken birkaç adım vardır (Jain ve Dubes 1988):

1. Örüntü seçimi (özellik çıkartma ya da seçme)
2. Veri kümesinde benzerliğinin ölçümünde kullanılacak uygun yöntemi seçme
3. Demetleme işlemi
4. Sonuçların özetlenmesi (gerekli ise)
5. Çıkışların saklanması (gerekli ise)

3.2.1. Örüntü Seçimi:

Örüntü seçimi kapsamında, demet sayısının seçimi, örüntü kümesi büyüklüğü, demetleme algoritmasında kullanılabilecek kayıt özelliklerinin sayıları, tipleri ve ölçekleri gibi bilgiler doğrultusunda demetleme işleminde kullanılacak özelliklerin seçimi yer alır.

3.2.1.1. Özellik Seçimi (Feature Selection)

Demetleme algoritmasında kullanılmak üzere, örüntü kümesi içerisinde varolan özellikler (kayıt alanları) içerisinde demetleme işleminde kullanılması en uygun olan

ve proses sonunda en verimli demetleri üretecek ilgili özelliklerin seçilmesidir. Veri ambarı üzerinde gerçeklenmesi durumunda ise değişik tablolardan değişik alanlar bir araya getirilerek demetlenecek veriler oluşturulur.

3.2.1.2. Özellik Çıkarımı (Feature Extraction)

Bir ya da birkaç özelliğin, demetleme algoritmasında kullanılmak üzere yeni bir yapay özelliğe dönüşecek şekilde işlem görmesinin sağlanmasıdır.

Bu iki teknikten herhangi birisi ya da her ikisi birden, demetlemede kullanılacak özelliklerin en uygun kümesini oluşturmak için kullanılabilir. (A.K. Jain ve diğ. 1999).

3.2.2. Benzerlik Yöntemi Seçimi

Veri demetlemede örüntü içerisindeki çiftlerin birbirlerine olan benzerliklerinin ya da aykırılıklarının belirlenmesi için bir uzaklık fonksiyonu tanımlanır. Değişik çevreler tarafından birkaç adet uzaklık fonksiyonu kullanılmıştır. (Anderberg 1973, Jain ve Dubes 1988, Diday ve Simon 1976). İki örüntü elemanı arasındaki uzaklığın bulunması için en basit yöntem olan Euclid Uzaklığı fonksiyonu kullanılabileceği gibi örüntü elemanları üzerinde kavramsal uzaklıkları bulan başka yöntemler de kullanılabilir (Michalski ve Stepp 1983).

3.2.3. Demetleme İşlemi

Demetleme işlemi temelde 2 şekilde gerçeklenebilir. Giriş verisi kesin sınırlarla demetlere ayrılacak şekilde ***keskin*** (hard) olarak ya da her örüntü elemanının her demete ne kadar yakın olduğunu belirlendiği ***bulanık*** (fuzzy) olarak gerçeklenebilir. Hiyerarşik demetleme yöntemlerinde ise benzerlik kriterine göre bölünen yahut birleştirilen demetlerden oluşmuş bir dizi ardışıl bölmelendirme işlemi gerçeklerler. Bölmeleme esasına göre çalışan algoritmalar genellikle yerel olarak benzerlik kriterini en fazla sağlayacak şekilde gruplandırma oluştururlar. (A.K. Jain e diğ. 1999). Demetleme işlemi için olasılık tabanlı (Brailovski 1991) ve graf teorisi tabanlı (Zahn 1971) kümeleme teknikleri de bulunmaktadır.

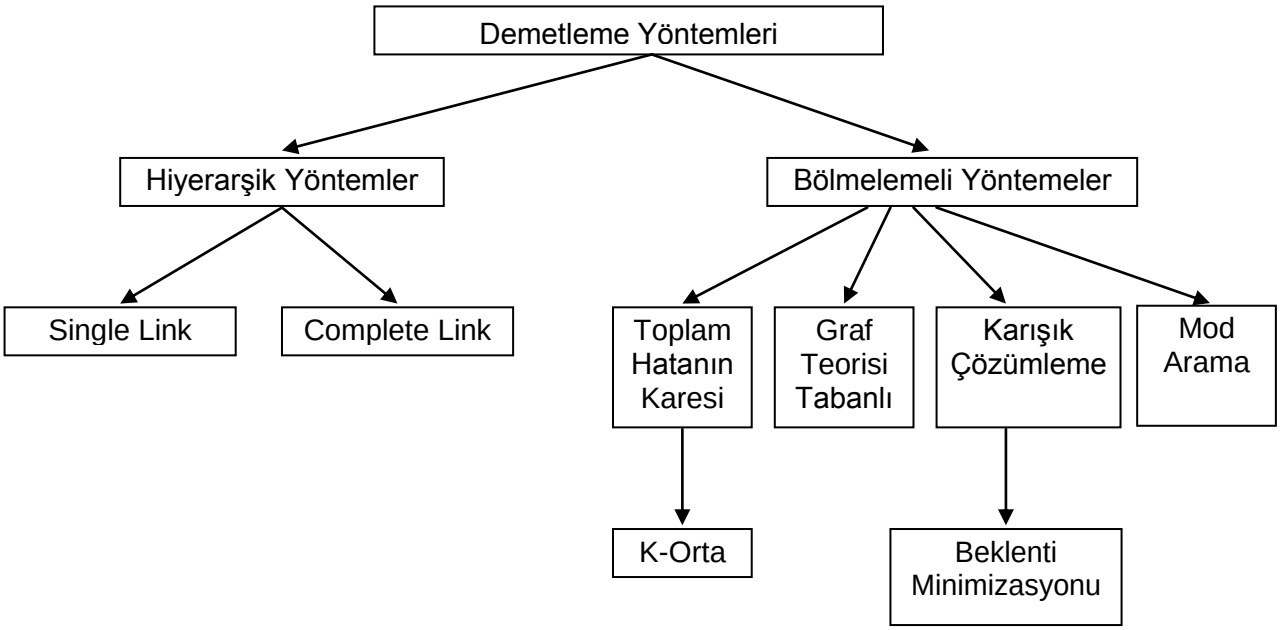
3.2.4. Sonuçların Özetlenmesi ve Saklanması

Demetleme işlemi sonucunda ortaya çıkan bölmelendirme sonuçlarının basit, anlaşılır ve kompakt bir yapıda sunulması adımıdır. Bu noktada basitlik kavramı iki anlam taşımaktadır: Ya bu kümeleme işlemlerinin sonucu bir başka algoritma tarafından giriş verisi olarak kullanılacaktır ya da bu sonuçların yorumlanmasını yönetici sıfatını taşıyan birisi yapacaktır. Bölmeleme sonuçlarının bir sonraki adımda başka bir algoritma tarafından giriş verisi olarak kullanılması durumunda

otomatikleştirilmiş analizler gerçeklenebilir. Demetleme işleminin sonuçlarının insan tarafından yorumlanması durumunda ise sonuçların özetlenmesi adımında üretilecek olan özetlerin anlaşılır ve aydınlatıcı olması istenir. Demetleme sonucu oluşan verilerin özetlenmesi işleminde genelde her demeti tanımlayan kriterlerin bir özeti hazırlanır. Bunun için örneğin her demetin temsilci elemanının (prototype), örneğin oluşan kümenin merkezinin, özellikleri (features) özetlenebilir (Diday ve Simon 1976).

3.3. Demetleme Algoritmalarının Yapısı

Demetleme yöntemleri temel olarak şu şekilde gruplandırılabilir:



Şekil 3.2 – Demetleme Yöntemleri Şeması

Şekilden 3.2.'den de incelenebileceği üzere birçok demetleme yöntemi vardır. Bu yöntemler çeşitli özellikleri bakımından sınıflandırılabilirler.

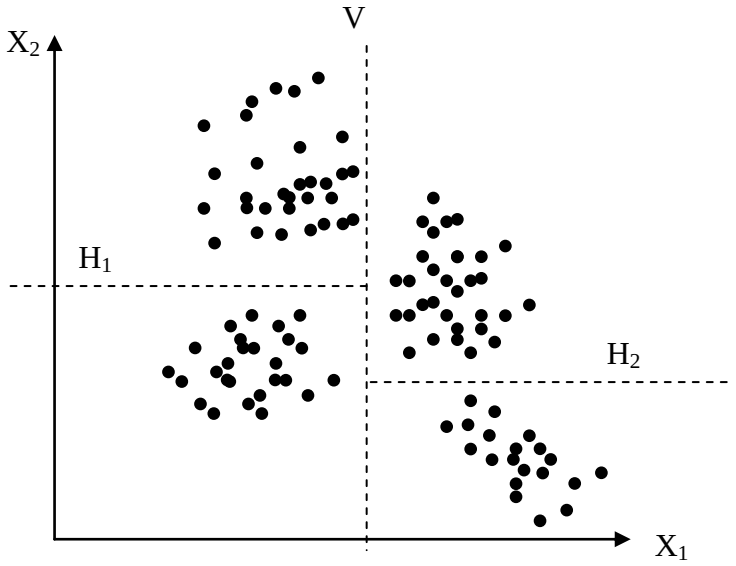
3.3.1. Bölmelemeli / Birleştirmeli Yöntemler

Mevcut demetleme yöntemlerinin hepsi ya bölmelemeli yöntemdir ya da birleştirmeli bir yöntemdir. Bu özellik doğrudan doğruya algoritmik yapı ile ilgilidir. Birleştirmeli bir yöntem ilk başladığında demetleme yapacağı veride kümesindeki her bir kaydı, tek elemanlı bir demet olarak belirler. Bu türden algoritmalar birbirine benzer demetleri arka arkaya birleştirmek sureti ile en sonunda bütün kayıtları içeren tek bir demete ulaşıncaya kadar ya da bir durma kriteri sağlanıncaya kadar (toplam demet sayısı n olunca) ilerler. Bölmelemeli yöntemler ise bütün kayıtları içeren tek bir demeti işleyerek başlar ve durma kriteri sağlanıncaya kadar bu büyük demeti bölmelere ayırarak ilerler.

3.3.2. Tek Etkili (Monothetic) ve Eş Etkili (Polythetic) Yöntemler

Demetleme yöntemleri arasında, demetleme işlemi devam ederken, her bir özelliğin aynı anda mı yoksa sırasıyla mı hesaplamaya dahil edileceği yönünde bir ayrım vardır. Çoğu teknik eş etkili çalışmaktadır. Yani örüntü içerisindeki iki ayrı kaydın arasındaki uzaklığın hesaplanması sırasında tüm özellikler eş zamanlı olarak

hesaplamaya girmektedir. Oysa Anderberg (1973) tarafından önerilen bölmeleme yönteminde her özellik hesaplamaya sırayla girmektedir. Örnek olarak aşağıdaki demetleme işlemi verilebilir:



Şekil 3.3 – Tek Etkili Kümeleme

Bu demetleme yönteminde tüm örüntü kümesi öncelikle X_1 özelliği kullanılarak V çizgisi ile ikiye bölünmüştür. Daha sonra bu iki parça X_2 özelliği kullanılarak kendi içlerinde H_1 ve H_2 çizgileriyle ikiye bölünmüştür. Görüldüğü üzere bölmeleme işleminde her özellik sırasıyla hesaba girmektedir. Bu tür yöntemlerin en büyük sorunu d boyutlu örüntü kümesinde 2^d adet demet üretmesidir. Büyük d değerleri için (bilgi çıkarım uygulamalarında genellikle d>100 [Salton 1991]) örüntü kümesi çok küçük ve pek fazla anlam taşımayan çok sayıda alt kümeye bölünmüş olur.

3.3.3. Keskin (Hard) / Bulanık (Fuzzy) Yöntemler

Keskin demetleme yöntemleri algoritma boyunca ve çıkış verisi olarak her bir kaydı kesin olarak varolan demetlerden birine dağıtır. Bulanık demetleme yöntemleri ise her bir örüntü elemanı için mevcut tüm demetler için bir üyelik katsayısı hesaplar. Bulanık demetleme yöntemine göre demetlenmiş veriler, her örüntü elemanını, en büyük üyelik katsayısına sahip kümeye atamak suretiyle keskin demetleme yöntemine çevrilebilirler.

3.3.4. Artımlı / Artımsız Yöntemler

Bu kriter, demetlenecek veri yığınının çok büyük olduğu durumlarda ve özellikle çalışma süresine ya da bellek kullanımına ilişkin kısıtlamalar olduğu durumlarda algoritmanın mimarisini etkilemektedir. İlk öbekleme yöntemleri çok büyük veri

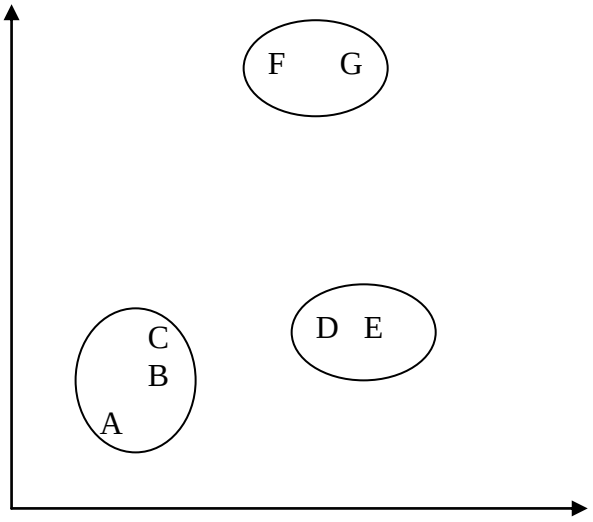
kümeleri üzerinde çalışmayı hedef almadığından dolayı, nispeten daha yeni olan veri madenciliği konularında çok büyük ölçekli veri kümeleri üzerinde demetleme yapmak için yeni yöntemler geliştirilmekte, algoritma esnasında veri kümesinin üzerinden geçiş sayısının azaltılmasına, işlenecek veri sayısının düşürülmesine ve algoritmanın kullandığı veri yapılarının boyutlarının azaltılmasına çalışılmaktadır. (Jain ve diğ. 1999).

3.4. Demetleme Algoritmaları

3.4.1. Hiyerarşik Yöntemler

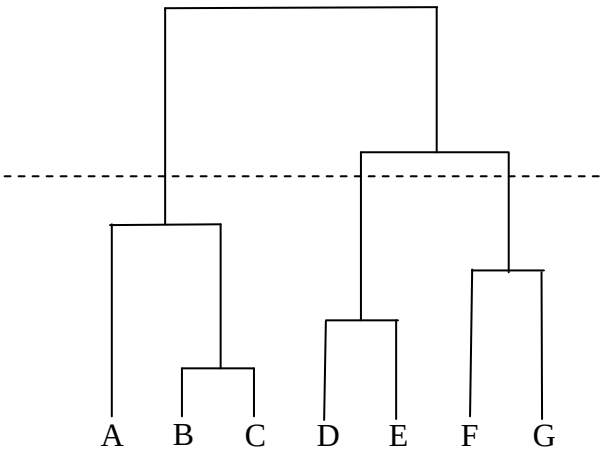
Hiyerarşik demetleme yöntemlerinin tümü ilk adımda bütün kayıtları, tek kayıttan oluşan ayrı bir demet olarak değerlendirir. Daha sonra bu demetler arasında uzaklıkları en az olan iki tanesini birleştirerek bir tane iki elemanlı demet haline getirir. Bunu izleyen her adımda, varolan demetler arasında, birbirlerine uzaklıkları minimum olan iki adet demet birleştirilerek tek bir demet haline getirilir. Bu birleştirme işlemleri, daha önceden belirlenen sayıda demet kalana kadar devam eder.

Aşağıda iki boyutlu bir özellik uzayında gerçeklenen bir hiyerarşik demetleme algoritması özetlenmiştir. Şekil 3.4.'de yer alan kayıtlar hiyerarşik demetleme yöntemi ile gruplanmışlardır.



Şekil 3.4 - Hiyerarşik Demetleme

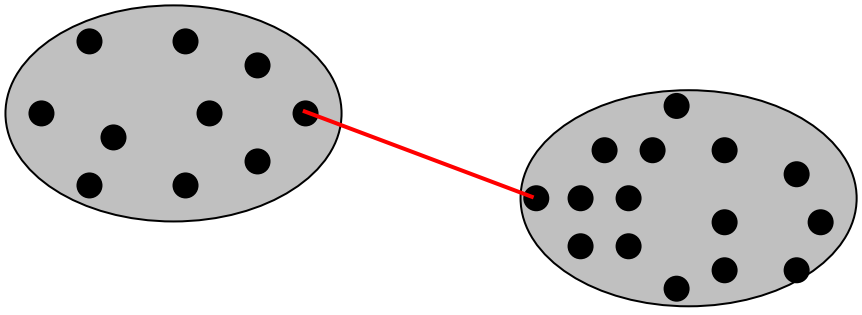
Hiyerarşik yöntemlerde her adımda hangi kayıtların bir araya getirilerek demetlendiğini gösteren DENDOGRAM (birleştirme ağacı) yapıları oluşturulur. Yukarıda Şekil 3.4. de yer alan kayıtlara ilişkin dendogram şu şekildedir:



Şekil 3.5 – Hiyerarşik Demetlemede Oluşan Birleştirme Ağacı

Şekildeki birleştirme ağacından da görüleceği üzere ilk adımda B-C, ikinci adımda D-E, üçüncü adımda F-G, dördüncü adımda A-BC birleştirilmiştir. Sonraki adımda DE-FG ile ve en son adımda da ABC-DEFG birleştirilerek bütün kayıtları içeren en büyük demet oluşturulmuştur. Adını bu hiyerarşik birleştirme ağacından alan hiyerarşik yöntemlerin sonlanması, daha önceden belirlenen demet sayısına bağlıdır. Örneğin bu birleştirme ağacında 3 demet oluşacak şekilde bir demetleme gerçeklenmiştir. Hiyerarşik yöntemlerin en güzel yanlarından bir tanesi de bir kez bu birleştirme ağacının oluşturulmasından sonra ağacın istenilen seviyelerden (istenilen sayıda demet sağlayacak şekilde) ayrılabilmesidir (Jain ve Dubes1988).

Çoğu hiyerarşik demetleme yöntemi, single-link (Sneath ve Sokal 1973), complete-link (King 1967) ve minimum-varyans (Ward 1963) algoritmalarının değişik türevleridir. Bunlardan single-link ve complete-link algoritmaları en yaygın kullanılan olanlarıdır. Bu iki algoritma arasındaki fark, birleştirme işleminde iki demet için hesapladığı benzerlik (uzaklık) çıkarma yönteminden ileri gelmektedir. Single-link algoritmasında iki demet arasındaki uzaklık, her iki demet arasında yer alan kayıtlardan birbirlerine en *yakın* olanların uzaklığı olarak değerlendirilmektedir. Bu işlem şekilde gösterilmiştir:

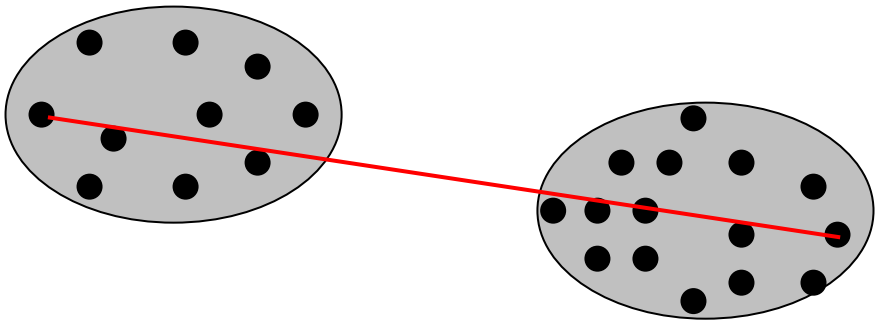


Şekil 3.6 - Single-Link Demetlemede İki Demet Arası Uzaklık

Single-link yöntemine göre iki demet arasındaki benzerliğin bulunmasında kullanılan uzaklık fonksiyonu aşağıdadır:

$$d_{kl} = \min_{i \in C_l, j \in C_k} d(x_i, x_j) \tag{3.1}$$

Complete-link algoritmasında iki demet arasındaki uzaklık hesap edilirken, bu iki demet içerisinde birbirlerine en *uzak* iki kaydın arasındaki uzaklık, bu iki demet arasındaki uzaklık olarak alınır.



Şekil 3.7 - Complete-Link Demetlemede İki Demet Arası Uzaklık

$$d_{kl} = \max_{i \in C_l, j \in C_k} d(x_i, x_j) \tag{3.2}$$

Her iki yöntem sonunda da birbirlerine en yakın (en benzer) iki demet birleştirilerek daha büyük tek bir demet haline getirilir. Single-link algoritması, complete-link algoritmasına oranla daha kullanışlı bir yöntemdir. Örneğin bu yöntem içiçe geçmiş demetleri bulabilirken, complete-link yöntemi bu demetleri bulamamaktadır. Ancak her ne kadar teorik olarak single-link yöntemi daha kullanışlı gözükse de bir çok uygulamada complete-link algoritmasının daha işlevsel olduğu ve ürettiği birleştirme ağaçlarının kullanılabilirlik oranının daha yüksek olduğu belirlenmiştir (Jain ve Dubes 1988).

Bunların dışında, iki demet arası benzerliğin hesaplanması için iki yöntem daha vardır. Bunlardan ilki, iki demet içindeki bütün noktaların birbirleri ile arasındaki uzaklıkların ortalamasını, iki demet arası uzaklık olarak almaktır (average-linkage). Bu uzaklığın formülü aşağıda verilmiştir:

$$d_{kl} = \frac{1}{|C_l \| C_k|} \sum_{i \in C_l, j \in C_k} d(x_i, x_j) \tag{3.3}$$

Son uzaklık formülü ise, iki demet arasındaki benzerliği bulmadan önce bu iki demet içerisindeki verilerden bir ortalama alarak demet merkezlerini bulmaktadır. Daha

sonra bu demet merkezleri arasındaki uzaklık, iki demet arası uzaklık olarak kullanılır:

$$d_{kl} = \max\ d(\bar{x}_i, \bar{x}_j), \qquad \bar{x}_l = \frac{1}{|C_l|}\sum_{i \in C_l} x_i \tag{3.4}$$

3.4.2. Bölmelemeli Yöntemler

Bölmeleme ile demetleme yöntemleri, hiyerarşik demetleme yöntemlerinde kullanılan birleştirme ağacı (dendogram) kullanmak yerine tüm kayıtların belirli bir şekilde bölmelenmesi ile demetler oluşturur. Demetlenmesi gereken çok sayıda kaydın bulunduğu uygulamalarda, birleştirme ağacının oluşturulması gibi çok ağır bir iş yükü getiren bir algoritma yerine bölmeleme yöntemleri daha verimli olmaktadır. Ancak bölmeleme yöntemlerinin zayıf tarafları da vardır. Bunlardan en önemlisi, demetleme işlemi sonucunda oluşması istenilen demet adedinin işlemden önce algoritma tarafından bilinmesinin gerekliliğidir.

Bölmeleme yöntemleri demetleri oluştururken çoğunlukla yerel ya da global olarak tanımlanmış olan bir benzerlik kriterini optimize etmeye çalışır. Bu fonksiyonun muhtemel optimum değerini bulmak için tüm kayıtlar üzerinde çeşitli kombinasyonlar denemek mümkün değildir. Bunun yerine pratik alanda birkaç başlangıç durumu için algoritma işletilir ve sonuç demetleri de bu önceki çalışmaların sonuçlarından elde edilerek oluşturulur.

3.4.2.1. K-Means

Bölmeleme ile demetleme yöntemlerinde en yaygın ve sıklıkla kullanılan benzerlik kriteri fonksiyonu, hatanın karesi (squared error) kriteridir. Bu kriter izole ve kompakt demetlerin bulunduğu veri kümeleri üzerinde en iyi sonucu vermektedir. Bu fonksiyona göre K adet demet içinde toplam X tane kayıt içeren bir örüntü kümesi üzerinde β demetlemesindeki hatanın karesi şu şekilde hesaplanır:

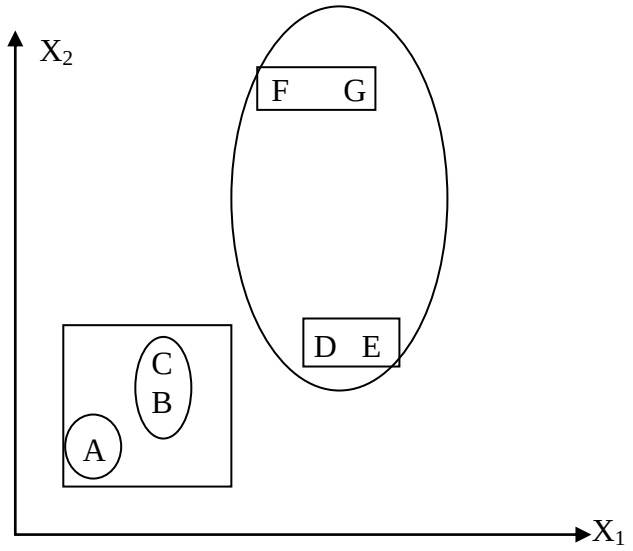
$$e^2(\chi, \beta) = \sum_{j=1}^{K}\sum_{i=1}^{n_j}\left\| x_i^{(j)} - c_j \right\|^2 \tag{3.5}$$

Burada $x_i^{(j)}$, j. demet içerisinde yer alan i. kaydı, c_j ise j. demetin merkez noktasını ifade etmektedir.

Hatanın karesini azaltma kriteri ilkesince çalışan en basit ve yaygın kullanılan algoritma K-Means algoritmasıdır (MacQueen 1967). Bu algoritmada öncelikle rastgele olarak başlangıç noktaları seçilir ve her adımda, demetlenmek istenilen örüntü kümesinin elemanları, bu başlangıç noktalarına olan uzaklıklarının değerine

göre en yakın olduğu demete atanırlar. Her adım sonunda da belirli bir demet içerisine giren kayıtların tüm özelliklerinin ortalaması alınarak bu demetin yeni merkezi hesap edilir. Bu işlemler, belirli bir durma kriterine yakınsayıncaya dek devam eder. Durma kriteri olarak belirli bir iterasyonda hiç bir demet elemanının başka demete atanmaması ya da hesaplanan uzaklıkların tümünün önceden belirlenen belirli bir uzaklığın altında olması verilebilir. K-Means algoritmasının sık kullanılmasının nedeni, uygulanmasındaki basitlik ve zaman karmaşıklığının *O(n)* olmasıdır.

K-Means algoritmasının en önemli zayıf noktası, başlangıçta rastgele olarak seçilen demet merkezlerinin seçimine çok duyarlı olmasıdır. Bu ilk değerlerin uygunsuz olarak seçilmesi durumunda algoritma, toplam hatanın karesi kriterinin yerel minimumunu bulmaktadır. Örnek vermek gerekirse aşağıda iki boyutlu olarak verilen analizde başlangıç noktaları olarak A, B, ve C seçilirse K-Means algoritmasının ürettiği demetler elips şekilleri ile gösterilmiştir. Bu durumda üretilen demetler {A}, {B,C} ve {D, E, F,G} olacak şekilde üretilmiştir.



Şekil 3.8 – K-Means Demetlemede Başlangıç Noktası Seçimi

Oysa bu verilerin yer aldığı bir örüntü kümesinin optimum bölmelenmesi {A,B,C}, {D,E} ve {F,G} biçimindedir. Bu bölmelenme şekilde dikdörtgenler ile gösterilmiştir ve göz ile de rahatlıkla yapılabilmektedir. Oysa ki ilk durumda üretilen bölmelemede toplam hatanın karesi fonksiyonunun değeri, ikinci bölmelemede elde edilen değerden daha büyüktür. Yani ilk demetleme işlemi sonucunda bu fonksiyonunun yerel bir minimumu elde edilmiştir ancak bu değer global minimum değildir. Global minimum değerini bulmak için seçilmesi gereken başlangıç noktaları A, D ve F olmaktadır. Görüldüğü gibi başlangıç demetlerinin merkez noktalarının seçimi,

algoritmanın optimum demetlemeyi sağlayacak şekilde toplam hatanın karesi fonksiyonunu minimize etmesi üzerinde doğrudan etkilidir.

K-Means algoritmasının adımları şu şekildedir:

---

1. *k demet merkezini k adet rasgele seçilen veriye ya da veri topluluğunu içeren uzayda k adet rasgele tanımlanan noktaya uygun şekilde seç.*
2. *Her bir veriyi en yakın demete ata.*
3. *Mevcut demet üyelik değerlerini kullanarak yeniden demet merkezlerini hesapla.*
4. *Eğer bir yakınsama koşulu karşılanmamışsa 2. Adıma geri dön. Tipik bir yakınsama ölçütü: toplam hatanın karesinin minimum olması veya verilerin yeni demetlere minimal düzeyde atanması veya hiç atanmaması. (Ölçüt fonksiyonunun optimum değeri)*
5. *Demet sayısını mevcut demetleri birleştirerek, parçalayarak, küçük veya sınır dışındaki demetleri silerek demet sayısını düzelt.*

Şekil 3.9 - K-Means Demetleme Algoritması

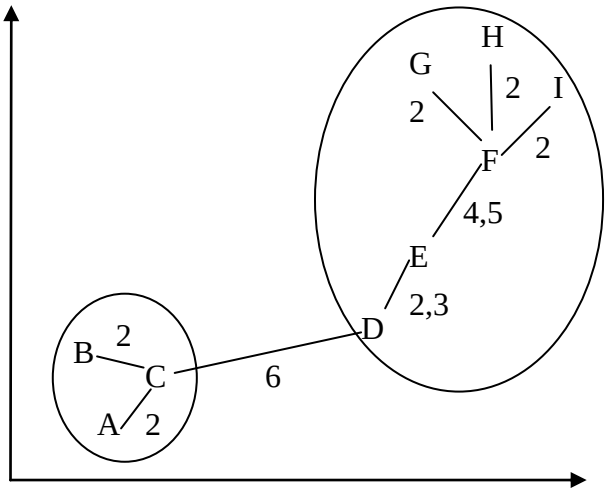
Literatürde K-Means algoritmasının birkaç türevi bulunmaktadır (Anderberg 1973). Bunlardan bazıları, toplam hatanın karesi fonksiyonunun global minimumunu sağlayacak şekilde daha doğru başlangıç noktalarının seçimi konusunda çalışmalar yapmışlardır.

K-Means algoritmasının bazı sürümlerinde ise her iterasyonda oluşan demetlerin silinmesi ya da birleştirilmesi yoluna gidilmiştir. Teorik olarak bir demet, demet içi varyansın belirli bir eşik değerinin altında olması durumunda ikiye bölünmekte, benzer şekilde iki demetin merkez noktaları arasındaki uzaklığın belirli bir eşik seviyesinin altında olması durumunda ise bu iki demet birleştirilmektedir. Bu genişletmeler kullanılarak uygun eşik değerleri ile herhangi bir başlangıç noktası ile optimal çözümler elde edilebilmektedir. Demet birleştirme ve bölme tekniklerini ISODATA (Ball ve Hall 1965) algoritması da kullanmaktadır. Örneğin yukarıdaki şekil 3.8.'de elipsler ile ifade edilen yerel minimumu bulan çözüm ISODATA algoritması ile işlenseydi, algoritma öncelikle {A} ve {B, C} demetleri, merkezlerinin birbirine çok yakın olmasından dolayı birleştirilirdi. Daha sonra da {D, E, F, G} demeti ise demet içi varyans çok büyük olduğundan dolayı {D,E} ve {F,G} şeklinde iki ayrı demete bölünürdü. Dolayısı ile sonuçta global çözüm sağlanmış ve optimum demetleme sonucu elde edilmiş olur.

K-Means algoritmasına benzer bir başka teknik de demet merkezlerinin yerine demet içerisinde merkeze en yakın elemanın uzaklık hesaplarında kullanıldığı dinamik demetleme yöntemidir. Bu yöntem en yakın komşu algoritmasına benzer bir şekilde çalışmaktadır (Diday 1973 ve Symon 1977).

3.4.2.2. Graf Tabanlı Demetleme

En çok bilinen graf tabanlı bölmeleme esasına göre işleyen demetleme algoritması, minimal kapsayan ağacın (MST) oluşturulması üzerine bina edilmiştir. Minimal kapsayan ağaç oluşturulduktan sonra bu ağacın en uzun olan ayrıtları silinerek demetler oluşturulmaktadır. Bu süreç, Şekil 3.10'da gösterilmiştir:



Şekil 3.10 – Graf Tabanlı Demetleme

Şekilde yer alan minimal kapsayan ağaç, iki boyutlu özellik uzayında yer alan 9 adet veri üzerinde oluşturulmuştur. Bu ağaç üzerinde 6 birimlik uzunluğu ile en uzun ayrıt olan CD ayrıtının silinmesi sonucunda {A, B, C} ve {D, E, F, G, H, I} şeklinde iki ayrı demet elde edilir. Daha sonra iki ayrı grafta yer alan 4,5 birimlik en uzun ayrıt olan EF ayrıtı silinirse {D, E, F, G, H, I} demeti de {E, D} ve {F, G, H, I} şeklinde iki ayrı demete bölünebilmektedir.

Hiyerarşik algoritmaları da graf tabanlı demetleme ile ilişkilidir. Esasında single-link algoritması sonucu üretilen demetler, minimal kapsayan ağacın bağlı (connected) olan alt kümeleridir (Gower ve Ross 1969). Aynı şekilde complete-link algoritması sonucu üretilmiş olan demetler de maximal complete alt graflardır (Gotlieb ve Kumar 1968).

3.4.3. En Yakın Komşu Yöntemi

Demetleme işleminde demet içi yakınlığın yüksek olması gereksinimi, en yakın komşu uzaklığının demetleme işlemi sırasında kullanılabilir olmasını sağlamaktadır. Bu doğrultuda Lu ve Fu 1978'de iteratif bir yöntem geliştirmişlerdir. Bu yönteme göre her etiketlenmemiş yani herhangi bir demete dahil edilmemiş bir örüntü elemanın, kendisine en yakın olan etiketlenmiş bir komşusu bulunur ve bu komşunun etiketi, bu etiketlenmemiş elemana aktarılır yani bu eleman demete dahil edilir. Bu işlemin

gerçeklenmesi için hesap edilen minimum uzaklığın, belirli bir eşik değerinin altında olması gerekmektedir. Demetleme işlemleri, bütün örüntü elemanları bir gruba dahil edilinceye kadar ya da gerçeklenen iteratif adımda herhangi bir demete dahil edilme işleminin olmadığı adıma kadar devam eder.

3.4.4. Bulanık Demetleme

Bulanık demetleme yönteminde her bir veri, belli bir demete belli bir üyelik değeri ile dahil olur. Bulanık demetlemede her bir demet bulanık olarak tanımlanmaktadır. Üyelik değerine belli bir eşik değeri uygulayarak bulanık demetlemeden keskin demetlemeye geçilir. En popüler bulanık demetleme algoritması Fuzzy c-Means (FCM)' dir. Bezdek tarafından 1981' de FCM algoritmasının genelleştirilmiş hali ortaya konmuştur. Şekil 3.11' de Temel Bulanık Demetleme Algoritması tanıtılmaktadır.

---

1. *N nesneyi K demete N x K üyelik matrisi* U *' yu seçerek ata. (* $\mu_{ij} \in [0,1]$*)*
2. U *' yu kullanarak bulanık ölçüt fonksiyonunu bul. Örn. Belli bir demetleme için ağırlıklı toplam hatanın karesi ölçütü. Bir olası bulanık ölçüt fonksiyonu:*

$$E^2(x,U) = \sum_{i=1}^{N}\sum_{k=1}^{K} u_{ij} \left\| x_i - c_k \right\|^2$$

Buradaki $c_k = \sum_{i=1}^{n} u_{ik} x_i$ *, k. bulanık demet merkezidir. Bu ölçüt fonksiyonunu azaltmak için verileri yeniden ata ve* U *' yu yeniden hesapla.*

3. *2.adımı* U *önemli ölçüde değişmeyinceye kadar tekrarla.*

Şekil 3.11 - Temel Bulanık Demetleme Algoritması

3.4.5. Yapay Sinir Ağları İle Demetleme

Yapay sinir ağları kaynağını biyolojik sinir sisteminden ilham almıştır. Yapay sinir ağları demetleme amacı ile son yıllarda yaygın olarak kullanılmaya başlamıştır (Sethi ve Jain 1991). Yapay sinir ağlarının demetleme işlemi için önemli olan birkaç özelliği bulunmaktadır:

- Yapay sinir ağları sayısal vektörler üzerinde çalışmaktadırlar dolayısı ile sadece sayısal verilerden oluşmuş örüntü kümesine ihtiyaç duyarlar.
- Yapay sinir ağları yapısı itibarı ile paralel ve dağıtılmış çalışma mimarileri için uygundur.

- Yapay sinir ağları, bağlantı ağırlıklarını adaptif olarak öğrenme yeteneğine sahiptir (Mao 1996, Oja 1982). Özelleştirilirse yapay sinir ağları örüntü kümesini demetleme işlemi için normalize ederler ve ağırlıkları uygun seçilmesi ile de özellik seçiminde bulunurlar.

Giriş verisini demetlemek için, kazananın hepsini aldığı yarışmalı (competetive) yapay sinir ağları kullanılmaktadır (Jain ve Mao 1996). Bu yöntemde benzer veriler bir grup halinde bir araya getirilirler ve oluşan grupların her biri, bir nöron olarak ifade edilirler. Bu gruplandırma işlemi, veriler arasındaki korelasyon kullanılarak otomatik olarak gerçeklenir. Demetleme için sık kullanılan yapay sinir ağları çeşitleri olarak Kohonen tarafından geliştirilen LVQ (learning vector quantization) ve SOM (self organizing map) (Kohonen 1984) bulunmaktadır.

Bu yapay sinir ağlarının çalışma yapıları genelde benzerdir. Tüm mimariler tek katmanlıdır. Örüntü kümesi girişlerden uygulanır ve çıkışlardan sonuçlar elde edilir. Giriş ile çıkış arasındaki ağırlıklar ise belirli bir sonlandırma kriteri sağlanana dek iteratif olarak değiştirilmektedir (öğrenme).

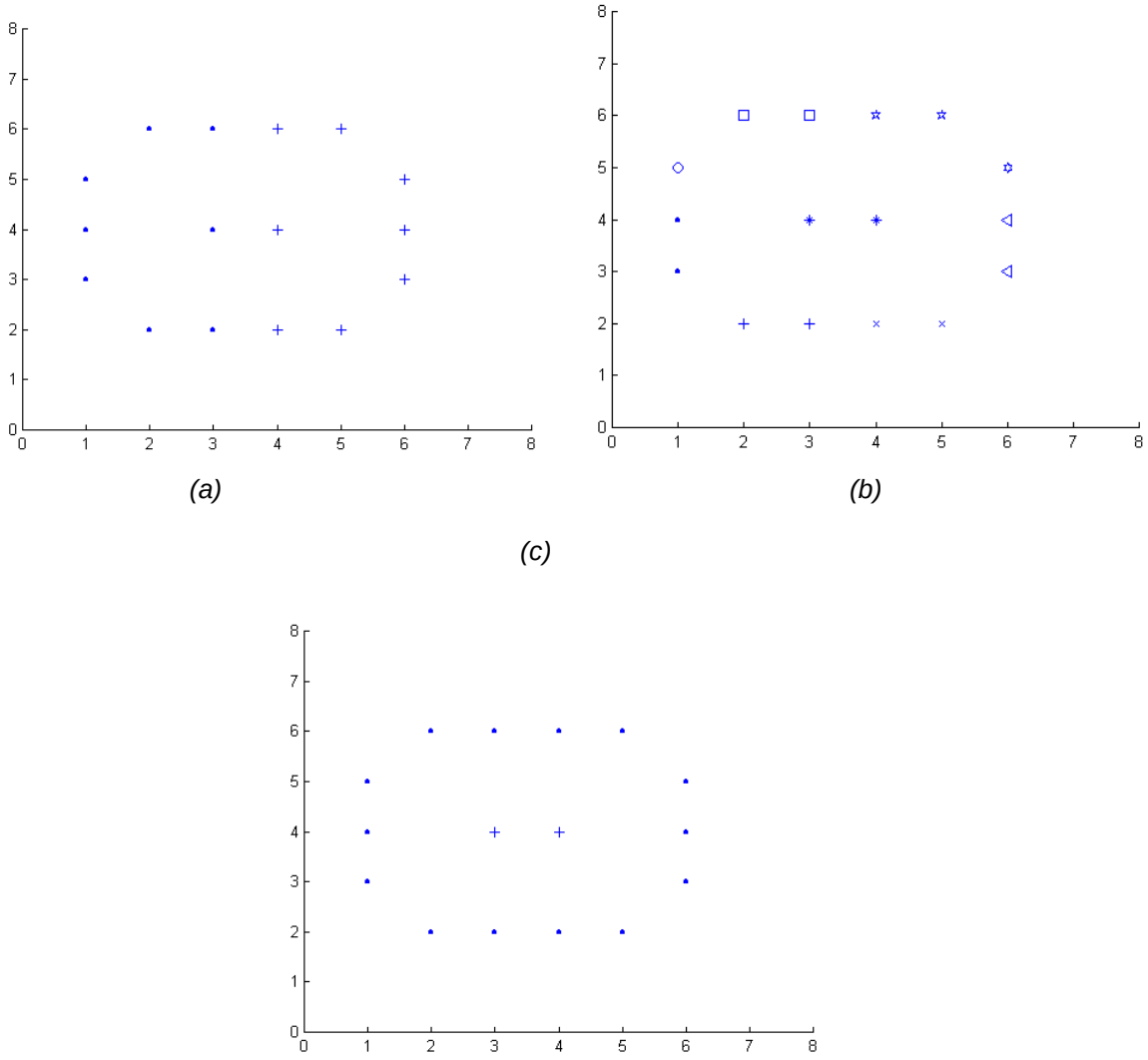
3.5. Demetleme Algoritmalarının Karşılaştırılması

K-Means yöntemi, yürütme zamanı açısından en etkin olan algoritmalardan biridir. K-Means algoritması ve diğer sürümleri büyük veri topluluklarına uygulanabilir. Ayrıca K-Keans algoritmasının yerel olarak optimum sonuca ulaştığı görülmüştür (Jain, 1999). Bu davranış, K-Means' te başlangıç dağılımının belirlenmesi ile birleştirilmesi sonucu etkin sonuçlara ulaşmak mümkündür. Dolayısıyla K-Means' in iyileştirilmiş bir sürümü olan FCM, iyi sonuçlar üretmektedir. Demet-tabanlı doküman sınıflama uygulamasında hiyerarşik algoritmaların kısmi algoritmalara göre daha iyi sonuçlar verdiği gözlenmiştir.

Demetleme algoritmaları için:

1. Tüm demetleme algoritmaları verilen bir veri topluluğu üzerinde demetler (olsa da olmasa da) bulacaktır. Dolayısıyla verinin demetlenme eğiliminin olup olmadığı demetleme algoritması uygulanmadan önce test edilmelidir.
2. En iyi demetleme algoritması yoktur. Dolayısıyla belli bir veri grubu için bir çok demetleme algoritması uygulanmalıdır.

Hiyerarşik algoritmalar, kısmi algoritmalardan daha esnektir. Örneğin Single-link algoritması yoğun olmayan, iyi bir şekilde ayrılmış, zincir-yapısında, ve aynı merkeze sahip bölütler üzerinde iyi çalışır. K-Means algoritması yoğun veriler üzerinde iyi çalışır. Bunun yanında kısmi algoritmaların zaman ve alan (space) karmaşıklıkları daha düşüktür. Her iki grubun da iyi özelliklerini alan hibrit algoritmalar kullanmak yararlı olabilir.

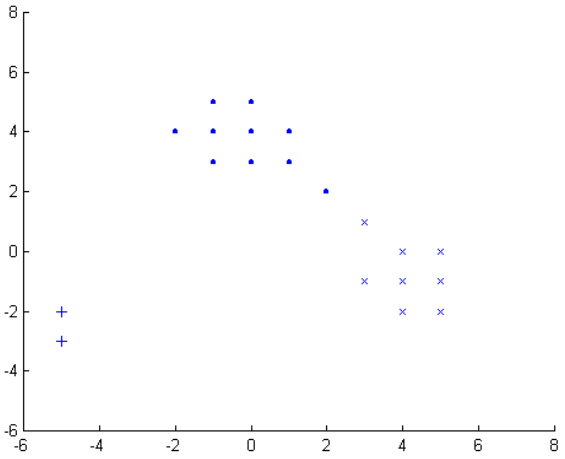


(a) *(b)*

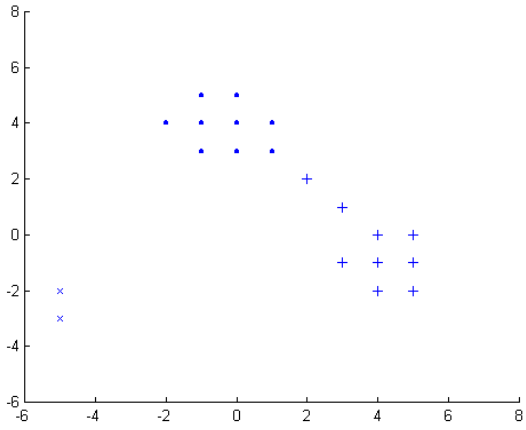
(c)

Şekil 3.12 - Bir Örüntü Kümesi İçin Single-link ve Complete-link Algoritmaları Sonuçları

Algoritmalar, farklı veri yapıları için farklı demet oluşturmaktadır. Bir veri topluluğu için FCM, Complete-link ve Single-link algoritmalarının yürütülmesi sonucu oluşan bölütler Şekil 3.12 (a), Şekil 3.12 (b) ve Şekil 3.12 (c)' de görülmektedir. Ortak merkezli iki demetli olarak algılanabilecek olan veri topluluğunun bu üç algoritma için ürettikleri farklı sonuçları uygulamaya göre farklı değerlendirilebilir.

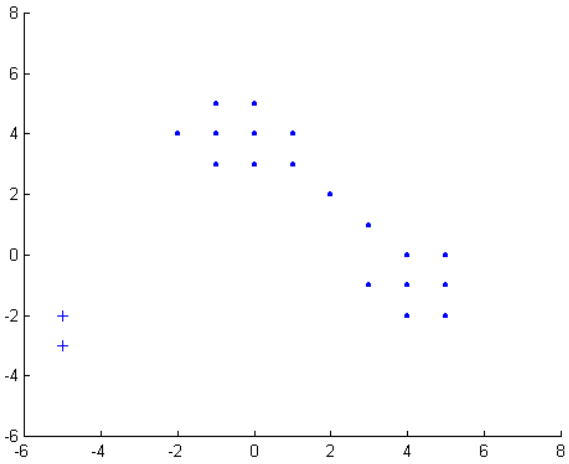


(a)



(b)

(c)



Şekil 3.13 - Bir Örüntü Kümesi İçin FCM, Single-link ve Complete-link Algoritmaları Sonuçları

Bir başka veri topluluğu için oluşturulan sonuç bölütleri Şekil 3.12 (a), Şekil 3.12 (b) ve Şekil 3.12 (c)' de verilmiştir. FCM ve Complete-link algoritmaları zincir yapısındaki veri grubu için iki demet üretmekte iken Single-link algoritması zincirleme etkisine uğrayarak zincir yapısındaki veri grubundaki verileri tek bir demete dahil etmiş yani veri üzerinde bir demetleme tanımlamamıştır. Bu da Single-link algoritmasının verinin üzerinde gürültü tanımlanması durumunda yanlış sonuçlar

üretebileceğini göstermektedir. Sonuçta oluşan durumlar uygulama açısından değerlendirilmelidir.

3.6. Büyük Ölçekleri Verileri Demetleme

Büyük hacimli örüntü kümelerinin demetlenmesini gerektiren birçok uygulama bulunmaktadır. Büyük ölçekli olma kavramı ise zamana göre görecelidir. 1960'lı yıllarda büyük ölçekli veriler demek binlerle ölçülen sayıda örüntü elemanına sahip olmak anlamına geliyordu [Ross 1968]. Günümüzde ise büyük özellik uzayı boyutlarına sahip milyonlarca örüntü elemanının demetlenmesini gerektiren uygulamalar bulunmaktadır. Örneğin 500*500 beneklik bir imgeyi bölmelemek için 250.000 beneğin demetlenmesi gerekmemektedir. Döküman içeriği araştırma ve bilgilerin filtrelenmesi gibi işlemlerde ise özellik uzayı boyutu 100'den büyük olan milyonlarca örüntü kümesi elemanının demetlenmesi gerekmektedir.

Varolan demetleme algoritmanın bir çoğu sadece belirli sayıda örüntü kümesi elemanının olduğu durumlarda performanslı olmaktadır. Ancak K-Means algoritması ve bunun Yapay Sinir Ağları eşleniği olan Kohonen Ağı, geniş ölçekli verileri demetlemede en yaygın kullanılan yöntemlerdir. K-Means algoritmasının geniş ölçekli verileri demetlemede üstün olduğu noktalar aşağıdaki maddelerle özetlenebilir:

1. K-Means algoritmasının karmaşıklığı *O(nkl)*'dir. Burada *n* örüntü kümesindeki eleman sayısı, *k* sonuç olarak üretilecek demet adedi ve *l* de algoritma bir sonuca yakınsayana kadar gerçekleşen iterasyon adedidir. Genel olarak *k* ve *l* sayıları sabittir dolayısı ile K-Means algoritması karmaşıklığı doğrusal olarak değerlendirilir (Day 1992).
2. Algoritmanın bellek karmaşıklığı ise *O(k)*'dır. Ayrıca verilerin bulunduğu matrislerin saklanması için ise ayrıca bir yere ihtiyaç vardır. Verilerin bulunduğu matrisleri ikincil belleklerde (disk, teyp vb.) saklamak ve ihtiyaç duyuldukları zaman buradan erişmek de mümkündür. Ancak bu tarz bir çalışma yöntemi çok aşırı miktarlarda disk erişim süresi gerektirecektir ve ardışıl işlemler için gerekli olan süre aşırı olarak artacaktır.
3. K-Means algoritmasının ürettiği sonuçlar, verilerin sırasına bağlı değildir. Belirli bir başlangıç seçimi için örüntü kümesinde yer alan verilerin sırası ne olursa olsun her çalıştırılışında aynı demetleri oluşturur.

Ancak bütün bu üstün özelliklerine rağmen K-Means algoritmasının ürettiği sonuçlar, başlangıç noktalarının seçimi ile doğrudan ilgilidir.

Hiyerarşik algoritmalar daha kullanışlı olmalarına rağmen büyük ölçekli verileri demetlemede bir takım zayıf yönleri bulunmaktadır:

1. Hiyerarşik algoritmaların zaman karmaşıklıkları $O(n^2 logn)$ 'dir (Kurita 1991). Verilerin Kapsayan Ağacını (Minimum Spanning Tree) kullanarak single-link demetler elde edilebilir. Ancak 2 boyutlu özellik uzayına sahip bir örüntü kümesi için MST'nin elde edilmesi için karmaşıklık $O(nlog^2 n)$'dir (Choudhury ve Murty 1990).
2. Hiyerarşik algoritmaların bellek karmaşıklıkları ise $O(n^2)$'dir. Bunun nedeni ise n x n boyutunda bir benzerlik matrisinin saklanmasının gerekliliğidir. 100 x 100 beneklik bir imgeyi hiyerarşik yöntemlerle demetlere ayırmak için gereken bellek miktarı 50 megabyte boyutundadır. Ancak benzerlik matrisini bellekte saklamak yerine, benzerliğin gerektiği her yerde benzerliğin tekrar hesaplanması seçeneği de bulunmaktadır. Ancak bu durumda da algoritmanın zaman karmaşıklığı çok artacaktır (Anderberg 1973).

Büyük ölçekli veri kümelerini demetleme sorunun olası bir çözümü, Ross tarafından (Ross 1968) önerilen hibrit bir yöntemi kullanmaktır. Bu yöntemde işlemin ilk başında aynı K-Means algoritmasında olduğu gibi veri kümesindeki bir kısım veriler başlangıç noktaları olarak belirlenir ve geri kalan tüm kayıtlar bu seçilen kayıtlara ya da demetlere atanırlar. Daha sonra oluşan her demet içerisinde ayrı ayrı bir kapsayan ağaç (MST) bulunur. Bulunan bu kapsayan ağaçlar bir sonraki adımda, tüm örüntü kümesi için yaklaşık bir kapsayan ağaç bulmak için birleştirilirler.

Bentley ve Friedman (1978) ise bahsedilen bu yaklaşık kapsayan ağacı $O(nlogn)$ karmaşıklığında bulabilen bir algoritma geliştirmişlerdir. Bruynooghe ise hızlı en yakın komşuluk (fast nearest neighbour) algoritması ile demetleme işlemini 25 kat arttırmıştır. Eddy (1994) ise single-link yöntemi ile belirlenen uzaklık yöntemine göre büyük ölçekli verileri demetlemek üzerine çalışmalar yapmıştır. Bu çalışmalarda gene yaklaşık kapsayan ağaç yöntemi kullanılmıştır. Sonuçta 40.000 kayıt üzerinde gerçeklenen algoritma birkaç dakika içerisinde yaklaşık kapsayan ağacı bulabilmektedir.

Yukarıda bahsedilen geniş ölçekli örüntü kümeleri üzerinde işlem yapan algoritmaların hepsi, verilerin bulunduğu veri matrisinin tamamının ana bellekte tutulabildiği durumlarda çalışabilmektedirler. Oysa gerçek hayatta verilerin tamamının ana bellekte tutulamayacağı kadar büyük olduğu uygulamalar bulunmaktadır. Örneğin özellik uzayı boyutu 1000 olan bir milyon kaydın demetlenmesi için 1000 mega byte gibi bir bellek alanı gerekmektedir. Bu bellek

miktarı günümüzün bilgisayarlarında dahi kolayca gerçeklenememektedir. Bu sorunu çözmek iki temel yöntem bulunmaktadır:

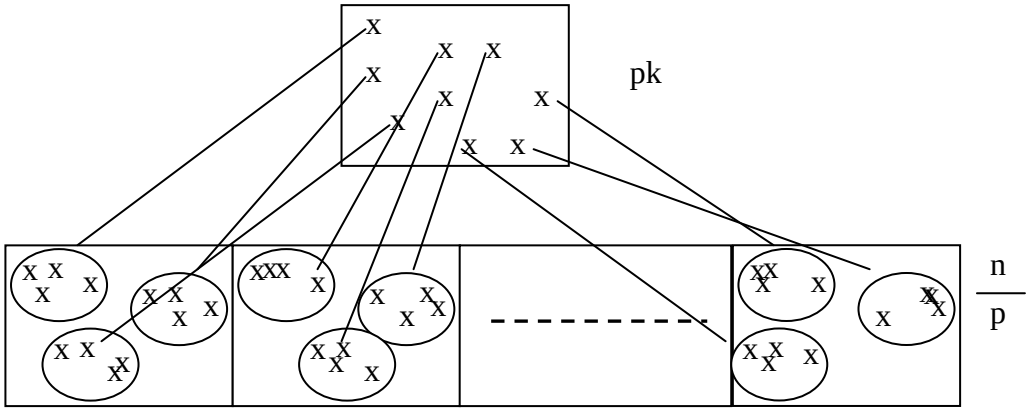
1. Tüm örüntü kümesi ikincil bir bellekte saklanabilir. Bu ikincil bellekten veriler parça parça getirilir ve her parça kendi içerisinde demetlenir. Daha sonra bu parçalardan elde edilmiş tüm demetler birleştirilerek tüm veri üzerinde geçerli bir demetleme oluşturulur. Bu yaklaşım böl ve yönet (divide and conquer) yöntemi olarak adlandırılmaktadır.
2. Diğer bir çözüm olasılığı ise artımlı demetleme algoritması kullanmaktır. Bu yöntemde verilerin bulunduğu matrisin tamamı ikincil bellekte örneğin diskte tutulur ve her seferinde demetleme işlemi için ihtiyaç duyulan veri, ana belleğe getirilir. Ana bellekte sadece o andaki demetlerin prototipleri bulunur.

3.6.1. Böl ve Yönet

Bu yöntemde demetlenecek verilerin bulunduğu n x d boyutundaki matris ikincil saklama birimlerinde (örneğin diskte) tutulmaktadır. Burada n, demetlenecek veri adedini, d ise özellik uzayı boyutunu ifade etmektedir. Verilerin tamamını her biri x adet eleman içeren p adet bloğa bölebiliriz. Buradaki p sayısı kullanılan demetleme algoritmasında optimum sonucu verecek şekilde seçilmelidir (Murty ve Krishna 1980). Bu durumda her bir blokta yer alan veri sayısı

$$x = \frac{n}{p} \tag{3.6}$$

olacaktır. Böl ve yönet yönteminde her blok sırasıyla ana belleğe getirilir ve ilgili algoritma bu blok üzerinde çalıştırılarak k adet demet oluşturulur. Algoritmanın her çalıştırılışında üretilen sonuçlar birbirlerinden ayrı olacak şekilde ana bellek içerisinde saklanırlar. Tüm p adet blok işlendikten sonra elimizde pk adet demet prototipi kalacaktır. Daha sonra bu pk adet prototip üzerinde tekrar demetleme algoritması çalıştırılarak k adet demet elde edilir. Bu işlem Şekil 3.13 de görüldüğü gibi iki seviyeli bir işlemdir.



Şekil 3.14 – Böl ve Yönet Tekniği İle Büyük Ölçekli Verileri Demetleme

Böl ve yönet algoritması herhangi sayıda seviyede gerçeklenebilir. Demetlenecek örüntü kümesinin eleman sayısının çok fazla, buna mukabil kullanılacak ana belleğin çok küçük olması durumunda seviye sayısının arttırılması gerekmektedir (Murty ve Krishna 1980).

Eğer demetleme algoritması olarak single-link algoritması kullanılırsa ve sonuçta 5 adet demet elde edilmesi istenirse seçilen p değerine göre hesaplamalarda sağlanılacak kazanç Tablo 3.1 de verilmiştir:

Tablo 3.1 - SLA ve İki Seviyeli Böl ve Yönet Algoritmalarında Uzaklık Hesapları Adedi

N	SLA	P	İki Seviyeli Böl ve Yönet
100	4.950	5	1200
500	124.750	20	10.750
1000	499.500	40	31.500
10.000	49.995.000	100	1.013.750

Bu yöntemin olumlu taraflarından bir tanesi de çok büyük ölçekli verileri demetlemeyi gerçeklenebilir boyutlara indirgemesidir. Ancak bu yöntemin kullanılabilir sonuçlar üretebilmesi de her bir blok içerisinde yer alan verilerin homojen olarak dağılması koşuluna bağlıdır. Bu koşul ise imgeleri temsil eden veriler tarafından kolaylıkla sağlanabilmektedir (Jain ve diğ. 1999).

Stahl (1986) tarafından geliştirilen bir başka yöntemde de büyük ölçekli verilerin demetlenmesi için iki seviyeli bir strateji gerçekleştirilmiştir. İlk seviyede veri kümesi çok fazla sayıda demete ayrılmıştır. Bu seviyede kullanılan demetleme algoritmasına ise lider algoritma denilmektedir. Daha sonra lider algoritma tarafından üretilen bu çok sayıdaki demetin prototipleri, tekrar ikinci seviye

demetlemesine maruz kalırlar. Bu şekilde sonuç olarak tüm veri üzerinde geçerli olan bir demetleme elde edilmiş olur.

3.6.2. Artımlı Demetleme

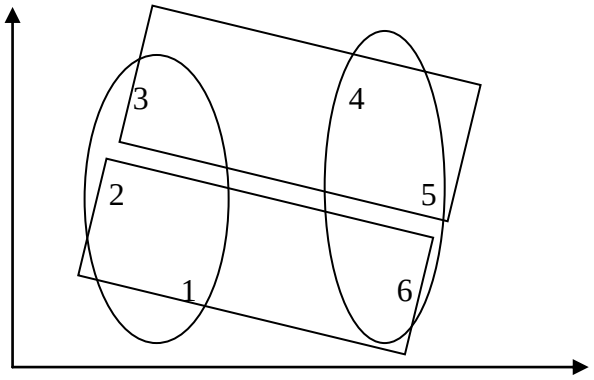
Artımlı demetleme algoritmasının en önemli avantajı algoritmanın yürütülmesi esnasında ihtiyaç duydukları bellek alanının çok küçük olmasıdır. Tipik olarak iteratif olmadıklarından dolayı zaman karmaşıklıkları da azdır. Birkaç tane artımlı demetleme yöntemi bulunmaktadır:

1. Lider demetleme algoritması (Hartigan 1975), zaman karmaşıklığının *O(nk)* olmasından dolayı kullanılabilecek en basit yöntemidir. Bu yöntem Yapay Sinir Ağlarında ART ağı (Carpenter ve Grossberg 1990) olarak gerçeklenebildiği için popülerlik kazanmıştır. Bu yöntemin gerçeklenmesi durumunda bellek karmaşıklığı *O(k)* olmaktadır.
2. Kısa kapsayan yol (SSP) algoritması (Slagle 1975) verilerin organize edilmesi işlemi için geliştirilmiştir. Kayıtların otomatik olarak yönetilmesini işleminde SSP yöntemi başarıyla uygulanmıştır (Lee ve diğ. 1978). Bu uygulamada SSP algoritması 2000 elemanlı, 18 boyutlu özellik uzayı olan bir örüntü kümesini demetlemiştir. Oluşan demetler ise bazı kayıtlarda eksik olan bazı özelliklerin ne olabileceği konusunda tahminde bulunmak için ya da bazı yanlış özelliklerin tespitinde kullanılmıştır.
3. Fisher tarafından geliştirilen cobweb (Fisher 1987) sistemi ise artımlı kavramsal demetleme algoritmasıdır. Bu sistem de bir takım mühendislik uygulamalarında başarılı bir şekilde gerçeklenmiştir (Fisher 1993).
4. Bilgilerin dinamik olarak işlenmesi için Can (1993), artımlı bir demetleme yöntemi kullanmıştır. Bu yöntemin amacı, daha önceden demetlenmiş bir veri tabanı üzerinde sonradan gerçekleşen yeni kayıt ekleme, varolan kayıtları düzeltme ve silme gibi dinamik olarak gerçekleşen işlemlerin halihazırdaki demetlere yansıtılmasını sağlamaktır. Bu algoritma da bilgisayar bilimleri ve elektrik mühendisliği konularında 12.684 döküman içeren INSPEC veri tabanı üzerinde artımlı bir demetleme gerçekleştirmek üzere kullanılmıştır.

Demetleme algoritmalarının, verilerin tutulduğu sıradan bağımsız olarak çalışabilmesi önemli bir özelliktir. Bir algoritmanın *sıradan-bağımsız* bir algoritma olabilmesi için, veriler hangi sırada işleme girerse girsin aynı sonuçları üretmesi gerekmektedir. Tersi durumda ise algoritma, *sıraya-bağımlı* bir algoritma olarak

adlandırılmaktadır. Yukarıda bahsedilen artımlı demetleme algoritmalarının çoğu *sıraya-bağımlı* algoritmalardır.

Örnek olarak aşağıdaki şekilde yer alan verilerin demetlenmesi gösterilebilir:



Şekil 3.15 – Demetlemede Sıraya Bağımlılık

Şekil 3.15'de 1'den 6'ya kadar numaralandırılmış, her biri iki özelliğe sahip veriler bulunmaktadır. Eğer bu veriler sıraya-bağımlı bir algoritmaya 2,1,3,5,4,6 sırasında verilirse oluşan demetler, elipsler ile gösterilmiştir. Aynı verilerin aynı algoritmaya 1,2,6,4,5,3 sırasında verilirse sonuçta üretilen demetler dikdörtgenlerle belirtilmiştir. SSP algoritması, cobweb ve Can'in yöntemlerinin hepsi *sıraya-bağımlı* demetleme algoritmalarıdır.

4. UYGULAMA

4.1. Amaç

Garanti Teknoloji A.Ş. ile ortak yürütülen bu tez çalışmasında amaç, halihazırda IBM Intelligent Miner ile gerçeklenen segmentasyon işlemini yapabilecek bir yazılım geliştirmektir. Bu yazılım geliştirilirken, her çeşit algoritmanın gerçeklenebileceği bir çerçeve mimari oluşturulması hedeflenmiştir.

4.2. Tanım

Bölüm 4.1'de amacı anlatılan yazılımı geliştirmek üzere bir yazılım mimarisi oluşturulmuştur. Bu yapının modülerlik ve esneklik açısından sahip olması gereken bir çok özellik bulunmaktadır. Bu özellikler aşağıda sıralanmışlardır.

- İşletim Sisteminden Maksimum Derecede Bağımsız Olarak Çalışma
- İstemci / Sunucu Mimarisi Desteği
- Modülerlik ve NYP (Nesne Yönelimli Programlama)
- Veri Tabanı Türünden Bağımsız
- Paralel Çalışmaya Olanak Sağlama
- Kabul Edilebilir Süreler İçerisinde Çalışma
- Minimum Miktarda Sistem Kaynağının Kullanılması

Yukarıda anılan özelliklerden işletim sisteminden bağımsız çalışabilme özelliğinin sağlanabilmesi için geliştirme ortamı seçiminin dikkatli yapılması gerekmektedir. Taşınabilir ya da değişik işletim sistemleri ve değişik platformlar üzerinde çalışabilecek bir yazılımlar üretmek amaçlanmıştır. Bu doğrultuda olası geliştirme ortamı seçeneği 2 adet olmuştur.

Seçeneklerden ilki olan Java, hem popülerliği hem de getirdiği yeni özellikler ile ilk bakışta cazip görünmesine rağmen Java ortamının yüksek ölçüde taşınabilirlik desteğinden doğan yavaşlık sorunu bulunmaktadır.

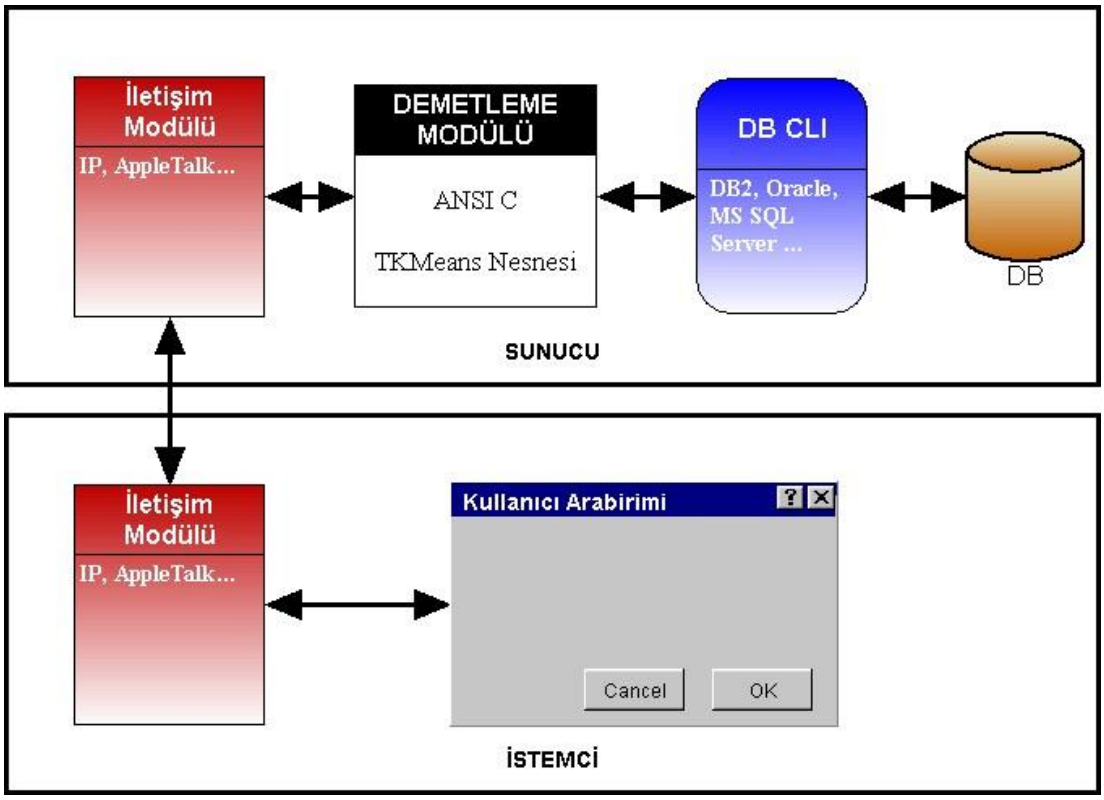
Geliştirme ortamı olarak seçilebilecek ortamlardan ikincisi ise C/C++ programlama dilidir. Bu dilin yaygın kullanımı, doğası gereği makine diline en yakın üst düzeyli dil

olması ve geniş desteğe sahip olması C dilinin kullanılması için yeterli nedenler olarak görülmüştür. Hemen hemen her platform için derleyicisi bulunan C/C++ programlama dili ile son derece hızlı çalışan yazılımlar geliştirmek mümkündür.

Ancak her platformda ve işletim sisteminde çalışacak bir C kodu yazmak için ANSI C kullanmak gereklidir. Bu kapsamda geliştirilen yazılım da ANSI C kullanılarak gerçeklenmiştir. Ancak kullanıcı arayüzlerinin ve programın algoritma ile ilgili olmayan diğer kısımlarının ANSI C ile kodlanması kullanıcı arayüzünün hoş olmamasına neden olmamaktadır. Bu nedenle yazılımın sadece veri madenciliği algoritmalarını gerçekleyen (bu yazılım kapsamında sadece K-Means demetleme algoritması) çekirdek yazılım ANSI C ile yazılmış ancak kullanıcı arayüzü ve iletişim gibi diğer kısımlar başka ortamlarda geliştirilmesine olanak sağlayacak şekilde modüler tasarlanmıştır.

İstemci / Sunucu mimarisini desteklemek üzere, ana bilgisayar ya da sunucu üzerinde sadece veri madenciliği işlemlerini gerçekleyen çekirdek program parçalarının çalıştırılması öngörülmüştür. Bu bağlamda kullanıcı arayüzü istemci bilgisayarlar üzerinde çalışacak, kullanıcılar çekirdek programın ihtiyaç duyduğu çeşitli parametreleri belirleyip ağ üzerinden ana bilgisayarda bulunan çekirdek yazılıma ileteceklerdir. Bu çalışma, Şekil 4.1.'de şematik olarak gösterilmiştir.

Nesne yönelimli programlama ilkesi uyarınca tüm program içerisinde çekirdek kısmı oluşturan yani demetleme işlemini gerçekleyen program parçası ayrı birer nesne olarak kodlanmış ve analiz için ihtiyaç duyulan tüm parametrelerin ve sonuçların metotlar üzerinden işlenmesi sağlanmıştır. Bu şekilde nesne yönelimli programlama desteklenmiş ve ileride gerçeklenecek diğer veri madenciliği algoritmaları için alt nesneler oluşturulmuştur. Örneğin verileri bellekte optimum şekilde tutan veri yapısı bulunduğu için bu veri yapısı üzerinde kolaylıkla başka algoritmaların da gerçeklenmesi sağlanabilir.



Şekil 4.1 – Uygulama Modülleri

Veri tabanı türünden bağımsız olması ve modülerliği sağlama amacı ile çekirdek algoritma, veri tabanı arayüzü, iletişim arabirimi ve kullanıcı arayüzü ayrı modüller olarak düşünülmüştür. Şekil 4.1.'de modüller arasındaki veri akışı gösterilmiştir.

Şekil 4.1.'de yer alan veri tabanı arayüzü (DB CLI – Database Call Level Interface) veri ambarından ya da veri tabanından demetlenmesi istenilen verilerin çekilip algoritmanın kullanımına sunmak üzere belirli bir veri yapısına taşımaktadır. Bu arayüz genelde her veri tabanı ile birlikte verilen ve bu veri tabanına uygulama programları içerisinden doğrudan erişmek ve işlem yaptırmak için kullanılan bir takım fonksiyonların bulunduğu bir kütüphaneyi kullanmaktadır. Bu arayüzde yer alan fonksiyonlar kullanılarak, derlemeden önce veri tabanı ile gelen ön-derleyici (pre-compiler) ile derlenen yazılımlar daha sonra gerçek derleyiciler ile derlendiklerinde ortaya çıkan yazılım, ilgili veri tabanına erişebilir hale gelmektedir.

İletişim modüllerinin ayrı olarak tasarlanmasının hedefi farklı protokolleri destekleyen modüllerin oluşturulmasına olanak vermektedir. Bu şekilde, geliştirilecek yazılımın değişik platformlar üzerinde çalışması istendiğinde, ilgili platform üzerinde çalışabilecek bir protokol kullanan bir iletişim modülü düzenlenip diğer kısımlarda bir değişik olmadan eski iletişim modülünün yerine geçebilmektedir.

Geliştirilecek yazılımın sağlaması gereken diğer bir özellik de paralel çalışmaya imkan sağlayacak şekilde gerçeklenmesidir. Bunu sağlamak üzere veriler aşağıdaki bölümlerde anlatılacağı üzere bloklar halinde bellek içerisinde tutulmuştur. Böylelikle bir proses içerisinde çalışan bir demetleme algoritması belirli bir blok içerisindeki verilerin demetlenmesi üzerinde çalışırken bir başka proses de bir başka blok içerisindeki verilerin demetlenmesi üzerinde çalışabilir. Bunu sağlamak üzere her blok ile ilgili tanımlayıcılar içerisine ilgili bloğun o anda bir proses tarafından işlenip işlenmediğinin bilgisi eklenmelidir. Bunun haricinde algoritmayı içeren nesnede yapılacak bir iki değişiklikten sonra yazılım, paralel çalışma için uygun hale getirilebilir.

4.3. İyileştirmeler

Çok geniş ölçekli verilerin demetlenmesinde karşılaşılan ana sorun, verilerin nasıl bir yapıda saklanacağıdır. Zaman karmaşıklığının azaltılması için verilerin ana bellek içerisinde bulunması gereklidir. Ancak bu durumda da algoritmanın bellek karmaşıklığı artmaktadır. Demetlenmesi istenilen verilerin çok büyük ölçekte olması durumunda da uygulamada bazı engellerle karşılaşılmaktadır. Örneğin verilerin tamamının belleğe sığmasının mümkün olmadığı uygulamalar bulunmaktadır. Bu tarz uygulamalarda genellikle demetlenmesi istenilen veriler milyonlarla belirtilir. Ayrıca bu örüntü kümesinin özellik uzayının boyutları da 50, 100 hatta çok daha fazla da olabilir. Bu ise sadece demetlenilmesi istenilen kayıtların verilerinin giga-sekizliler düzeyinde yer gereksinimi olduğu anlamına gelmektedir. Oysa günümüzün gelişmiş bilgisayarlarında dahi bu boyutta bellekler bulunmamaktadır. Üstelik tüm bunlara ek olarak esnek bir veri yapısının kullanılmasının gerekliliğinden dolayı bir takım işaretçiler için de ana bellekte yer ayrılmalıdır. Demetleme algoritmasında kullanılacak esnek bir veri yapısı kurmak da çok önemlidir. Çünkü her demetleme işlemi için o soruna özel matrisler ya da n x m boyutunda diziler oluşturulması çok kullanışlı bir yöntem değildir. Üstelik bu dizilerin boyutlarının tanımının, kaynak kod içerisinde yapılmasının getirdiği bir başka zorluk vardır.

Özet olarak veri madenciliğinde çok büyük ölçekteki verileri demetlemek için kullanılacak algoritmanın verileri verimli ve esnek bir şekilde ana bellekte saklaması gerekmektedir. Ana belleğe sığmayacak kadar büyük örüntü kümelerinde ise mümkün olduğu kadar çok miktarda veriyi ana bellek içerisinde sakladıktan sonra geri kalan verileri, zamanı geldiğinde ana belleğe geri yüklenmek üzere ikincil saklama ortamlarına kaldıran bir mekanizma gerekmektedir.

Ayrıca günümüzün veri ambarlarında bulunan verilerin tiplerinin değişiklik göstermesi de demetleme algoritmalarının değişikliğe uğramasını zorunlu hale getirmektedir. Veri madenciliğinden çok uzun yıllar önce geliştirilmiş olan demetleme teknikleri (K-Means demetleme algoritması, hiyerarşik demetleme algoritmaları vb.) sadece sayısal veriler üzerinde çalışabilmektedir. Oysa günümüzde veri ambarı içerisinde her türlü veri bulunmaktadır. Bunlara tamsayılar, ondalıklı sayılar dahil olduğu gibi aynı zamanda kategorik ve karakter katarı (string) tipinde veriler de bulunmaktadır. Sayısal verilerin çok büyük bir aralıkta değerler alabildiği durumlarda bu aralık belirli bir sayıda (çoğunlukla 3 ila 10 arası) kategorilere bölünmekte ve bu kategorilerin demetlenmesi yapılmaktadır. Ayrıca doğası itibariyle kategorik olan bazı alanların da özellik uzayına dahil edilmesi gerekmektedir. Tüm bunlara ek olarak karakter katarı tipinde bir özelliklerin de demetlemeye dahil edilebilmesi zorunludur.

Sonuç olarak geliştirilen demetleme tekniklerinin değişik tiplere sahip özelliklerin de işlenebilmesi zorunluluğunu getirmektedir.

4.3.1. Veri Tipi Çeşitliliği Desteği

Veri ambarlarının anlatıldığı 1. bölümden de görüleceği üzere veri ambarı içerisinde yer alan kayıtların alanları çok değişken olabilmektedir. Veri ambarına gelen kayıtlar gerçek dünyadaki canlı sistemdeki kayıtlardan oluştuğu için gerçek dünyada yer alan her bilgi bir alan olabilmektedir. Örneğin müşterilerin yaşı bir bilgi olabileceği gibi adı da bir alan olabilir, eğitim seviyesi de (ilkokul, lise üniversite gibi) bir alan olabilir.

Veri ambarı üzerinde demetleme işlemi gerçekleştirilmek istenildiğinde, bu kayıtların sahip olduğu alanlardan bir grup seçilerek bir özellik uzayı oluşturulur. Bu özellik uzayına dahil edilen değerler kullanılarak örüntü kümesi demetlenir. Demetleme algoritmasının kullanmak zorunda olduğu veri tipleri aşağıda listelenmiştir.

1. Sayısal Alanlar
2. Kategorik Alanlar
3. Karakter Dizisi Alanlar

4.3.1.1. Sayısal Veriler

Doğrusal değişkenler olarak da bilinen sayısal alanlar en sık rastlanılan veri tiplerinden birisidir. Bu veri tipinde alanlar belirli bir aralıkta değerler alırlar. Gerçek hayattaki karşılıkları sıcaklık, ağırlık, yükseklik, konum gibi bilgilerdir. Ölçü birimi

seçiminden bağımsız demetler elde edebilmek için verilerin özellikleri standart hale getirilir. Bunları gerçeklemek için birkaç adım bulunur :

- Ortalama mutlak sapma hesaplanır:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + ... + |x_{nf} - m_f|) \quad (4.1a)$$

$$m_f = \frac{1}{n}(x_{1f} + x_{2f} + ... + x_{nf}). \quad (4.1b)$$

- Daha sonra mutlak sapma kullanılarak z-skor değeri hesaplanır

$$z_{if} = \frac{x_{if} - m_f}{s_f} \quad (4.2)$$

- Nesneler arasındaki benzerlik genelde her nesne çifti arasındaki uzaklık hesaplanarak bulunur. En çok kullanılan yöntemlerden biri Euclidean mesafesidir:

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2} \quad (4.3)$$

- Kullanılan bir diğer yöntem Manhattan mesafesidir:

$$d(i, j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + ... + |x_{i_p} - x_{j_p}| \quad (4.4)$$

- Euclidean ve Manhattan mesafeleri genelleştirilerek Minkowski mesafesi bulunur:

$$d(i, j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + ... + |x_{i_p} - x_{j_p}|^q)} \quad (4.5)$$

q: pozitif tam sayı

- Uzaklıklar ağırlıklı olarak da hesaplanabilir:

$$d(i, j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + ... + w_p|x_{ip} - x_{jp}|^2} \quad (4.6)$$

4.3.1.2. Kategorik Veriler

Kategorik alanlar, varolan seçeneklerden bir tanesinin ya da birkaç tanesinin seçili olduğu yapılardır. Örneğin Eğitim Durumu ile ilgili bir veri tabanı kaydında aşağıdaki olası kategoriler bulunmaktadır:

İlkokul, Ortaokul, Lise, Üniversite

İlgili müşterinin eğitim durumu bunlardan birisi olmak zorundadır. Bu tarz alanlar doğrusal yani sayısal alanlardan farklıdırlar çünkü kategorik değere sahip iki kayıt arasındaki uzaklık (ya da benzerlik) hesabı yapılırken az önce bahsedilen Euclid ya da en genel haliyle Minkowski mesafesi yöntemleri kullanılamaz.

Kategorik veri tipleri iki şekilde gruplandırılabilir:

İkili Değişkenler

Herhangi bir kaydın bir özellik yalnızca 1 ya da 0 alabiliyorsa bu özelliklere ikili değişkenler adı verilmektedir. Bu şekilde 1 ve 0 şeklinde iki değer alabilen özelliklere sahip i. ve j. kayıtlar arasındaki uzaklık (benzerlik) aşağıdaki tablo yardımı ile hesaplanabilir:

Tablo 4.1 - Örnek kategorik veriler

Nesne i \ Nesne j	0	1	Toplam
0	a	b	a+b
1	C	d	c+d
Toplam	a+c	b+d	p

Tabloda yer alan *a* sayısı i. kaydın k. özelliği ile j. kaydın k. özelliğinin sıfır olduğu durumların sayısını verir. Aynı şekilde *b* sayısı da i. kaydın k. özelliğinin 1, j. kaydın k. özelliğinin 0 olduğu durumların adedidir. Bu şekilde yukarıdaki tablo oluşturulduktan sonra iki kayıt arasındaki uzaklık şu şekilde belirlenebilir:

- Yalın uyum katsayısı (ikili değişkenin simetrik olduğu durumlarda sabit)

$$d(i,j) = \frac{b+c}{a+b+c+d} \tag{4.7}$$

- Jaccard katsayısı (İkili değişkenin asimetrik olduğu durumlar) :

$$d(i,j) = \frac{b+c}{a+b+c} \tag{4.8}$$

Örneğin aşağıda Tablo 4.2.'de verilen kayıtlar arasındaki benzerlikleri inceleyelim:

Tablo 4.2 - Örnek Kategorik Alanlar İçeren Kayıtlar

Ad	Cinsiyet	Ateş	Öksürük	Test-1	Test-2	Test-3	Test-4
Ali	Erkek	E	H	P	N	N	N
Ayşe	Bayan	E	H	P	N	P	N
Mehmet	Erkek	E	E	N	N	N	N
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Bu tabloda hastalar ve hastalıkların teşhisi ile ilgili bir takım bilgiler bulunmaktadır. Bu tabloda yer alan kayıtların cinsiyet özelliği simetrik bir özelliktir. Diğer özellikler ise asimetrik yapıdadır. Bu tabloda yer alan bilgilerden E (Evet) ve P (Pozitif) 1'e, H(Hayır) ve N (Negatif) değerleri de 0'a eşitlenirse Jaccard katsayısı şu şekilde hesap edilebilir:

$$d(Ali, Ayse) = \frac{0+1}{2+0+1} = 0{,}33 \tag{4.9a}$$

$$d(Ali, Mehmet) = \frac{1+1}{1+1+1} = 0{,}67 \tag{4.9b}$$

$$d(Mehmet, Ayse) = \frac{1+2}{1+1+2} = 0{,}75 \tag{4.9c}$$

Nominal Değişkenler

İkili değişkenlerin genelleştirilmiş hali olan nominal değişkenler ikiden fazla değer alabilmektedir. Örneğin renk bilgisi gibi (kırmızı yeşil mavi beyaz gibi). Bu şekilde nominal değişken tipinde özellikler içeren iki kaydın benzerliğinin hesap edilmesinde 2 yöntem kullanılabilir:

Yalın Uyum

Bu yöntemde iki değişken arasında uzaklık şu şekilde hesap edilir:

$$d(i, j) = \frac{p - m}{p} \tag{4.10}$$

Burada *m* birbirine uyan özellik sayısı, *p* ise toplam özellik sayısıdır.

Çok Sayıda İkili Değişken Kullanmak

Bu yöntemi gerçeklemek için M adet nominal durumun herbiri için ayrı ayrı ikili değişkenler oluşturulur. Örneğin 4 renk bulunan bir renk özelliği için her renk bir ikili değişkenle kodlanır. Kırmızı renkte olan bir nesne için kırmızı değişkeni 1, diğer tüm değişkenler 0 değerini alır.

Tablo 4.3 - Kategorik Uzaklık Hesaplamasında Çok Sayıda İkili Değişken Kullanımı

Kırmızı	Mavi	Yeşil	Sarı
1	0	0	0

Tablo 4.3'den de görüleceği gibi renk bilgisini içeren bir kategorik alanın değeri tablodaki ifade edilmektedir. Ancak tabloda sadece bir renk 1 değeri almış olmasına karşın gerçekte bu, kategorik alanlar üzerinde sadece bir kategori seçilebilir anlamına gelmemektedir. Örneğin bu alan "müşterinin en sevdiği renk(ler)" alanı olsa idi ve müşteri kırmızıyı ve yeşili çok seviyorsa hem *kırmızı* hem de *yeşil* kategorileri 1 değerini alabilirlerdi.

İkili değişken yerine olasılıksal ve istatistiksel bir hesap da nominal kategorik değişkenlerin ifade edilmesinde yardımcı olmaktadır. Örnek olarak bir şirket kaydında "personelin eğitimi" ile ilgili bir kategorik alan olabilir. Bu alanda kategorik değerler ilkokul, lise, üniversite olmaktadır. Burada kategorik değerin her bir kategorisi için, toplamdaki payı ondalıklı olarak verilir. Örneğin eğer şirkette en fazla %34 oran ile lise mezunları bulunuyorsa ise lise kategorisi daha önceki örneklerde olduğu gibi 1 yerine 0,34 ile ifade edilir. Bu şekilde hiç bir bilgi kaybı olmaksızın tüm kategorik bilgiler saklanmış olur.

4.3.1.3. Karakter Dizisi Veriler

Veri madenciliği kapsamında demetlenmesi gereken kayıtların özelliklerinin çeşitliliği demetleme algoritmalarının her tipte veriyi kullanılabilir hale gelmesini zorunlu kılmıştır. Bu tiplerden bir tanesi de karakter dizisi yani karakter katarı veri tipidir. Veri ambarı içerisinde bir çok yerde karakter dizisi tipinde alanlar bulunmaktadır. Örneğin isimler, semtler, meslekler ve benzeri alanlarda nominal değerlerle ifade edilemeyecek kadar çok değer bulunduğunda karakter dizileri kullanılır. Bu karakter dizilerinin içerikleri bilgilerin de demetleme sırasında kullanılması gerekebilmektedir. Örneğin benzer isimlere sahip müşterin bir araya getirilmesi ya da benzer meslekleri icra edenlerin bir demet içerisinde toplanması gibi işler gerçeklenebilir hale getirilmelidir.

Karakter dizisi tipindeki alanları demetlemede uzaklık hesabına dahil etmek için en basit yol karşılaştırma yapmaktır. Ancak klasik karakter katarı karşılaştırma yöntemleri demetleme algoritması için gerekli olan iki karakter katarının birbirine benzerliğini vermemektedir. Bu karşılaştırma yöntemleri genelde alfabetik olarak hangi karakter katarının daha önce ya da sonra geldiğini bulmaktadırlar. Oysa

demetleme algoritmasında kullanılması gereken uzaklık ölçütü iki karakter katarının birbirine ne kadar benzediğini bulmalıdır.

Bunu gerçeklemek üzere bulanık mantık kullanılabilir. Bu yöntem kullanılarak verilen iki karakter katarı için benzerlik ölçütü bulunabilir. Bu yönem iki aşamalı olarak çalışmaktadır

Karakter Kullanım Benzerliği

İlk aşamada verilen iki karakter katarı içerisinde kullanılan karakterlerin ve bu karakterlerin adetlerinin benzerliği kontrol edilir. Öncelikle her iki karakter katarı için de bir karakter kullanım dizisi oluşturulur. Örneğin *GÖZLÜKÇÜ* kelimesi ve *GÖZLÜKÇÜLÜK* kelimeleri için aşağıdaki iki boyutlu diziler oluşturulmuştur:

A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y
0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	2	0	0

GÖZLÜKÇÜ

A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y
0	0	0	1	0	0	0	1	0	0	0	0	0	2	2	0	1	0	1	0	0	0	0	0	0	3	0	0

GÖZLÜKÇÜLÜK

Şekil 4.2 – Karakter Kullanım Benzerliği

Bu iki karakter katarının iki ayrın kaydın “meslek” alanında bulunduğu varsayalım. Bu durumda esasında aynı mesleğe sahip olan bu kişilerin mesleklere göre yapılan bir demetleme sonucunda aynı demet içerisinde çıkması beklenir. Bunu sağlamak üzere de yukarıdaki iki diziyi işleyerek bu iki karakter katarı arasında 0 ile 1 arasında bir benzerlik değeri üreten bir karşılaştırma yöntemi olması gerekmektedir.

İlk aşamada, her iki dizide de aynı indise sahip elemanların arasındaki farklar bulunur. Karakter katarlarını karşılaştırmak için kullanılan alfabeye Ω denilirse, n de bu alfabedeki harf sayısını temsil etmek üzere her karakter katarı için n boyutlu ve indisleri $i \in \Omega$ olmak üzere S dizisi oluşturulur. Bu S dizisinin her elemanı S_i , o karakter katarı içerisinde i. indis olan harften kaç adet bulunduğunu içermektedir. Bu koşullar altında iki karakter katarına ait yukarıda anlatılan şekilde oluşturulmuş S ve P dizileri kullanılarak şu şekilde hesap yapılır:

$$d_{SP} = \sum_{i=1}^{n} \left| S_i - P_i \right|, \quad i \in \Omega \tag{4.11}$$

Örneğin yukarıda yer alan gözlükçü ve gözlükçülük kelimelerinin birbirlerine olan uzaklıkları 3’dür. Bu uzaklık ve ikinci aşamada hesaplanan benzerlik değerleri

kullanılarak her kaydın ilgili karakter katarı tipindeki özelliği arasındaki benzerlik hesaplanabilir dolayısı ile demetleme gerçeklenebilmektedir.

Karakter Yeri Benzerliği

Çeşitli sebeplerden dolayı yanlış girilen ve eksik girilen karakterlerden oluşan karakter katarları için bir karşılaştırma sağlayan karakter yeri benzerliği yönteminde bulanık mantık bulunmaktadır. Klasik karakter katarı karşılaştırma işleminde bir karakter katarı içinde sırası gelen karakterin diğer karakter katarı içerisinde aynı yerde olup olmadığı kontrol edilir. Klasik karşılaştırma uyarınca aşağıdaki iki karakter katarı karşılaştırılırsa sonuç sıfır çıkar.

Mehmet

Mehmed

Oysa isme göre bir demetleme yapılırken bu iki ismin aynı demet içerisinde olması gerekir. Klasik karakter katarı karşılaştırma yöntemine biraz bulanık mantık yöntemi katılarak, küçük farklar için iki karakter katarının daha yüksek benzerlik göstermesi sağlanabilir. Şöyle ki, her aynı karakter için aşağıda gösterildiği gibi 1, farklı karakter için ise 0 puan verilirse ve tüm karakterler değerlendirildikten sonra verilen puanlar toplanıp toplam karakter sayısına bölünürse bir benzerlik oranı elde edilmiş olur.

M	e	h	m	e	t			
M	e	h	m	e	d			
1 +	1 +	1 +	1 +	1 +	0	=	5 / 6 =	0,83

Şekil 4.3 – Karakter Yeri Benzerliği Genel Hesaplaması

Görüldüğü gibi aslında çok benzer olan bu iki isim klasik karakter katarı karşılaştırma işlemi sonucunda farklı olarak nitelendirilirken, bulanık mantık ile karşılaştırma sonucu 0,83 oranında benzerlik göstermektedir.

Bu yöntemi daha da iyileştirmek amacıyla bazı eklemeler de yapılabilir. Örnek olarak yanlış sırada yazılan karakterlerin daha da iyi bir şekilde sezilip benzerlik oranını yükseltecek şekilde düzenlenmesi sağlanabilir. Bu tarz yanlışlıklar çoğunlukla bilgisayarla yazımda ortaya çıkmaktadır. Örneğin tuşların yanlış basılması sonucunda *Mehmet* yazılmak istenirken *Memhet* yazılması sıklıkla rastlanılan bir durumdur. Bu durumu tölare etmek amacıyla aynı indise sahip karakterlerin birbirini tutmaması durumunda hemen 0 puan verilmez. Öncelikle bir soldaki ve bir sağdaki karakterlere bakılır. Aranılan karakter bir solda ya da sağda varsa 0 yerine 1'den belirli bir puan düşülerek hesaba dahil edilir. Eğer bir komşu sol ya da sağ karakterde aranılan karakter bulunmuyorsa, bu kez 2. komşu sola ve sağa bakılır. Aranılan karakter burada ise bu kez daha yüksek bir ceza puanı 1'den düşülür. Bu

işlem belirli bir sayıda adım için devam ettirilir. Ancak çoğunlukla uygulamalarda bu sayı 2 ya da en fazla 3'tür.

Örnek vermek gerekirse, aşağıdaki iki karakter katarını iyileştirme olmadan karşılaştırırsak,

M	e	h	m	e	t			
M	e	m	h	e	t			
1 +	1 +	0 +	0 +	1 +	1	=	4 / 6 =	0,66

Şekil 4.4 – Klasik Karakter Yeri Benzerliği Karşılaştırması

Benzerlik oranı 0,66 çıkan fakat aslında aynı olup da sadece iki karakterin yanlış sırada olduğu iki karakter katarının, ceza puanının 0,2 olduğu bir bulanık mantık karakter katarı karşılaştırmasında benzerliği,

M	e	h	m	e	t			
M	e	m	h	e	t			
1 +	1 +	0,8 +	0,8 +	1 +	1	=	5,6 / 6 =	0,93

Şekil 4.5 - Bulanık Mantık İle Karakter Katarı Karşılaştırma

0,93 olarak çıkmaktadır. Görüldüğü gibi bu iyileştirme sonucunda daha önceki yönteme göre 0,66 çıkan benzerlik oranı çok büyük bir artış göstererek 0,93'lere çıkmıştır.

Sık rastlanan hatalardan birisi de bir karakterin ya da harfin eksik yazılması olayıdır. Örneğin kullanıcı Mehmet yerine Memet yazabilir. Bu durumda da benzerliği fazla olduğu halde bu iki karakter katarı klasik karşılaştırma yöntemi ile karşılaştırılırsa benzerlik oranı 0 çıkacaktır. Oysa yukarıda bahsedilen bulanık mantık prensibine göre çalışan karakter katarı karşılaştırma yöntemi kullanılırsa bu iki karakter katarının benzerlik oranı şu şekilde hesaplanır:

M	e	h	m	e	t			
M	e	m	e	t				
1 +	1+	0 +	0,8 +	0,8 +	0,8	=	4,4 / 6 =	0,73

Şekil 4.6 – Bulanık Karakter Katarı Karşılaştırma İle Noksan Yazılan Karakterlerin Sezilmesi

Görüldüğü gibi bu şekilde bulanık mantık kullanılarak karşılaştırılan bu iki karakter katarının benzerlik oranı çok yüksek çıkmaktadır. Normal karşılaştırma yöntemleri kullanılarak karşılaştırıldığında tamamı ile farklı olarak nitelendirilecek ancak aslında yazım hatalarından ya da benzer hatalardan dolayı farklı veya eksik karakterler içeren bu karakter katarlarının, bulanık karakter katarı karşılaştırma yöntemleri kullanıldığında benzerlik oranları çok yüksek çıkmaktadır.

Ancak bu yöntemin zayıf olduğu noktalar da bulunmaktadır. Örneğin tamamen farklı olan *yastık* ve *köstek* kelimeleri, bulanık mantık kullanılarak karşılaştırılırsa aşağıdaki gibi

y	a	s	t	ı	k			
k	ö	s	t	e	k			
0 +	0 +	1 +	1 +	0 +	1	=	3 / 6 =	0,5

Şekil 4.7 - Bulanık Mantık İle Karakter Katarı Karşılaştırmada Hatalı Durum Örneği

yüzde 50'ye varan bir benzerlik hesaplanır. Bu dezavantajı ortadan kaldırmak için öncelikle bu iki kelime arasındaki karakter kullanım uzaklığı hesaplanır. Yukarıda bölüm 4'de karakter kullanım benzerliği aşamasında anlatıldığı üzere hesaplanan karakter kullanım uzaklığı değeri, belirli bir değerden fazla ise bu iki karakter katarı tamamen farklıdır denilebilir. Örnek vermek gerekirse yukarıda yer alan iki kelimenin karakter kullanım uzaklığı 3'tür (y-k, a-ö, ı-e). Eğer eşik uzaklık olarak 2 seçilise, karakter kullanım uzaklığı 2'den fazla olan iki karakter katarı hiç bulanık mantık kullanılarak birbirleri ile karşılaştırılmadan doğrudan farklıdır sonucuna varılabilir.

4.3.2. Bellek Karmaşıklığının Azaltılması

Veri madenciliği uygulamalarında demetlenmesi istenilen verilerin adedinin çok fazla sayıda olması çok sık rastlanılan bir durumdur. Demetlenmesi istenilen verilerin özellik uzayının boyutunun her analiz için farklı olması ve özelliklerin tiplerinin değişken olması gibi nedenlerden ötürü çok fazla sayıda ve değişken özelliklere sahip n adet kaydı tutacak bir veri yapısına ihtiyaç vardır. Bu veri yapısı verilerden mümkün olduğu kadar fazlasını ana bellek içerisinde erişilebilir halde tutmalı, ana belleğe sığmayanları da ikincil belleklerde saklamalıdır. Bu ikincil belleklerde, örneğin diskte, saklanılan kayıtlar ihtiyaç duyulduğu anda ana belleğe getirilmelidir.

Demetlenecek verileri ve bu verilerin özellik bilgilerini algoritmanın kullanımına sunmak için en basit yol bir matris kullanmaktır. *n* adet kaydın demetleneceği bir analizde *m* boyutlu özellik uzayı kullanılacaksa *n x m* elemanlık bir matris tanımlanır:

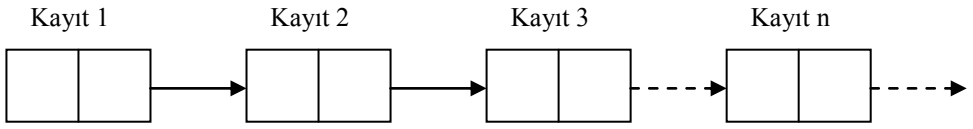
Tablo 4.4 – Demetlenecek Verileri Saklayan En Basit Veri Yapısı

	1	2	3	4	5	6	...	m-1	m
1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	⋯	$X_{1(m-1)}$	x_{1m}
2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	⋯	$X_{2(m-1)}$	x_{2m}
⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮
n	x_{n1}	x_{n2}	x_{n3}	x_{n4}	x_{n5}	x_{n6}	⋯	$x_{n(m-1)}$	x_{nm}

Bu veri yapısı çok basit olarak gerçeklenebilir ancak birçok dezavantajı vardır. Bunlardan ilki statik olmasıdır. Yani *n* ve *m* sayılarının belirli olması gerekmektedir. Matrisin satır sayısının arttırılması için yani demetlenecek elemanların adetlerinin arttırılabilmesi için algoritmayı içeren programın kaynak koduna erişip 2 boyutlu dizi tanımlamasını değiştirmek gereklidir. Aynı sınırlamalar, özellik uzayı içinde geçerlidir. Özellik uzayı boyutu olan *m* sayısının analizden önce programın kaynak koduna erişilip değiştirilmesi gerekmektedir.

Ayrıca matrisin elemanlarının yani 2 boyutlu dizinin elemanlarının tipleri sabit olmak zorundadır. Bu zorunluluk özellik uzayı içindeki bütün özelliklerin tiplerinin aynı olmasını zorunlu kılar. Şöyle ki, tamsayı tipinde sayısal değerler alan bir özellik ile ondalıklı ya da kategorik değerler alan bir özelliğin birlikte aynı *X* özellik uzayında bulunmalarına imkan tanınmamaktadır.

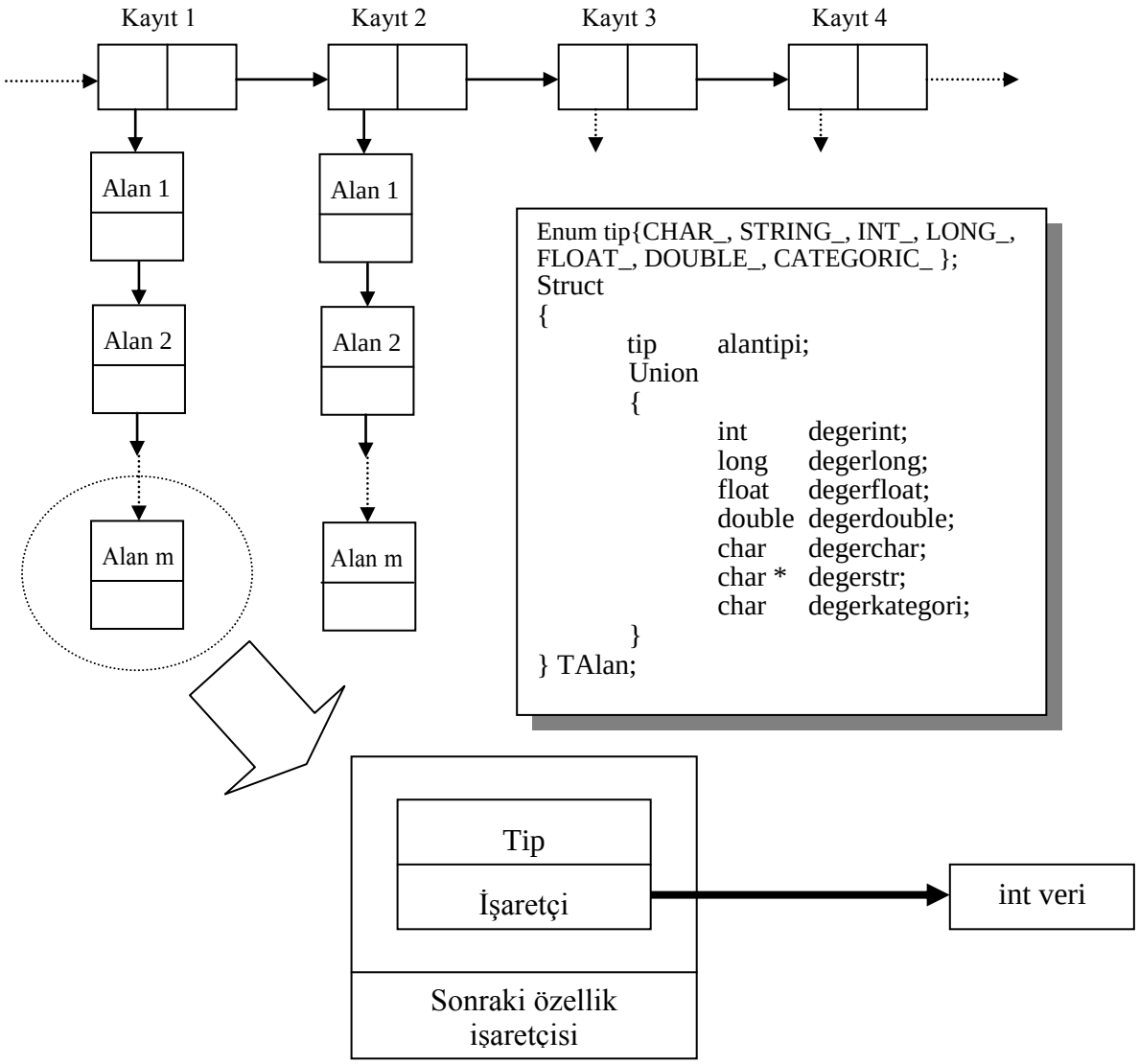
Bu yapının sınırlandırılmalarını kaldırmak üzere bir takım iyileştirmeler düşünülebilir. Bunlardan ilki kayıt adedi üzerindeki sınırlandırılmanın kaldırılması olacaktır. Yani algoritmayı gerçekleyen program ya da program parçası en fazla *n* adet kaydı demetleme yeteneğine sahip olmayacak, aksine sınırsız sayıda (belleğin ya da ikincil belleğin boyutu oranında) kaydın demetlenmesine olanak verecek şekilde olmalıdır. Bunu sağlamak üzere her bir kayıt bir bağlantılı liste yapısı içerisinde yer alan bir düğüme yerleştirilir. Bu yapıda her düğüm, kendisinden sonra gelen düğümü gösterdiği için herhangi bir sınırlandırma bulunmamaktadır. Yeni eklenen elemanlar yapının en sonuna eklenecek şekilde *dinamik* olarak oluşturulurlar. Bu yapı kabaca şöyledir:



Şekil 4.8 – Demetlemede Verileri Saklamak İçin Kullanılabilecek En Basit Dinamik Veri Yapısı

Bu yapı, kayıt adedi sorununu çözmekle birlikte özellik uzayının temsili sorununu çözememektedir. Bilindiği gibi demetlenmesi istenilen her kaydın analize bağlı olarak değişen sayıda özelliği bulunmaktadır. Yani bu özellik adedine *m* denilirse, dinamik bağlantılı listede her bir düğüm içerisinde *m* adet özellik bulunmalıdır. Oysa ki standart bağlantılı liste gerçeklemelerinde düğüm yapısı sabittir. Yani düğüm içerisinde analize bağlı sayıda m adet özellik bulunamaz. Bu sorunun giderilmesi için, bir önceki sorunun giderilmesinde izlenilen yaklaşıma benzer bir yaklaşım düşünülebilir. Her bir düğümün statik bir yapıya sahip olduğu klasik dinamik bağlantılı liste yerine her bir kayıt düğümünün, o düğüme ait özelliklerin bulunduğu

bir özellik listesine ait bir işaretçiye sahip olduğu bir yapı düşünülebilir. Bu şekilde esasında her bir düğüm hem kendinden bir sonra gelen kaydın adresini, hem de kendisine ait özelliklerin bulunduğu özellik listesinin adresini bulundurmuş olur. Bu şekilde gerçeklenmiş bir veri yapısı aşağıda şematik olarak gösterilmiştir.



Şekil 4.9 – Demetlemede Verilerin Tutulabileceği En Esnek Veri Yapısı

Şekilden de görüleceği gibi her bir kaydın yani düğümünün, kendine ait bir özellikler listesi bulunmaktadır. Bu özelliklerin ya da alanların bulunduğu veri yapısı da bir dinamik bağlantılı liste olduğu için, her kayıtta bulunabilecek özellik sayısında yani özellik uzayı boyutunda herhangi bir sınırlama yoktur.

Yukarıda geçen sorunlardan bir tanesi de özellik uzayı boyutunun değişmesi gibi her bir özelliğin tipinin de değişmesiydi. Yani her kaydın sahip olduğu özellikler analize bağlı olarak nicelik ve nitelik olarak farklılıklar göstermektedir. Düzenlenecek yeni

veri yapısı her türlü veriyi saklayabilecek tipte olmalıdır. Olası veri tipleri aşağıda listelenmiştir:

- Tamsayı
- Uzun tamsayı
- Reel
- Uzun Reel
- Karakter
- Kategorik
- Karakter katarı (String)

Çözüm olarak bu 7 ayrı veri tipini içinde barındıran bir union yapı düşünülebilir. Bu yapının tanımı şekil 4.9'da verilmiştir. Buradan da görüleceği üzere *TAlan* veri yapısında gereken tüm veri tipleri bulunmaktadır. Ancak bilindiği üzere union veri yapısında olası en uzun boyut, byte cinsinden hesaplanarak en uzun boyut için yer ayrılır. Dolayısı ile bir karakterlik alan için bile en az 4 ya da 8 byte yer harcanmaktadır. Özellikle veri madenciliğinde özellik uzayı ve demetlenecek örüntü kümesinin boyutlarının büyük olmasından ötürü çok fazla sayıda bellek alanı boşa harcanacaktır. Bunu engellemek üzere union yapı yerine şekil 4.9'un alt kısmında yer alan yapı kullanılır. Bu yapıda özelliğin tipi ve bellekte nerede olduğu bilgisi yani bir void işaretçi vardır. Bu yapının düğümleri şu şekilde olmaktadır:

```
typedef struct s_{
      tip   veri_tipi;
      void    *deger_isaretci;
} TAlan;
```

Yukarıda bahsedilen tüm bu veri yapıları, analizden bağımsız olarak, verileri algoritmanın işletileceği program tarafından kullanabilecek bir şekilde ana bellekte en esnek yoldan saklamak amacı ile geliştirilmişlerdir. Ancak bu esnekliğin maliyeti fazladan yer kaybı olmaktadır. Pragmatik bir örnek vermek gerekirse, demetlenecek kayıt adedinin *n,* özellik uzayı boyutunun *m* olduğu bir analizde esneklik uğruna harcanacak ana bellek miktarının boyutunu hesaplayalım. İlk olarak *m* adet kaydı içeren bir bağlantılı listede yer alan işaretçilerin boşa harcadığı yer miktarı olan b_1 hesaplanmalıdır:

$$b_1 = 2 \times m \times S_P \qquad (4.12)$$

Burada S_P, algoritmanın çalıştırıldığı sistemde bir işaretçi için ayrılan bellek miktarıdır. Bu değer genellikle 4 sekizlidir. Burada her bir düğümde, kendisinden sonraki elemanı gösteren bir işaretçi, bir de kendisine ait özellik listesi işaretçisi bulunduğu için, kayıtların tutulduğu dinamik listede işaretçiler için harcanan yer 2 x m x S_P sekizli olacaktır.

İkinci olarak özellikler bağlantılı listelerinde boşa harcanan b_2 bellek miktarının hesaplanması gereklidir. Bu hesap sırasında zorunlu olarak tutulan tip alanının kapladığı alan da hesaba dahil olmalıdır çünkü bu alanda, algoritma içerisinde işlenecek herhangi bir bilgi yoktur; sadece esnekliği sağlamak açısından eklenmiştir. Bu bilgilerin ışığı altında gereksiz yere harcanan b_2 bellek miktarı hesaplanacak olursa:

$$b_2 = (2 \times n \times m \times S_P) + (n \times m \times 1) \tag{4.13}$$

sekizli boyutunda gereksiz bellek harcandığı ortaya çıkar. *n* adet kayıttan herbirinin *m* adet özelliği olması durumunda ile n x m adet özellik listesi düğümü bulunmaktadır. Bu düğümlerde 2 adet işaretçi bulunduğundan işaretçiler için ayrılan bellek miktarı 2 x n x m x S_P olmaktadır. Ayrıca her bir düğümde bir de tip alanı bulunduğundan ve bu tip alanı 1 sekizli ile ifade edilebileceğinden dolayı da n x m x 1 boyutunda bir bellek alanı, bu iş için gereksiz yere harcanmış olmaktadır.

Sonuç olarak, sadece algoritmanın ihtiyaç duyduğu verileri esnek ve herhangi bir sınırlandırma olmaksızın ana bellek içerisinde saklamamızı sağlayan bu veri yapısı ile *b* boyutunda bir bellek alanı, demetleme algoritması tarafından kullanılmayacak ve gereksiz yere harcanmış olacaktır. Burada yer alan b,

$$b = b_1 + b_2 = 2n\,(1 + m)S_P + nm \tag{4.14}$$

formülü ile hesaplanabilir.

Örnek olarak 4 milyon adet kaydın demetleneceği bir analiz örneği düşünelim. Özellik uzayı boyutunun 35 olması durumunda gereksiz yere harcanan bellek miktarını hesaplayalım. $n=4.10^6$, m=35 olduğu için ve S_P=4 olarak alınırsa,

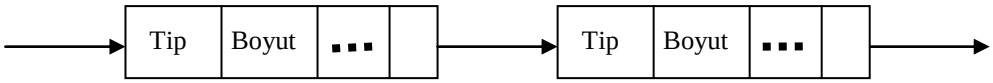
$$b = 2.4.10^6(1+35).4 + 4.10^6.35 \tag{4.15a}$$

$$b = 1292.\ 10^6 \text{ sekizli} \tag{4.15b}$$

olarak hesaplanır. Sonucu yorumlayacak olursak, sadece esnek veri yapısını sağlamak için, demetleme için kullanılacak verilerin dışında tam olarak 1.2 giga - sekizlik bir alanın kullanıldığı anlaşılır. Bu, günümüzün bilgisayarlarında bile yüksek sayılan bir bellek alanıdır. Üstelik bu toplam hesabına, uzaklık hesaplarında kullanılacak olan esas veriler girmemiştir. Analizde kullanılacak 35 boyutlu özellik

uzayındaki tüm özelliklerin tamsayı tipinde olduğu varsayılsa bile n x m x 2 sekizli yere daha ihtiyaç olacaktır ki bu basitleştirilmiş varsayım bile 280 mega-sekizlik bir bellek alanı gerektirmektedir. Sonuç olarak, önerilen bu yapının, esnekliği sağladığı yani sınırsız sayıda kaydın ve sınırsız boyutta özelliğin algoritma tarafından işlenebilecek bir şekilde, çalışma anında (runtime) dinamik olarak saklanan bir veri yapısında bulunduğu söylenebilir. Ancak bu yapının verimliliği çok düşüktür. Yukarıda yer alan basitleştirilmiş örnekte verimlilik neredeyse %17'dir. Dolayısı ile hem esnekliği sağlayacak hem de verimli olarak çalışacak yeni bir dinamik veri yapısına ihtiyaç duyulmaktadır.

Bunu sağlamak amacıyla yeni bir veri yapısı gerçeklenmiştir. Bu veri yapısında öncelikle gereksiz bellek alanı işgal eden işaretçilerden kurtulmak hedeflenmiştir. Bu itibarla öncelikle özellik uzayını tanımlayan bir veri yapısı gerçeklenmiştir. Bu veri yapısı analize başlamadan önce oluşturulacak ve tüm analiz boyunca kullanılacaktır. Özellik uzayında yer alan özellikleri, bu özelliklerin tiplerini ve her bir özelliğin bellekte kapladığı yerin büyüklüğünü içeren bilgilerin bulunduğu dinamik bir veri yapısı oluşturulacaktır. Bu veri yapısında kullanıcı bir kullanıcı arabirimi vasıtası ile analiz öncesi özellik uzayını tanımlayacaktır. Bu özellik uzayı tanımını içeren veri yapısının dinamik olmasını sağlamak üzere, bu veri yapısı bağlantılı liste olarak gerçeklenmiştir. Aşağıda ayrıntıları verilen bu veri yapısında hem sınırsız sayıda özelliğe ait bilgiler bulunabilmekte hem de her özelliğin nasıl kullanılacağına ilişkin ayrıntılar yer almaktadır.



Şekil 4.10 –Özellik Uzayını Tanımlayan Veri Yapısı

Tablo 4.5 – Özellik Uzayı Tanımlama Listesi Düğüm Yapısı

TDim ⇒ Boyut Tip Tanımlaması	
TDimTypes	**DimType**
	Bu boyuttaki verilerin tipi
char	**Definiton[40]**
	Boyutun tanımı
İnt	**Size**
	Veri tipinin uzunluğu
float	**Weight**
	Bu boyutun hesaplamalardaki ağırlığı
int	**Status**
	Bu boyutun hesaplamalara girip girmeyeceği
short	**ClusterNoField**
	Bu boyutun KÜME numarası tutan bir alan olup olmadığı
short	**TotalCategories**
	Kategorik veri ise toplam kategori adedi
TCompNodePtr	**CompTable**
	Eğer kullanıcı tarafından girilmiş bir özel karşılaştırma tablosu varsa buna ilişkin tabloya işaretçi
struct Node	***prev,*next**
	Bir önceki ve sonraki alan düğümlerini gösteren işaretçiler

Yukarıda yer alan yapıda, özellik uzayını belirleyen bağlantılı listedeki düğümlerde, ilgili özellik boyutu ile ilgili ayrıntılı bilgiler bulunur. Bu bilgilerden ilki, o özelliğin tipidir. Bu tipler

- CHAR_ : karakter
- STRING_ : karakter katarı
- INT_ : tamsayı
- LONG_ : uzun tamsayı
- FLOAT_ : reel
- DOUBLE_ : uzun reel
- CATEGORIC_ : kategorik

tiplerinden birisi olabilir. Bu tipler en genel anlamda hemen hemen tüm değişken tiplerini karşılamaya yeterlidir. Bunlara ek olarak kullanıcının bu boyut ile daha anlaşılır bir bilgi alabilmesi için *definition* alanı eklenmiştir.

Her bir özellik boyutunun bellekte saklanması için gerekli olan bellek miktarı ise sekizli cinsinden olmak üzere *size* alanında bulunmaktadır. Örneğin, kullanılan sisteme özgü olarak değişmekle beraber, genel olarak tamsayı tipinde bir özellik değeri için ayrılması gereken bellek alanı boyu 2 sekizli olmakta ve bu miktar ilgili özelliğin size alanında bulunmaktadır.

Ayrıca Bölüm 3'de de değinildiği gibi her bir özellik boyutunun, toplam uzaklık hesabındaki etkisinin aynı olmasının istenmediği durumlar bulunmaktadır. Bu tür durumlarda her özellik boyutuna bir ağırlık tanımlaması yapılır ve uzaklık hesabında her özellik kendi ağırlık değeri ile çarpılarak toplama girmektedir. Bu durumda uzaklıklar aşağıdaki gibi hesaplanabilmektedir:

$$d(i,j) = \sqrt{w_1\left|x_{i1} - x_{j1}\right|^2 + w_2\left|x_{i2} - x_{j2}\right|^2 + ... + w_p\left|x_{ip} - x_{jp}\right|^2} \quad (4.16)$$

Ayrıca gene Bölüm 3'de anlatıldığı gibi bazı özelliklerin benzerlik ya da uzaklık hesaplarında hesaba girmemesi istenilir. Bu durum genellikle demetlemeye etki etmesi istenmeyen ancak demetleme işlemi sonrasında sonuçları incelerken gereksinim duyulan alanlar üzerinde uygulanır. Bu ihtiyaç göz önünde tutularak her bir özelliğin uzaklık hesaplarına dahil olup olmayacağını belirleyen bir belirteç alan da ilgili düğüm yapısına eklenmiştir. Bu alan değeri *pasif* olursa o özellik boyutu ile ilgili herhangi bir hesaplama yapılmamaktadır. Ancak *status* alanı değeri AKTİF olan özellik boyutları hesaplamalara dahil edilmektedir. Bunu formülüze etmek gerekirse,

$$d(i,j) = \sqrt{w_1 s_1\left|x_{i1} - x_{j1}\right|^2 + w_2 s_2\left|x_{i2} - x_{j2}\right|^2 + ... + w_p s_p\left|x_{ip} - x_{jp}\right|^2} \quad (4.17)$$

olmaktadır. Burada yer alan s_i değeri status alanı değeridir. Status alanı *aktif* olan boyutlarda $s_i = 1$, *pasif* olan boyutlarda ise $s_i = 0$ olmaktadır.

Ayrıca eğer özellik tipi kategorik ise, bu kategoride yer alan toplam kaç adet farklı değerin olduğunun bilinmesi, demet merkezlerinin tekrar hesaplanmasında gereklidir. Bu yüzden olası kategorik değerlerin kaç adet olduğu bilgisi burada tutulmaktadır. Genellikle çok geniş bir aralıkta değerler alabilen özellikler, alt aralıklara bölünür ve bu aralıklara düşen değerlere bir kategori seviyesi atanarak bu şekilde demetleme algoritmasına dahil edilirler. Örneğin bir banka müşterisinin ortalama aylık mevduatının tutarı düşük, orta ve yüksek biçiminde kategorilere ayrılarak demetleme işlemine girer.

Demetleme algoritmasının yürütülmesi sırasında hesaplamalarda kullanılan uzaklık ölçütü yerine kullanıcı tanımlı uzaklık tanımlarına gerek duyulabilmektedir. Özellikle kategorik tipte veriler için önem taşıyan bu kullanıcı tanımlı uzaklıkların bir tablo halinde düzenlenmesi uygun olacaktır. Bu şekilde iki kategorik veri arasındaki uzaklık kullanıcı tarafından analiz başlamadan önce girilebilmektedir. Bunu sağlamak üzere her özellik boyutu için kullanıcının elle girdiği uzaklıkların bulunduğu bir bağlantılı liste oluşturulmuştur. Bu bağlantılı liste içerisinde aşağıdaki gibi tanımlanmış düğümler bulunmaktadır:

```
// Comparision Table Nodes
typedef struct _CompTableNode{
  char              *Value1, *Value2;
  float             Result;
  struct _CompTableNode  *next;
} TCompNode;
```

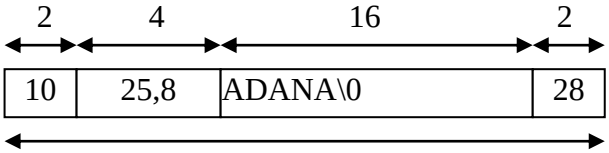
Bu düğümlerde karşılaştırılacak iki değerin adresleri ve bu iki değer karşılaştırıldığında, aralarında benzerlik olarak hesaplamaya katılacak olan reel sayı tipinde bir benzerlik bilgisi mevcuttur.

Bu şekilde tanımlanan özellik uzayı tanımlama listesi, analizden önce kullanıcı tarafından yapılacak demetlemeye uygun olarak oluşturulur. Bu adımda kullanıcı demetlemede hangi özelliklerin olacağını, bunların hangi tipte değerler aldıklarını, bu özelliklerin hangilerinin uzaklık hesaplarında kullanılacağını hangilerinin kullanılmayacağını, kendisinin girmek istediği kategoriler arası uzaklıklar değerleri varsa bunları tablosunun hazırlanması ve benzeri bilgilerin sağlanmasını gerçekleştirmektedir.

4.3.3. Kompakt Ve Modüler Veri Yapısı

Bölüm 4.3.1'de anlatıldığı gibi çok fazla sayıda verinin demetlenmesinin efektif ve kabul edilebilir bir biçimde sağlamak üzere kompakt veri yapısı önerilmektedir. Bu yapı, daha önce anlatılmış olan özellik uzayı tanımlama yapısına bağlı olarak çalışmaktadır. İlk önerilen esnek veri yapısında Şekil 4.9'da yer alan her bir kayda ait özelliklerin temsil edilmesinde kullanılan 2 x m x n adet işaretçinin algoritmada ihtiyaç duyulmadığı halde esnek veri yapısını sağlamak amacı ile yapıya dahil edildiği görülmektedir. Bu işaretçilerden tasarruf edilebilirse çok büyük miktarlarda bellek alanı algoritmanın esas kullanacağı verileri saklamakta kullanılabilir hale getirilebilir. Bu nedenden ötürü, verilerin çok daha sıkışık bir düzende yer aldığı kompakt bir veri yapısı önerilmiştir.

Önerilen kompakt veri yapısı uyarınca, her bir kayda ait özellikler ardışıl bellek alanlarına arka arkaya gelecek şekilde yerleştirilirler. Bu işlem aşağıdaki şekilde görülmektedir:



Şekil 4.11 – Kompakt Veri Yerleşimi

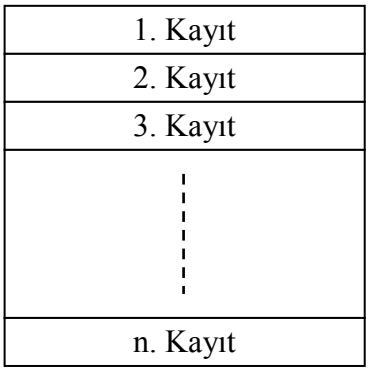
Örnekten de görüleceği üzere özellik uzayı boyutu 4 olan bir analizde belirli bir kayıt için yukarıdaki bellek yerleştirilmesi elde edilmiştir. Analizde yer alan özelliklerin ayrıntıları aşağıdaki tabloda verilmiştir:

Tablo 4.6 – Örnek Analiz İçin Özellik Uzayı

Özellik Boyutu	**Tip**	**Kapladığı Bellek Miktarı**
1	INT_	2 sekizli
2	FLOAT_	4 sekizli
3	STRING_	10 sekizli
4	INT_	2 sekizli

Dolayısı ile ardışıl olarak 24 sekizlik bellek alanı bu kayda ilişkin özellikleri saklamak için ayrılmıştır. Oysa ki daha önce önerilen bağlantılı liste yapısı kullanılsaydı 24 sekizlik yer için her biri 4 sekizlik 8 adet işaretçi (4 tanesi bağlantılı liste için 4 tanesi alan tipinin değişken olmasından dolayı veriyi gösteren işaretçi) kullanılacak ve ayrılacak toplam bellek alanı 24 + (4 x 8) = 56 sekizli olacaktı. Bu durumda belleğin kullanım verimliliği %42 olmaktadır. Ancak şekil 4.11'de önerilen yapıda verimlilik %100 olmaktadır çünkü algoritmanın çalışması için ihtiyaç duyulan verilerden başka hiçbir nedenle fazladan bellek alanı kullanılmamıştır.

Algoritmanın üzerinde çalıştırılacağı örüntü kümesine ait her bir kayıt, yukarıda anlatılan veri yapısına uygun olarak ardışıl bellek alanları içerisinde saklanırlar. Bu durumda n kayıtlık bir analizde ana bellekte bulunması gereken n adet kayıt şu yapı içerisinde bulunacaktır:



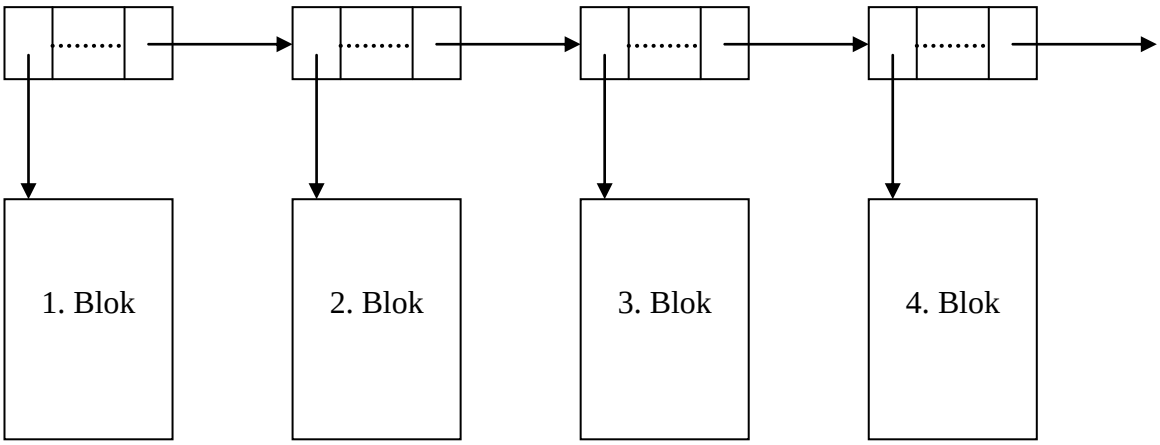
Şekil 4.12 - Demetlenecek Kayıtların Ana Bellekteki Yerleşimi

Bu şekilde kayıtlar ardışıl bellek gözlerine yerleştirilerek ana bellek içerisinde saklanabilirler.

4.3.4. Bellek Yönetim Sistemi

Bu varsayım ana bellek içerisinde *m* adet özelliği olan *n* adet kaydın sığabileceği kadar büyük bir bellek alanının bulunduğu koşulu altında geçerlidir. Örneğin demetlenecek örüntü kümesinin eleman sayısının 4 milyon olduğu ve özellik uzayı boyutunun az önce verilen örnekteki gibi 24 sekizli olduğu bir analizde tek parça halinde işletim sistemden alınması gereken bellek miktarı $24 \times 4 \times 10^6 \approx 91$ mega sekizlidir. Bu bellek miktarı, analizde kullanılacak boyut uzayı küçük olmasına rağmen gene de işletim sistemi tarafından rahatlıkla karşılanabilecek bir bellek miktarı değildir. Üstelik ana bellek her zaman bu isteği karşılayacak kadar büyük olmayabilir ya da büyük olsa dahi parçalanmış boş yerler olduğu için tek bir blok halinde algoritmanın kullanımına ayrılamayabilir. Örneğin 64 mega-sekizlik ana belleğe sahip bir bilgisayar sisteminde 91 mega sekizlik yer istenilirse bilgisayar işletim sistemi devreye girerek görüntü belleği yani ikincil belleğin bir kısmını ana bellek gibi kullanmaya başlayacak dolayısı ile işlemleri yavaşlatacaktır.

Oysa gereksinim duyulan bellek miktarını tek parça halinde almaktansa bu belleği parçalar halinde almak her bakımdan avantajlı olacaktır. Bunu sağlamak üzere *bloklar* tanımlanmıştır. Tanım olarak blok, içinde belirli sayıda kayıt bulunan bellek bölgesidir. Blokların alabileceği maksimum kayıt adedi belirlidir ancak “her blok içerisinde maksimum kayıt adedi kadar kayıt bulunur” şeklinde bir ifade doğru değildir. Bu blokların her birisi *t* adet örüntü kaydının bilgisini içerisinde tutabilecek ve istenildiğinde blok içeriği tamamen ana belleğe kaldırılabilecektir. Her bir bloğun kapasitesi olan *t* kayıt adedi analiz başında değiştirilebilecektir. Bu şekilde değişik işletim sistemlerinde ve değişik ortamlarda optimum seviye sağlanabilecektir. Her bir bloğa ait bir takım bilgiler bulunmaktadır. Bunların başında blok numarası, ilgili blokta kaç adet kayıt bulunduğu (bloğun tamamı dolu olmayabilir), bloğun bellekte bulunduğu yerin başlangıç adresi, bloğun o anda ana bellekte mi yoksa ikincil bellekte mi olduğunu belirten bir belirteç ve benzeri bilgiler yer alır. Tüm bu bilgiler, blokların bilgilerini içeren bir bağlantılı liste yapısında bulunmaktadır. Bu şekilde hem blok sayısına bir sınırlama getirilmemiş olur hem de dinamik bir veri yapısı kurulmuş olur. Bu yapının şematik gösterilimi şu şekildedir:



Şekil 4.13 – Verileri Tutan Blok Yapısı

Tablo 4.7– Verileri Tutan Blok Yapısındaki Düğüm Tanımlanması

```
// Blok Veri Yapısı
typedef struct NodeType{
    char             *Address;        //Blok baş. adresi
    short            BlockNo;         //Blok no
    int              BlockCapacity;   //Blok Max. kapasite
    int              BlockByteSize;   //Bloğun byte boyu
    int              ActualRecInBlock;//Bloktaki kayıt adedi
    int              AgingCounter;    //Yaşlanma sayacı
    TStatus          Status;          //Bellek-Disk bayrağı
    struct NodeType  *Prev,*Next;     //Bağlantı alanları
}  TBlock;
```

4.3.4.1. Adaptif Blok Yönetimi

Bu yapıda yer alan blokların boyutları, işletim sistemi ve diğer kriterlere göre analiz parametresi olarak algoritma çalıştırılmadan önce değiştirilebilmektedir. Üstelik ana belleğin çok fazla parçalanmış (fragmantated) olduğu durumlarda blok için yer alma mekanizmasına iyileştirilme getirilmiştir. Bu iyileştirme kapsamında kullanıcıdan analiz öncesi bir blok içerisinde olabilecek en fazla ve en az kayıt adedi istenmektedir. Örneğin kullanıcı en fazla 100.000 kayıt içeren bloklar olsun en az da 1000 kayıt içeren bloklar olsun diyebilmektedir. Bu bağlamda algoritma, verileri bulunduğu ortamdan (veri tabanı veya veri ambarı), algoritmanın kullanabileceği veri yapısına aktarırken öncelikle en fazla kaydı alabilecek kadar büyük bir blok için gerekli alanı işletim sisteminden talep eder. İşletim sistemi eğer bu talebi karşılayamazsa, kayıt adedi belirli bir miktar kadar azaltılır ve bu yeni sayıda kaydın sığabileceği bir bellek alanı alınmaya çalışılır. Bu da işletim sistemi tarafından

karşılanamazsa bir miktar daha azaltılarak tekrar istenir. Bu süreç, işletim sisteminden alınmaya çalışılan yer, kullanıcı tarafından analiz öncesinde belirtilen minimum blok boyundan daha küçük olana kadar devam eder. Her adımda ise kullanıcının belirlediği *adım* kadar istenilen bellek alanı azaltılır. Böylelikle çeşitli programlar tarafından sistemin ana belleği parça parça olmuş olsa dahi aralardaki boşluklar optimum şekilde blok yapılarıyla doldurulmaktadır.

4.3.4.2. Takas Yönetimi

Uygulamanın çalışması süresince yukarıda anlatılan blok yapıları içerisinde yer alan örüntü kayıtlarının sayısının çok olması durumunda bütün blok yapılarının ana bellek içerisinde yer alması mümkün olmayacaktır. Algoritmanın o anda ihtiyaç duyduğu kayıtları içeren blokların ana bellekte olması yeterli olmakta diğer kayıtları içeren blokların ise ana bellekte yer kazanmak amacıyla diske kaldırılması gerekmektedir. Tasarlanan uygulamada bu mimari gerçeklenmiş ve verilerin işlenmesini hızlandırmıştır. Bu sayede sınırsız sayıda ve sınırsız boyutlu uzay içindeki verilerin demetlenmesine olanak sağlanmıştır.

Bir bloğun o anda diskte mi olduğu yoksa ana bellekte mi yer aldığı bilgisi ise blokların bilgilerini tutan br bağlantılı liste içerisinde bulunmaktadır. Burada ayrıca bloğun diskte bulunması halinde blok içeriğinin disk üzerinde hangi dosyada saklandığı bilgisi de bulunmaktadır.

4.3.4.3. Çerçeve Bellek Kullanımı

Geliştirilen çekirdek mimaride hem paralel çalışmaya imkan sağlamak hem de algoritmanın çalışması sırasında diğer uygulamaların da çalışmalarını doğru devam ettirebilmeleri amacı ile algoritmanın kullanacağı bellek alanı önceden sınırlandırılabilir. Böylelikle hem algoritmanın çalıştığı prosesin ne kadar bellek alanı kullanacağını önceden belirleyebilir hem de algoritmanın birden fazla proses ile paralel olarak çalıştırılması esnasında her bir prosesin ne kadar bellek kullanacağını ayarlayabiliriz.

Bunu sağlamak üzere sürecin başında ayarlanması gereken parametre aktif blok adedidir. Bu sayı ile belirli bir anda bellekte bulunabilecek en fazla blok adedi belirtilmiş olur. Algoritmanın işleyişi esnasında sadece bu parametrede belirtilen blok sayısı kadar blok bellekte bulunurken diğer bloklar disk üzerinde tutulurlar. İşlenen son blok verinin işi de bitince bellekte yer alan ve en uzun süredir işlem görmeyen blok diske kaldırıldıktan sonra işlenmesi gereken yeni verileri içeren blok ana

belleğe getirilir. Bu sayede belirli bir anda ana bellekte sadece çerçeve miktarı kadar blok bulunur.

Çerçeve blok kullanımı ile algoritmanın paralel çalışmaya olan desteği arttırılmış ayrıca bir den fazla uygulamanın çalıştığı sunucular ya da mainframe makinalar üzerinde de diğer uygulamaların daha rahat çalışmaları sağlanmıştır.

5. Sonuç ve Tartışma

Sonuç olarak veri madenciliği algoritmalarının çalışabileceği bir çerçeve veri yapısı ortaya konulmuştur. Bu veri yapısı ile bellek karmaşıklığı azaltılmış, en yüksek derecede esneklik, yani işlenecek kayıt adedinin ve özellik uzayının boyutlarının farklı olmasının algoritmanın çalışmasına etkileri yokedilmiş, yazılım içerisinde kod değişikliğine neden olmasına engel olunmuştur. Geliştirilen bellek yönetim sistemi, veri madenciliğinde işlemlerinde kullanılacak verilerin ana belleğe optimum şekilde bloklar halinde yerleştirilmektedir. Bunlara ek olarak işletim sisteminin sağladığı bellek yönetimi katmanının üzerinden bir bellek yönetim sistemi de yapılan bu çalışma kapsamında gerçeklenmiştir.

Yapılan çalışmada tüm veri madenciliği işlemleri için kullanılabilecek bir çerçeve veri yapısı tanımlanmıştır. Bu yapı ile çalışmak üzere ilk etapta sadece K-Means algoritması kodlanmıştır. Daha farklı veri madenciliği algoritmalarının da gerçeklenmesi durumunda geliştirilmiş olan yazılım çok daha işlevsel olacaktır. Bu çalışmada, işin en zor yanlarından birisi olan, işlenecek veriyi en az yer kaplayacak şekilde saklayacak bir mimarinin geliştirilmesi gerçekleştirilmiştir. Analiz yapılacak örüntü sayısının çok büyük olması durumunda dahi optimum sonucu veren veri yapıları tasarlanmıştır. Ayrıca örüntü uzayının boyutlarının ve boyut tiplerinin her analizde değişmesine rağmen örüntü kayıtlarını saklamak üzere jenerik bir yapı gerçeklenmiştir. Bu şekilde saklanan veriler üzerinde analiz yapmak çok daha kolay hale gelmiştir. Burada gerçeklenen K-Means algoritmasının haricinde hemen hemen her türlü demetleme ve veri madenciliği algoritmasının üzerine bina edilebileceği bu mimari yapının geliştirilmesi, yapılan bu tez çalışmasının bir sonraki adımıdır.

Geliştirilen karakter katarı benzerliği yöntemlerinde anlatılan ölçütlerden sadece “karakter kullanım benzerliği” ölçütü K-Means algoritması için uygun olmaktadır. Oysa önerilen “karakter yeri benzerliği” yönteminin de K-Means algoritmasında kullanılacak şekilde yeniden düzenlenmesi ya da K-Means algoritmasında bu benzerliği kullanmak üzere, aynı demet içerisine düşen karakter katarı tipindeki verilerin merkezlerinin hesaplanmasında yeni bir yöntemin geliştirilmesi ile daha uygun demetler oluşabilir. Bu adımlar da tez kapsamında gelecek çalışmalar kapsamındadır.

Ayrıca çekirdek mimari üzerinde gerçeklenen K-Means algoritmasının ihtiyaç duyduğu başlangıç merkezlerinin seçilmesi adımının da geliştirilmesi gerekmektedir. Burada gerçeklenen modelde, literatürde geleneksel olarak kullanılan rastgele seçme yöntemi kullanılmıştır. Bunun sonucu olarak algoritma, toplam hatanın karesi fonksiyonunun global minmumunu bulmak yerine çoğunlukla yerel minimumlarını bulmaktadır. Yani algoritmanın yürütülmesi sonucunda üretilen demetleme şekli, o veriler üzerinde yapılabilecek en iyi demetleme şekli olmayabilir. Bu eksikliğin giderilmesi için bazı çalışmalar yapılabilir. Bu çalışmalar ile K-Means algoritması çalışmaya başlamadan önce örüntü kümesi analiz edilerek başlangıç noktalarının seçimi daha doğru yapılabilir. Ancak K-Means başlangıç noktalarının seçimi konusunda bugüne kadar yapılan çalışmalar sonucunda genel olarak geçerliliği olan ve uygulanabilirliği yüksek olan öneriler getirilememiştir. Önerilen başlangıç noktası seçimlerinin çoğu belirli özellikleri taşıyan veri kümeleri üzerinde uygun demetleme sonuçları üretebilmektedir. Bazı yöntemler de çalışma süreleri bakımından geniş ölçekli örüntü kümeleri üzerinde çalıştırılmak için uygun değildir.

Sonuç olarak veri madenciliği algoritmalarının gerçeklenmesinde kullanmak üzere, kendi bellek yönetim sistemi olan bir çekirdek mimari gerçeklenmiş, karakter katarı ve kategorik tipteki alanların demetlemeye dahil edilebilmesi için yeni benzerlik ölçütleri tanımlanmış ve oluşan yapı üzerinde ilk olarak K-Means algoritması gerçeklenmiştir.

KAYNAKLAR

Anderberg, M.R., 1973. Cluster Analysis for Applications. Academic Press Inc., New York.

Bezdek, J.C., 1981. Pattern Recognition With Fuzzy Objective Function Algorithms. Plentum Press New York.

Bentley, J.L. and Friedman, J.H., 1978. Fast Algorithms for Constructing Minimal Spanning Trees in Coordinate Spcaes. *IEEE Trans. On Compu.* C-27, 6 (June), 97-105.

Can, F. 1993. Incremental Clustering For Dynamic Information Processing. *ACM Trans. Inf. Sys.* 11,2, (Apr 1993), 143-164

Diday, E. And Simon, J.C. 1976. Clustering analysis in Digital Pattern Recognition, K. S. Fu, Ed. Springer-Verlag, Secaucus, NJ, 47-94

Dubes, R.C. and Jain, A. K. 1995. Clustering Techniques: The user's dilemma. *Pattern Recogn*., 8, 247-260.

Fisher, D. And Langley, P. 1986. Conceptual clustering and its relation to numerical taxonomy. In *Artificial Intelligence and Statistics*, A.W. Gale Ed. Addison-Wesley Longman Pub. Co. Inc., Reading. MA, 77-116.

Gotlieb, G.C. and Kumar, S. 1968. Semantic Clustering of Index Terms. J. ACM 15, 493-513.

Huang, Z. 1997. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets In Data Mining. *Research Issues on Data Mining and Knowledge Discovery*.

Jain, A.K. and Dubes, R.C. 1988**.** Algorithms for Clustering Data.Prentice-Hall advanced reference series. Prentice Hall Inc. New Jerset.

Jain, A.K. and Mao, J. 1994. Neural Networks and Pattern Recognition. In computational Intelligence: Inititating Life, J. M. Zurada R. J. Marks and C. J. Robinson 194-212.

Kohonen, T. 1989. Self-Organization and Associative Memory. 3rd ed. Springer informaion sciences series. Springer-Verlag, New York, NY.

Kurita, T. 1991. An eficient agglomerative clustering algorithm using a heap. *Pattern Recog 24, 3 (1991), 205-209.*

Linoff G. Ve Berry M.J.A., 1997. Data Mining techniques For Marketing Sales and Customer Support, Wiley Computer Publishing, New York.

McQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceeding of the 5th Berkeley Symposium On Mathematical Statistics and Probability, 281-297.*

Michalski, R., Stepp, R. E., and Diday, E. 1983. Automated construction of classifications:conceptual clustering versus numerical taxonomy. *IEEE Trans. Pattern Anal Mach. Intell. PAMI-5, 5(Sept), 396-409.*

Murty, M.N. and Krishna, G. 1980. A computationally efficient technique for data clustering. *Pattern Recog. 12, 153-158.*

Gower, J. C. And Ross, G. J. S. 1969. Minimum spanning tree and single-linkage cluster analysis. *Appl. Stat. 18, 54-64.*

Sarıel, S. 2002. Kullanıcı Kesitleri İle Yüz İfadelerini Analiz Eden Bir Çoklu Etmen Sistemi Uygulaması, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstabul.

Sethi, I. and Jain, A. K. 1991. Artificial Neural Networks and Pattern Recognition: Old and New Connections. Elsevier Science Inc., New York, NY.

Stahl, H. 1986. Cluster analysis of large data sets. In Classification as a Tool of Research, W. Gaul and M. Schader, Eds. Elsevier North Holland Inc., New York, NY, 423-430.

Su, M.C. and Chou, C.H. 2001. A Modified Version of the K-Means Algorithm with a Distance Based On Cluster Symmetry. *IEE Trans. Pattern Anal. And Mach. Intell., 23 (June), 674-680.*

Zahn, C. T. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput. C-20 (April) 68-86.*

ÖZGEÇMİŞ

Ahmet Cüneyd TANTUĞ, 04/09/1978 tarihinde Adana'da doğmuştur. Orta ve lise eğitimini Adana Anadolu Lisesi'nde tamamladıktan sonra Hacettepe Üniversitesi Bilgisayar Bilimleri Mühendisliği Bölümü'nde lisans eğitimine başlamış ve İstanbul Teknik Üniversitesi Kontrol ve Bilgisayar Mühendisliği Bölümünü 1. olarak bitirerek lisans eğitimini bitirmiştir. 2000-2002 yılları arasında Garanti Teknoloji A.Ş.'de Bilgisayar Mühendisi olarak çalışmış, Mart 2002'den itibaren de İ.T.Ü. Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak çalışmalarına devam etmektedir.