MÜHENDİSLİKTE YAPAY ZEKA VE UYGULAMALARI

Editörler

Prof. Dr. Orhan TORKUL - Prof. Dr. Sevinç GÜLSEÇEN

Doç. Dr. Yılmaz UYAROĞLU

Yrd. Doç. Dr. Gültekin ÇAĞIL

Arş. Gör. Dr. Muhammed Kürşad UÇAR

MÜHENDİSLİKTE YAPAY ZEKA VE UYGULAMALARI

Editörler

Prof. Dr. Orhan TORKUL - Prof. Dr. Sevinç GÜLSEÇEN

Doç. Dr. Yılmaz UYAROĞLU

Yrd. Doç. Dr. Gültekin ÇAĞIL

Arş. Gör. Dr. Muhammed Kürşad UÇAR

Sakarya Üniversitesi, Mühendislik Fakültesi / www.mf.sakarya.edu.tr/

Sakarya Üniversitesi Yapay Zeka Sistemleri Uygulama ve Araştırma Merkezi / www.yazsum.sakarya.edu.tr

İstanbul Üniversitesi / www.istanbul.edu.tr

Kocaeli Üniversitesi / www.kocaeli.edu.tr

Bu kitap ücretsiz dağıtılmak üzere ülkemizin geleceği için yapılmış bir hizmettir.

1. Baskı, Ekim 2017, SAKARYA
ISBN, 978-605-4735-98-3
Sakarya Üniversitesi Yayınları No: 184

Gece destan yazan, sabahında devleti için hizmet eden kahraman milletimize ithaf olunur.

İçindekiler

ÖNCE SÖZ

"Düzenim bozulur, hayatım üstü altına gelir" diye endişe etme. Nereden biliyorsun hayatın altının üstünden daha iyi olmayacağını.
Şems-i Tebrizi

Her başlangıç için önce söz vermek gerekir. Verilen sözün yerine getirilmesini için de canla başla yılmadan çalışmak gerekir. Bu kitabı elinize aldıysanız sizin de kendi kendinize söz verme zamanınız geldi demektir. Kendi kendinize çalışmaya başladığınız yapay zeka alanında yılmadan yorulmadan devletimiz için bir şeyler ortaya çıkaracağınıza söz verin ve başlattığınız zinciri kırmadan, yolunuzdan ayrılmadan, çalışmaya söz verin. Her gün bir adım ...

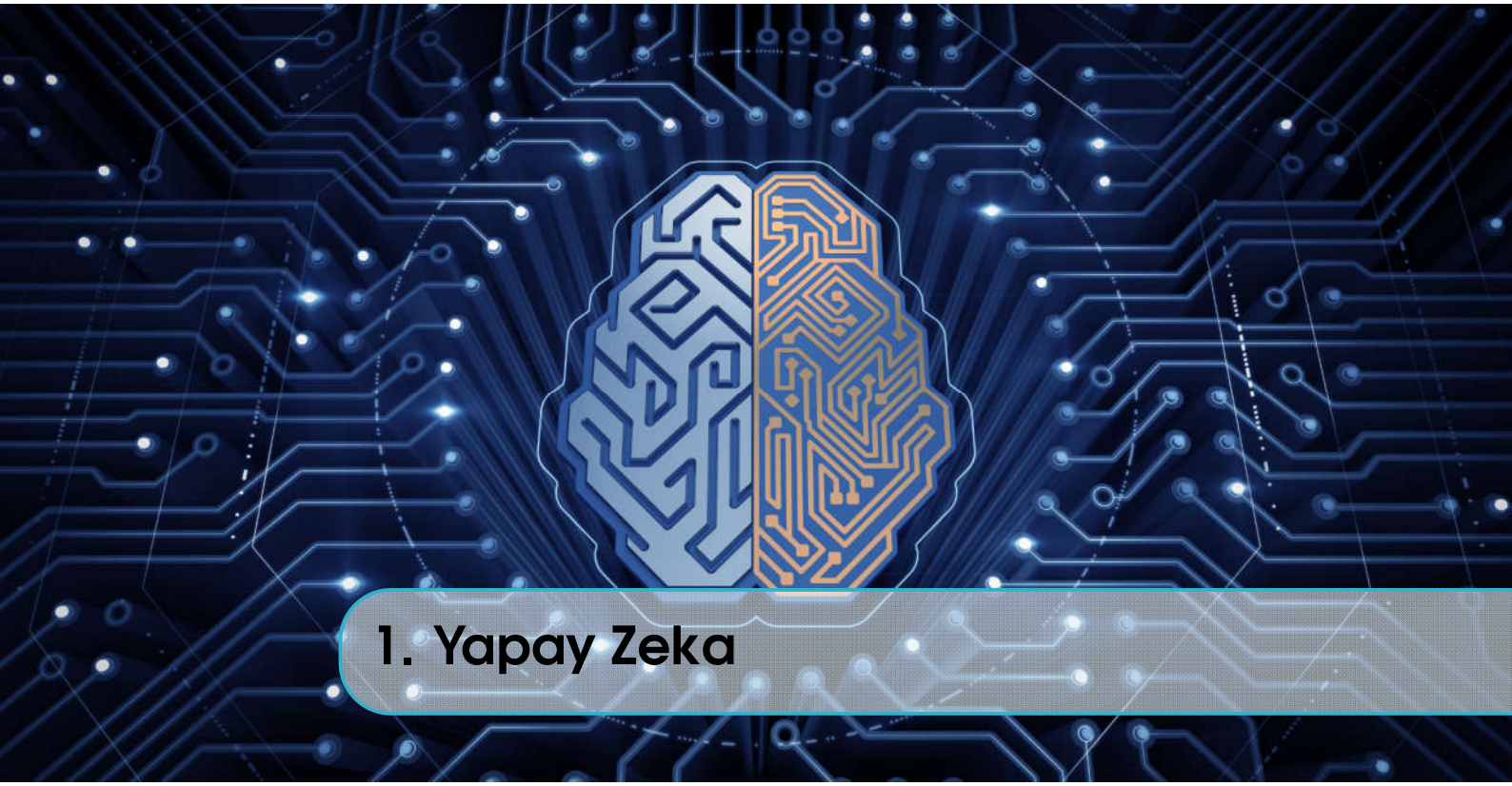
Teknolojinin gelişmesiyle ortaya çıkan yeni sanayi devrimi Endüstri 4.0 ve bunun en önemli bileşeni olan yapay zeka teknolojisine hakim olmak ülkelere muhakkak güç katacaktır. Endüstri 4.0 devrimini kaçırmadan adapte olabilmek ülkemize yapabileceğimiz en büyük hizmettir. Yapay zeka yavaş yavaş teknolojinin ortasına yerleşmeye başladı. Sistemlerin karar verebilmesi için algoritmalar yerine yapay zeka uygulamaları kullanılması kaçınılmaz. Sizlerin bu teknolojiye kendinize adapte etmeniz ve hızla çalışmalar yapmanız gerekmektedir. Bu yüzden, bu kitap yapay zeka alanında çalışmaya başlayanlar için önemli bir rehber niteliğindedir.

Kitaptaki her bölüm birbirinden bağımsız olarak okunabilmektedir. Kitabın 1. bölümünde yapay zeka tarihi hakkında kapsamlı bilgi verilmiştir. 2. bölümde gerçek zamanlı çalışan yapay zeka tabanlı sistemlerde yapay zekanın rolü anlatılmıştır. 3. bölümde bulunduğunuz durumlar hakkında karar ve bilgi vermek için kullanılabilecek bulanık mantık yöntemi anlatılmıştır. 4. bölümde uygulamalarınızda kullanılabileceğiniz açık kaynak kodlu makine öğrenmesi kütüphanelerine yer verilmiş, 5. bölümde ise açık kaynak kodlu makine öğrenmesi kütüphanelerinden olan Python ile görüntü işleme uygulamasına yer verilmiştir. 6. bölümde akıllı binalar hakkında anlatım gerçekleştirilmiştir. 7. bölümde belli kurallara göre en iyi sonucu bulmanıza yardımcı olacak genetik algoritmalara yer verilmiştir. 8. bölümde sınıflandırma çalışmalarında yaşanan dengesiz veri sınıflandırma problemine uygun bir çözüm sunulmuştur. 9. bölümde ise nesne ağırlık hesabının yapılmasına dair uygulama anlatılmıştır. Son olarak 10. bölümde ise kaba kümeleme hakkında bilgi verilmiştir.

Yapay zeka alanında çalışmaya başladıysanız hayal kurmaya devam edin. Çünkü bu alanda her şey bir hayal ile başladı. Bu kitap sadece size ufuk açma mahiyetinde bilgi vermektedir. Hayal kurup gerçekleştirmek sizin elinizde.

Hayallerinizin peşinden koşabilmeniz dileğiyle.

Editörler
Ekim 2017



1. Yapay Zeka

Yapay Zeka

Yrd. Doç. Dr. Zerrin Ayvaz REİS

1.1 Giriş

Bilgisayarların hayatımıza girmesi ile insanoğlunun milattan önceki çağlarda başlayan işleri mekanik araçlarla otomatikleştirme isteği yapay zeka kavramının ortaya atılmasına neden olmuştur. Sanayide bilgisayar teknolojisinin bant üretim süreçlerinde kullanılmasıyla da robot kavramı kullanılmaya başlanmıştır.

Yapay zekanın temellerinin; felsefe (milattan önce 428'den günümüze), matematik (8. yüzyıldan günümüze), psikoloji (1879'dan günümüze), bilgisayar mühendisliği (1940'dan günümüze) ve dilbilim (1957'den günümüze) disiplinlerinden oluştuğu görülmektedir (Russel&Norvig; 1995).

Yapay zeka kavramının geçmişi görüldüğü gibi çok eskilere dayanmaktadır. Fikir babası, "Makineler düşünebilir mi?" sorunsalını ortaya atarak makine zekâsını tartışmaya açan Alan Mathison Turing'dir. II. Dünya savaşı sırasında kripto analizi gereksinimleri ile üretilen elektromekanik cihazlar sayesinde bilgisayar bilimi ve yapay zekâ kavramları doğmuştur. Enigma makinesinin şifre algoritmasını çözme amacı ile başlatılan çalışmalar, Turing'in prensiplerini oluşturduğu bilgisayar prototipleri olan Heath Robinson, Bombe Bilgisayarı ve Colossus Bilgisayarları, Boole cebirine dayanan veri işleme mantığı ile Makine Zekâsı kavramının oluşmasına sebep olmuştur (kaynak: Anonim-1, n.d.) ve bu da bilgisayarların hayatımıza girmesinin temelini oluşturmuştur.

Bilindiği gibi bilgisayarlar insan beyni örnek alınarak geliştirilmiştir. Bilgisayarların gerek bilimsel anlamda gerek sanayide üretim sürecinde gerekse günlük hayatımızda kullanılmasındaki amaç, işlemlerin daha hızlı ve hatasız yapılmasının sağlanmasıdır ve bu gerçekleşmektedir. Bunun için verilerin bilgisayara uygun giriş birimleri ile girilmesi bu verilerin bilgisayarda işlenmesi ve istediğimiz sonuçları istediğimiz formatta bize sunması yeterlidir. Veriye uygun giriş birimleri derken; ses verileri için mikrofon, fotoğraf ve resim gibi görsel veriler için uygun tarayıcılar, alfabetik ve sayısal veriler için de ekran, klavye veya sanal klavye gibi giriş birimleri kastedilmektedir. Giriş birimleri ile bilgisayara girilmiş olan veriler işlenmesi önceleri sadece bu verilerin nasıl işleneceğini

ele alan komutlardan oluşan programlar olmuşken, artık günümüzde komutları sesli olarak da verebiliyoruz. İşlenmiş verilerden elde edilen sonuçlar sesli ve/veya zengin içerikli grafik ekran gibi çıkış birimlerinden sunulmaktadır.

Bilgisayar teknolojisindeki özellikle programlama dillerindeki gelişmelere baktığımızda; birinci kuşak programlama dillerinin bilgisayara doğrudan komut vermek üzere kullanılan "makine dili" olduğunu görmekteyiz. İkinci kuşak programlama dillerinin artık sadece makinenin anlayacağı bir kodlama biçimi olmak yerine, daha çok insanoğlunun konuşma diline yakın komutlar olduğunu ve bu şekilde hazırlanmış programların bilgisayarın anlayacağı hale derleyiciler (compiler) vasıtasıyla dönüştürüldüğünü görmekteyiz. Üçüncü kuşak programlama dillerinin ise artık tek düze yazılmış komut satırları olmak yerine modüler yapıda hazırlanmış, bir program içinde defalarca tekrarlanan komut satırları yerine, ihtiyaç duyulduğunda o işlevi yerine getirecek modülleri çağıran programlama dilleri olduğunu görmekteyiz. Dördüncü kuşak programlama dilleri programlama dilleri dediğimiz programlar ise program yazan programları üretme isteği ile bilgisayar bilimleri alanında yerini CASE (Computer Aided Software Engineering) teknolojisi ile bulmuştur. Bunlar yakın zamana kadar her biri birer hayal olmaktan öteye gitmeyen düşüncelerdi, teknolojideki gelişmelerle birer birer gerçekleştiğine şahit olmaktayız.

Yapay zeka konusundaki çalışmalar 1960'lardan beri gündemde olmasına karşın yapay zeka uygulamalarının güçlü bilgisayarlara ihtiyaç duyması nedeniyle araştırmaların hızlı ilerlemesine engel olmuştur. Ancak günümüzde bilgisayar teknolojisinde yaşanan gelişmelerin sağladığı ucuz ve güçlü bilgisayarlar sayesinde yapay zeka alanında büyük ölçekli araştırma yapabilmek ekonomik açıdan mümkün hale gelebilmiştir. Bunun sonucu olarak, yapay zekanın bir alt alanı olan uzman sistemler (expert systems) konusunda daha şimdiden önemli gelişmeler sağlanmış olup, iş aleminin karar verme sürecinde uzman sistemlerden önemli ölçüde yararlandığı gözlenmektedir (Sönmez, n.d.).

Bu çalışmada yapay zeka kavramının tanımı, amacı, tarihçesi ve güncel gelişmeler üstünde durularak, hayatımıza ne ölçüde girdiğine değinilecektir.

1.2 Tanım

Farklı uygulama alanları olan yapay zeka kavramının uygulama alanlarına bağlı olarak pek çok tanımı vardır. Bundan önce zekanın tanımı üstünde durmak faydalı olacaktır.

Zeka kavramının sözlük anlamına baktığımızda TDK (Türk Dil Kurumu) tarafından "İnsanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma yeteneklerinin tümüdür." (TDK, n.d.) olarak tanımlandığı görülmektedir.

Lenat ve Feigenbaum (1991) zeka için "Karmaşık bir problemi çözüm arama alanını daraltarak kısa yoldan çözmek için gerekli bilgileri toplayıp birleştirme kabiliyetidir" ifadesini kullanmıştır. Uğur ve Kınacı (2006) yapay zekayı Lenat ve Feigenbaum (1991)'ın tanımına bağlayarak; "bu özelliklere sahip organik olmayan sistemlerdeki zekadır." ifadesiyle tanımlamıştır.

Yapay zekanın bazı tanımları ve karakteristikleri (Bakınız Şekil 1) de karar verme ve problem çözme üzerinde yoğunlaşmıştır.

Sembolik İşleme: Uzmanlar, yapay zekaya uygun tipte olan problemleri çözmek, problem içeriklerini tanımlamak için sembol kullanırlar ve bu içerikleri işlemek için değişik stratejiler ve kurallar uygularlar. Waterman'ın (1986) yapay zeka yaklaşımı bilgiyi, problem konseptlerini temsil eden semboller kümesi olarak tanımlar. Yapay zekanın teknik dilinde sembol gerçek dünyanın bazı içeriklerini temsil eden karakterler kümesidir.

Şekil 1.1: Yapay zekanın karakteristikleri

Bulgusallık: Bulgusallık, kurallar yumağı olarak, tanımda yapay zekanın anahtar elemanı olarak kullanılmıştır. "Yapay zeka, bilgisayar biliminin, bilgiyi rakamlardan ziyade sembollerle temsil etme yöntemleri ve kurallar yumağını içeren bulgusal yapıda veya bilgiyi işleme metodları ile çalışan bilim dalıdır" (Copeland, 2017).

Anlam Çıkarma: Yapay zeka, makinenin sebep bulma yeteneği sergilemesini gerektirir. Sebep bulma, bulgusal yada diğer arama yaklaşımlarını kullanarak, olaylardan ve kurallardan anlam çıkarma sürecinden oluşur. Bu süreç üzerinde semboller üzerinde gerçeklenen en basit şekildeki örüntü uydurma ve tanıma (pattern matching, recognation) bu işin esasını oluşturmaktadır. Yapay zeka, bu yaklaşımını uygulayarak anlam çıkarmada eşsizdir.

Russel & Norvig (1995) yapay zekayı Şekil 2'de görüldüğü gibi sekiz (8) farklı tanımdan yola çıkarak iki ana boyuta göre sınıflandırır.

Şekil 1.2: Yapay zeka tanımlamaları ve sınıflandırılması

Biri; düşünce süreçleri ve mantıkla ilgili, diğeri kişiler ve davranışları ele almakla ilgilidir. Detaylandırıldığında;

- İnsan gibi davranmak: Turing test yaklaşımı ile
- İnsan gibi düşünmek: Bilişsel modelleme yaklaşımı ile
- Akılcı düşünmek: Düşünce kanunlarının yasaları ile
- Akılcı davranmak: Akılcı ajan yaklaşımı ile

açıklanmaktadır.

Yapay zeka, insanlarda zeka ile ilgili zihinsel fonksiyonları bilgisayar modelleri yardımıyla inceleyip bunları formel hale getirdikten sonra yapay sistemlere uygulamayı amaçlayan bir araştırma alanıdır. "Yapay zeka terimi ilk olarak önemli yapay zeka programlama dillerinden biri olan LISP'i geliştiren ve yapay zeka alanındaki öncülerden biri olan John McCarthy tarafından 1956 yılında ortaya atılmıştır" (Aktaran: Russell & Norvig, 1995, s. 17-18).

Yapay zeka, bir bilgisayarın veya kontrol altındaki bir robotun normal bir zeki canlı gibi davranma yetisidir (Akpınar, 2015). Genelde bu konu üzerindeki çalışmalar insanı temel alarak benzer yapay zeka üretmeyi amaçlamaktadır.

1.3 Yapay Zekanın Amaçları

Yapay zeka çalışmalarının amacı, insan zekasını örnek alarak, insan zekası gerektiren görevleri yapabilecek makinalar yapmaktır. Yani şu anda insanların bilgisayarlardan daha iyi yaptığı şeyleri bilgisayarların daha iyi yapmasını sağlama çalışmasıdır. Genel olarak yapay zekanın amacı üç ana başlık altında toplanabilir:

1. Makinaları daha akıllı hale getirmek:
   - Geleceğin bilgi toplumunun kurulmasında önemli rol oynayacak 'genel bilgi sistemleri' geliştirmek,
   - Öğrenme metotlarını formel hale getirmek ve bilgisayarlarda bilgi sistemleri halinde uygulamak,
   - Belli bir uzmanlık alanı içindeki bilgileri bir 'bilgi sistemi' veya 'uzman sistem' halinde toplamaktır.
2. Zekanın ne olduğunu anlamak:
   - İnsan beyninin fonksiyonlarını bilgisayar modelleri yardımıyla anlamaya çalışmak,
   - İnsanların sahip olduğu zihinsel yetenekleri, bilgi kazanma, öğrenme ve buluş yapmada uyguladıkları strateji, metot ve teknikleri araştırmaktır.
3. Makinaları daha faydalı hale getirmek:
   - Yapay zeka iş yardımcıları ve 'zeki robot timleri' geliştirmek,
   - İnsanlarını bilgisayar kullanımını kolaylaştıracak insan/bilgisayar ara birimleri geliştirmek,
   - Bilimsel araştırma ve buluşlarda faydalanmak üzere, 'araştırma yardımcıları' geliştirmektir.

Birçok davranış türü, zekanın işaretleri olarak kabul edilebilir. Aşağıda bunun tipik örnekleri görülmektedir.

- Tecrübelerden öğrenmek,

- Karışık ve zıt mesajlardan anlam çıkarmak,
- Yeni bir duruma başarılı ve çabuk bir şekilde cevap vermek,
- Problemlerin çözümünde muhakeme yeteneğini kullanmak,
- Bilgiyi anlamak ve kullanmak,
- Alışık olunmayan, şaşırtıcı durumların üstesinden gelebilmek,
- Düşünmek ve muhakeme etmek.

Yapay zeka programları her geçen gün daha ileriye gitmekte ve insan zekası gerektiren bazı işlere rehberlik etmekte oldukça faydalı olmaktadırlar.

1.4 Yapay Zekanın Tarihçesi

Modern yapay zekanın başlangıcının izlerini, klasik filozofların insan düşünce sistematiğini simgesel sistem olarak tanımlama girişiminde görmek mümkündür, fakat, yapay zeka alanı 1956'ya kadar resmi olarak oluşturulmamıştı. Yapay zeka terimi ilk defa 1956'da Hanover, New Hampshire, Dartmouth College'da yapılan bir konferansta ortaya atılmıştır (Durakcan, 2015).

Rönesans döneminde (14-17. yüzyıllar arası) Leonardo da Vinci otomatik makinalar, daha sonra Fransız matematikçisi Pascal'ın mekanik bir hesap makinesi konusunda çalışmıştır. 19. yüzyılda ise, İngiliz bilgini Babbage, Fransız mühendis Jacquard'ın dokuma tezgahları için icad ettiği bir tekniği kullanarak ilk programlanabilir mekanik bilgisayar olan Analytical Engine'i geliştirmeye çalışmıştır (Russell & Norvig, 1995), fakat o zamanki teknoloji yeteri kadar hassas olmadığı için, projesi yarım kalmıştır. Babbage'ın çalışmalarının sürdüğü sıralarda, bir başka İngiliz matematikçi George Boole iki tabanlı sayılarla ortaya konmuş Boole Cebri ile modern mantığın temellerini atmıştır. Sembolik mantığın 19. yüzyılın ikinci yarısında Boole ve daha sonra Frege'nin çalışmalarıyla başlayıp yirminci yüzyılda Russell ve Whitehead'in çalışmalarıyla gelişmesi de bilgisayar bilimlerinin ve yapay zekanın gelişmesinde önemli rol oynamıştır. İkinci Dünya Savaşı öncesinde Turing'in hesaplanabilirlik teorisi üzerine yaptığı çalışmalar zeki sistemler üzerine yapılan önemli çalışmaların başında gelmektedir(Kocabaş & Langley, 2000).

1940'larda sibernetik alanında yapılan çalışmalar insan ve makine arasındaki birçok paralelliği ortaya çıkarmıştır. Wiener (1948)'in sibernetik üzerine yayınladığı yazısında, insan beynindeki tüm fonksiyonların elektronik olarak kopyalanmasının mümkün olduğunu iddia etmektedir. Sonraki yıllarda sibernetik, bilgi teorisi, geri beslemeli kontrol sistemleri ve elektronik bilgisayarlarla ilgili kavramları birleştiren önemli bir araştırma alanı haline gelmiştir. Günümüz bilgisayarlarının temelini oluşturan özellikler 1940'larda John von Neumann tarafından açıklanmış ve modern bilgisayarların mimarisi tasarlanmıştır (Poundstone, n.d.).

1950'lerde bilgisayarların ticari şirketlerde kullanılmaya başlaması ile yapay zeka ayrı bir araştırma alanı olarak ortaya çıkmıştır. Claude Shannon ve Allen Newell'in geliştirdikleri satranç programları ve diğer oyunları oynayan programlar ortaya çıkmıştır (Kocabaş & Langley, 2000). Otomatik çeviri programları üzerine çalışmalar da bu yıllarda başlamıştır.

1956 yılında IBM tarafından düzenlenen konferansa yapay zekanın öncüleri olarak sayılan Marvin Minsky, Allen Newell, Claude Shannon ve Herbert Simon katılmış ve aynı toplantıda John McCarthy bu alandaki çalışmalara "yapay zeka" adını vermiştir. Allen Newell ve Herbert Simon

daha sonra ilk teorem ispatlayıcısı olan Logic Theorist programını geliştirdiler (Russell & Norvig, 1995; Aktaran:Kocabaş, 2013).

1950'lerin sonlarına doğru çalışmalar şekil tanıma ve kendi kendine adapte olan sistemler üzerine yoğunlaştı. Aynı dönemde McCarthy (1960) MIT'de önemli yapay zeka programlama dillerinden biri olan Lisp'i geliştirdi.

1965 yılında DENDRAL programı çalışması J. Lederberg, Edward Feigenbaum ve Carl Djerassi tarafından Stanford Üniversitesi'nde başlatıldı. DENDRAL programı, ilk bilgi tabanlı uzman sistem olarak geliştirildi (Feigenbaum, Buchanan & Lederberg, 1971; Lindsay, Buchanan, Feigenbaum & Lederberg, 1980).

1965 yılında Weizenbaum ELIZA adındaki ilk psikiyatrist programını geliştirdi (Russell & Norvig, 1995). Basit fakat etkileşimli bir programdı. 1968'de ise bir çok matematik problemini çözen MACSYMA programının çalışmaları MIT'de Carl Engelman,William Martin ve Joel Moses tarafından başlatıldı (Firebaugh, 1988).

1972'de; bu gün yapay zeka çalışmalarında kullanılan önemli bir programlama dili olan Prolog (PROgramming LOGic), Alain Colmerauer tarafından geliştirildi (Colmerauer, 1990). İlk Prolog derleyicisi 1977'de D. Warren tarafından geliştirildi (Webber, n.d.).

1981 yılında Japonlar "5. Nesil Bilgisayar Projesi"nde Prolog programlama dilini kullandılar. Bilgi tabanlı sistemlerin başarılı uygulamaları daha sonra genel bilgi sistemlerinin geliştirilmesine öncülük etti ve 1983'de Amerika' da CYC (Encyclopaedia), 1984'de Japonya'da EDR (Elektronik Dictionary) Projeleri başlatılmıştır (Kocabaş & Langley, 2000).

Kocabaş (2013) son 60 yıldaki yapay zeka çalışmalarını üç safhada özetlemiştir:

1. Algılayıcılar ve Yapay Sinir Ağları üzerinde yapılan çalışmalar. Bu safhadaki çalışmalar 1940-1965 yılları arasında ve 1982'den günümüze kadar devam etmiştir.
2. Sembolik Yapay Zeka dönemi 1965-1975 yılları arasında ön plana çıkmıştır.
3. Bilgi Tabanlı Sistemler alanındaki çalışmalar 1975'ten günümüze kadar gelmektedir.

Social BusinessTR tarafından hazırlanmış olan yapay zekanın hayatımıza girişi ile ilgili aşağıdaki kronolojik sıradaki bilginin yararlı olacağı düşünülmektedir (Anonim, 2017).

MS 1.yy

• İskenderiyeli Heron adına otomotlar dediği, su ve buharla çalışan mekanik düzenekler yaptı.

1206

• Artuklu sarayında yaşayan Ebu'l İz El Cezeri, suyla çalışan otomatlar yaptı.

1822-1859

• Charles Babbage ve Ada Lovelace, programlanabilir mekanik hesap makineleri yaptılar. Ada Lovelace delikli kartlar kullanarak Babbace'ın makinelerini yeniden programlayan ilk kadın programcıdır.

1923

• Karel Capek'in R.U.R. adlı tiyatro oyununda "Robot" sözcüğünü ilk kez telaffuz edildi.

1950

• Isaac Asimov, bilim kurgu öykülerinden oluşan "I, Robot" adlı eserini yayınladı.

• Alan Turing, Turing testini "Computing Machinery and Intelligence" adlı makalesinde yayınladı.

1951

• İlk yapay zekâ programları Ferranti Mark 1 adlı aygıt için Manshester Üniversitesi'nde yazıldı.

1956

• Darthmouth Konferansı'nda "yapay zekâ" terimi kullanarak kavramın adı kondu ve yapay zeka alanı tanıtıldı.

• Bugün Carnagie Mellon Üniversitesi olarak bilinen Carnagie Teknoloji Enstitüsü'nde Allen Neweell, J.C. Shaw ve Herbert Simon'un yazdığı Logic Theorist (Mantık Teorisi-LT) adlı program tanıtıldı. Matematik problemleri çözmeye yarayan bu program ilk yapay zekâ programı olarak kabul ediliyor.

1958

• MIT'den John Mc Carty LISP dilini yarattı.

1962

• Endüstüriyel robot üreten ilk firma olan Unimation kuruldu.

1965

• ELİZA adlı yapay zekâ programı yazıldı.

1966

• Stanford Üniversitesi'nde ilk hareketli robot "Shakey" üretildi.

1968

• Arthur C. Clarke'ın romanından uyarlanan "2001: A Space Odyssey" Stanley Kubrick filmi izleyicilerine ünlü bilgisayar Hal 9000'i sundu.

1972

• Prolog, Alain Colmerauer tarafından geliştirildi

1974

• 1974 yılından erken dönem 80'lere kadar azalan fonlama dönemleri ve ilginin azalması ile yapay zekanın ilk kış dönemi başladı.

1977

• İlk Prolog derleyicisi D. Warren tarafından geliştirildi.

1978

• Herbet Simon, yapay zeka alanındaki önemli adımlardan biri olan Sınırlı Rasyonalite Teorisiyle ekonomi dalından Nobel Ödülünü kazandı.

• Orjinal Battlestar Galactica dizisinde Cylons denen savaşçı robotlar kullanıldı.

1984

• ilk "Terminator" filmi Skynet yapay zekası tarafından yönetilen ve yakın geleceği ele geçiren katil makineler tasvir edildi.

1987

• Star Trek: The Next Generation dizisi bilinçli android Data'yı sundu.

1993

• MIT'de Cog adlı insan biçimli bir robotun yapımına başlandı.

1997

• IBM'in Deep Blue bilgisayarı satrançta dünya şampiyonu Gary Kasparov'u yendi.

1998

• Tiger Electronics firması evlere girmeyi başaran ilk yapay zekâ oyuncağı olan Furby'yi piyasaya sürdü.

2000

• Cynthia Braezeal, "Kısmet" adını verdiği robotu tanıttı. Bu robot karşısındaki kişiyle konuşurken mimik ve yüz hareketlerini kullanabiliyor.

2001

• Steven Spielberg, orjinali Stanley Kubrick tarafından geliştirilen ve robot bir çocuk hakkındaki "Artificial Intelligence: AI" filmini yaptı.

2005

• Stanford aracı, 211 kilometre ile çölü otonom şekilde geçerek Darpa Büyük Yarışı'nı kazandı.

• Futurist ve mucit Ray Kurzwell, 2045'te yapay zekanın insan beyninin önüne geçeceğini öngördüğü "The Singularity" teorisini sundu.

• Honda Firması Asimo adını verdiği yapay zekaya sahip insansı robotunu tanıttı. ASIMO o güne kadar yapılmış olan en becerikli insansı robottu.

2010

• Asimo zihin gücüyle hareket ettirildi.

2011

• IBM Watson, "Jeopardy!" adlı bilgi yarışmasını, yarışmanın önceki şampiyonları olan Brad Rutter ve Ken Jennings'i yenerek kazandı.

• Apple, akıllı kişisel asistan Siri'yi iPhone 4S ile tanıttı.

2012

• Google'ın geliştirdiği yapay beyin, YouTube'da izlediği milyonlarca görüntü arasında ilk olarak kedileri keşfetti.

2013

• "Her" adlı filmde Joaquin Phoenix, bilgisayar işletim sistemi olan ve Scarlett Johansson'un seslendirdiği yapay zekaya aşık bir karakteri canlandırdı.

2014

• "Transcendence" filminde Jonny Depp, zihnini bilgisayara aktaran ve süper zeka olarak geliştiren yapay zeka araştırmacısını canlandırdı.

• Chatbot Eugene Goostman'ın Turing testini geçtiği belirtildi.

2015

• Natal Projesi (X-box)– Milo & Claire tanıtıldı.

Yapay zeka; Sibernetik Yapay Zeka ve Sembolik Yapay Zeka olarak ikiye ayrıldıktan sonra, çalışma sonuçlarının önemli eksiklikler nedeniyle pekte başarılı olmadığı görülmüştür. Bu iki yaklaşımın çökmesi Uzman Sistemlerin oluşumuna neden olmuştur. Uzman sistemler yapay zekanın tüm sorulara cevap vermesini değil de belirli bir konu üzerinde cevap vermesini temel almaktadır (Akpınar, 2015).

1.5 Yapay Zekanın Araştırma Alanları

Yapay zeka çalışmaları sadece bilgisayar bilimlerinde değil, oyun, matematiksel teoremlerin otomatik olarak ispatlanmasında, doğal dil anlama ve çeviri işlemlerinde, görüntülerin işlenmesinde, genel bilgi sistemlerinde, makine öğrenmesinde, bilgi tabanlı sistemlerde, veri madenciliğinde, robotik gibi farklı alanlarda gerçekleştirilmektedir (Kocabaş, 2013). Her geçen gün farklı alanlarda yapay zekanın kullanıldığını görmekteyiz.

1.5.1 Oyunlar

Satranç, dama, tavla gibi oyunlar araştırmacılar için yapay zekanın ilk zamanlarından bu yana tercih edilen bir alan olmuştur. Başlangıçta kısıtlı bir zamanda çok sayıdaki çözüm yolunu göz önünde bulundurma becerisi üzerine kurulmuş olan sistemler artık tecrübelerden yararlanılan bilgi stratejileri kullanılarak genel çözüm arama kavramlarına dönmektedir. Bu alanda yapılan çalışmalar Yapay Hayat (Artificial Life) adında yeni bir araştırma alanının ortaya çıkmasına yol açmıştır.

1.5.2 Otomatik Teorem İspatlama

1950'lerde yapay zekanın erken döneminde başlayan bu alandaki çalışmalarla özellikle sembolik mantıkta ispatlanan teoremlerin daha basit ispat yollarının bulunması konusunda kayda değer

sonuçlar elde edilmiştir. Bu çalışmaların sonuçlarından biri olarak da, sembolik mantığa dayanan güçlü bir yapay zeka dili olan Prolog programlama dili ortaya çıkmıştır. Günümüzde paralel Prolog derleyicileri geliştirme çalışmaları halen devam etmektedir (Kale & Ramkumar, 1990).

1.5.3 Doğal Dil İşleme

Bu alandaki çalışmalar, otomatik tercüme, doğal dilde yazılmış metinlerin açıklanması ve üretilmesi ve konuşmaların otomatik işlenmesi gibi faaliyetleri kapsar. Son yıllarda CYC ve EDR gibi genel bilgi sistemleri kullanılarak tercüme sistemlerindeki doğruluk oranı arttırılmaktadır. Makine tarafından bir cümlenin anlaşılması, birçok bilgiyi devreye sokan bir süreçtir. Fakat kimi zaman gürültüler ve akustik değişkenlik benzeşen sinyalin işlenmesini zorlaştırır. Günümüzde deneysel insan-makine diyalogu sistemleri geliştirilmiş durumdadır. Bununla birlikte, bu sistemler, problemin olağanüstü zorluğu yüzünden özel kelimelerle sınırlı bir kelime dağarcığını tanımakla sınırlı kalırlar.

1.5.4 Görüntü İşleme

Görme bir makinenin çevresini fark etmeye yönelik başka bir özelliğidir. Görme probleminin basitleştirilmesi, basit şekillerin algoritmik şekil tanıma metotları yardımıyla tanımlanmasından ibarettir. Bunlar bir metin içindeki harfler veya bir uydu resmi üzerindeki özel bölgeler olabilir. Bir sahnenin veya görüntünün gerçek olup olmadığının anlaşılması, tıbbi teşhis amacı ile radyolojik görüntülerin açıklanması, basılı ya da el yazısı bir metnin anlaşılması robotbilime aittir. Görüntülerin işlenmesi yapay zekanın endüstriyel alandaki ilk uygulamalarından biridir.

1.5.5 Robotik

Robotbilim geniş bir alana yayılmış durumdadır. Özellikle sanayide iş otomasyonu etkinliklerini kapsamaktadır. Varolan robotların büyük bölümü işleri sıra ile durmaksızın tekrarlar ve zeki davranış göstermezler. Buna karşın yeni kuşak robotlar, giderek çevrelerini algılamaya ve hareketlerini planlamaya yönelik zeka yeteneği ile donatılmaktadır. Robotik alanındaki çalışmalar çoğunlukla, robot görmesi, görev planlama, robot timleri, mikro robotlar ve robot kolonileri üzerinde yoğunlaşmaktadır (Kocabaş, 2013).

1.5.6 Bilgi Tabanlı Sistemler

Bu alandaki çalışmaları; Bilgi Gösterimi, Bilgi Tabanlı Simülasyon, Uzman Sistemler ve Genel Bilgi Sistemleri olmak üzere dört alt başlık altında değerlendirebiliriz:.

Bilgi Gösterimi

Yapay zekada bilgi gösterim metotlarını üç seviyede sınıflandırılmaktadır. Bilgi düzeyi, sembol düzeyi, aygıt düzeyi. bilgi düzeyi metotlarda bilgi, kurallar, mantık yapıları, çerçeveler, senaryolar ve vaka kayıtları şeklinde gösterilmektedir. Sembol düzeyi metotlarda ise bilgi, vektörler ve matris yapıları içinde gösterilmektedir. Aygıt düzeyinde bilgi, bir ağ yapısı içinde gösterilmektedir.

Uzman Sistemler

Tasarım, planlama, teşhis, özetleme, kontrol etme ve tavsiyede bulunma gibi konularda insan uzmanların yaptıkları faaliyetleri, otomatik olarak uygulamak üzere geliştirilen bilgisayar programlarıdır. Uzman sistemler, sınırlı bir alan içinde uzmanlık bilgisini depolayabilir, mantıksal sonuçları takip ederek problemin çözümüne ulaşabilir. İlk geliştirilen uzman sistemlerden biri 1970'lerde tıpta bazı hastalıkların teşhisini yapabilen MYCIN programıdır (Copeland, 2017). Günümüzde tıp-

tan mimarlığa, bankacılıktan kadar akla gelebilecek birçok alanda geliştirilmiş uzman sistemler bulunmaktadır.

Bilgi Tabanlı Simülasyon

Bu alanda kullanılmak üzere geliştirilen sistemler afet yönetimi, kriz yönetimi, stratejik planlama ve bazı askeri alanlarda uygulanmaktadır. Batı Avrupa ülkelerinin katıldığı EUCLID projeleri çerçevesinde Kocabaş, Öztemel, Uludağ ve Koç (1996) tarafından Tübitak MAM'da geliştirilen AISim sistemi hava muharebesi simülasyonu ortamında bir F16 uçağını hiçbir pilot veya operatör müdahalesi olmadan yönetebilen bir sistemdir.

Genel Bilgi Sistemleri

Uzman sistemlerin en zayıf tarafı insanların sahip olduğu sağduyu bilgisine ve genel bilgilere sahip olmamasıdır. Bundan dolayı uzman sistemlere, insan uzmanların aksine kendi uzmanlık alanlarının biraz dışındaki problemler verildiği zaman ya çözüm üretemezler veya anlamsız bir çözüm verirler. Uzman sistemlerin bu eksikliğinin giderilmesi için insanların sahip oldukları genel bilgileri ve sağduyu bilgilerini de taşıyan genel bilgi sistemleri geliştirilmeye başlanmıştır.

1.5.7 Makina Öğrenmesi

İlk çalışmalar "algılayıcılar" (perceptrons) adı verilen, aygıt düzeyinde basit sistemler üzerinde başlamıştır. Daha sonra sembol düzeyi öğrenme metotları geliştirilmeye başlanmıştır. 1970'lerin sonlarına doğru bilgi tabanlı sistemlerin ortaya çıkmasıyla bilgi düzeyi öğrenme metotları geliştirilmeye başlanmıştır.

Aygıt düzeyinde öğrenme özelliğine sahip sistemlerde başlangıçta çok az bilgi ve yapılanma vardır. Öğrenme, çıkış sinyallerindeki geri beslemeyle ağ üzerindeki bağlantı ağırlıklarının değiştirilmesi ile gerçekleşir. Öğrenen sistemin amacı optimum düzeye ulaşmaktır. Sembol düzeyinde öğrenme özelliğine sahip sistemler yüklemler mantığını ve sembolik matematiği kullanabilen sistemlerdir. Bunlar yeni kavramları pozitif ve negatif örneklerden öğrenirler.

Bilgi düzeyinde öğrenme özelliğine sahip sistemler "bilgi kuvvettir" prensibine dayanarak geliştirilmişlerdir. Bu anlayışa göre bir sistem ne kadar bilgiye sahipse, o kadar çok şey öğrenebilir. Bu çerçevede geliştirilen sistemler tümdengelim (dedüktif), tümevarım (indüktif), ve analojik öğrenme özelliklerine sahiptirler.

1.5.8 Makina Buluşları ve Veri Madenciliği

Makina öğrenmesi üzerine yapılan çalışmalar daha sonra makina buluşları alanına doğru gelişmeye başlamıştır. Büyük veri tabanlarından tümdengelim, tümevarım ve analojik öğrenme metotlarıyla yeni bilgiler ortaya çıkarılmıştır. İlk olarak tıp veri tabanlarındaki ilaç uygulamaları ve bunların etkileri üzerinden yeni tedavi yolları ortaya çıkarılmıştır. Bu çalışmalar daha sonra yeni bilimsel araştırmalardan elde edilen veriler üzerinde de uygulanmaya başlamıştır.

1.5.9 Bilimsel Buluşların Modellenmesi

Bu alanda geliştirilen sistemler bilim tarihinde gerçekleştirilmiş olan buluşları modellemek üzere geliştirilen programlardır. Bunlardan ilk geliştirilenler matematikteki buluşları modelleyen Automated Mathemetician (Lenat, 1979) ve klasik fizikteki buluşları modelleyen BACON (Langley, 1978) sistemleridir. Daha sonra 17-19 yüzyıllarda kimyadaki buluşları modelleyen GLAUBER, STAHL sistemleri (bakınız: Langley, Simon, Bradshaw & Zytkow, 1987) ve STAHLp sistemi (Rose & Langley, 1986) geliştirilmiştir. Bizim geliştirdiğimiz BR-3 (Kocabaş, 1991) ve BR-4 (Kocabaş &

Langley, 2001) sistemleri ise parçacık fiziğinde 1930-60 yılları arasındaki buluşların modellendiği sistemlerdir.

1.5.10 Bilimsel Araştırma Yardımcıları

Bilimsel buluşların modellenmesi alanındaki başarılar, günümüzde yapılan bilimsel araştırmalarda kullanılabilecek yardımcı programlar geliştirme çalışmalarına yol açmıştır. Bu alanda geliştirilen sistemlerden biri, nükleer astrofizikte yıldızlarda meydana gelen çekirdek reaksiyonlarını ve elementlerin sentez mekanizmalarını formüle edebilen ASTRA sistemi (Kocabaş & Langley, 2000; Kocabaş, 2001) diğeri katalitik kimyada metanol sentez mekanizmalarını formüle eden MECHEM (Valdes-Perez, 1995) sistemidir. Diğer bir sistem ise organik molekül sentezlerini formüle etmede kullanılan SYNGEN (Hendrickson, 1997) sistemidir. Bu çerçevede geliştirilen sistemlerin; araştırma amaçlarını formüle edebilme, deneyler önerebilme, hipotezler kurup test edebilme, çelişkileri çözümleyebilme ve açıklamalar yapabilme özelliklerine sahip olması beklenmektedir.

1.6 Sonuç

Bilgisayarlar; nesneler, olaylar ya da süreçler hakkında bilgi toplamak için kullanılabilir ve tabii ki bilgisayarlar çok büyük miktarda verileri insanlardan daha hızlı ve daha verimli işleyebilirler. Bununla birlikte insanlar, bir bilgisayar programında yapılması çok zor olan bazı şeyleri içgüdüsel olarak yaparlar, nitelikleri hissedebilirler ve değişik elemanların birbiriyle olan ilişkisini açıklamaya yarayan modelleri görebilirler ve ayırt edebilirler. Mesela gazete fotoğraflarını incelemek böyle bir şeydir, insanlar, bilinçli bir çaba göstermeden bu fotoğraflardaki yüzleri ve nesneleri ortaya çıkaran modelleri bulurlar. Benzer şekilde, insanların dünyada olup bitenleri hissetmesini sağlayan yollardan biri de karşılaştıkları nesnelere ve olaylara anlam vermesine yardımcı olan ilişkilerin ve modellerin farkına varmasıdır. Eğer bilgisayarlar da insanlar kadar (veya daha fazla) zeki olacaklarsa, insanlara çok doğal gelen nesneler, olaylar ve süreçler üzerinde düşünsel olarak bir araya gelebilmelidirler (Sönmez, n.d.).

İnsan ve makine zekası arasındaki farklar (Anonim-2, n.d.) üç maddede gösterilmektedir;

1. İnsanlar kalıplarla algılarken makineler kurallar ve verilerle algılarlar.
2. İnsanlar bilgiyi kalıplara göre depolar ve geri çağırır, makineler algoritmaları araştırarak yapar. Örneğin, deseni basit olduğu için 40404040 numarası hatırlamak, depolamak ve geri çağırmak kolaydır.
3. İnsanlar, bir kısmının eksik veya çarpıtılmış olsalar bile, komple nesneyi bulabilir; makineler doğru şekilde yapamazlar.

Modern bilgisayarların kullanılmaya başlanmasıyla birlikte insan ve makine zekası arasındaki farkları kapatma isteğiyle, bilgisayarlardan maksimum yararlanmak üzere çalışmalar gerek donanım gerekse yazılım alanında sürekli gelişim göstermektedir. Makineleri daha zeki yapma temel amacıyla başlayan yapay zeka çalışmaları hızla devam etmektedir.

ARPA Raporu'na göre (Grosz & Davis, 1994) yirmi birinci yüzyılda yapay zekanın etkili olacağı alanlar:

* Zeki simülasyonlar, ,
* Bilgi kaynaklarına ulaşım sistemleri,
* Zeki proje yardımcıları,

∗ Robot timleri olarak belirtilmiştir.

ve çalışmalar hız kesmeden devam etmiştir. Bugün geldiğimiz noktada hayatımıza girmiş yapay zeka ürünlerinin özellikleri, hizmet verme kapasiteleri üretici firmaları tarafından sürekli olarak geliştirilmektedir.

1. Apple Siri: temel düzeyde bir asistan olarak hizmet vermektedir.
2. Microsoft Cortana: öğrenen makinaların en temel özelliklerinden birisi olarak iyi bir örnektir.
3. Google Now: gerçek insan-makina diyalogları konusunda büyük bir adım olmuştur.
4. IBM Watson: özellikle sağlık alanındaki karmakarışık hastane kayıtlarını analiz ederek, mantıklı desenler keşfedebilen ve bunlardan öğrendiği sonuçları sunabilen bir ürün olarak hizmet vermektedir. Dahası, doktorların öngöremeyeceği kadar veriyi bir arada işleyerek, teşhis ve tanıda tavsiyelerde de bulunmaktadır.
5. IPsoft Amelia: Amelia'da, daha önceden saydığımız özelliklerin üzerine, bir de duygusal farkındalık eklenmiş bir üründür. İnsan gibi düşünebilen ve hisseden bir sanal asistan olarak hizmet vermektedir (Kara, 2014).

Uzmanlara göre 2050'yılında 1000 dolara alınan bir PC dünyadaki tüm insanların beyin gücünden daha fazla bir güce sahip olacaktır (Akpınar, 2015). Yapay zekanın diğer alt dalları olan Konuşma Sentezi, Konuşma Anlama, Örüntü Tanıma, Genetik Algoritmalar, Genetik Programlama, Bulanık Mantık konuları da araştırma alanı olarak ilgi odağı haline gelmektedir.

1.7 Kaynakça

Akpınar, M.Y. (2015), Yapay Zeka Ve Tarihi, Erişim: http://bit.ly/2gKUSzd, Erişim tarihi: 18.08.2017

Anonim-1, (n.d.), Yapay Zeka'nın Tarihçesi Ve Gelişim Süreci, Erişim: http://bit.ly/2x8nZm0, Erişim tarihi: 18.08.2017

Anonim-2, (n.d.), Artificial Intelligence - Intelligent Systems, Erişim: http://bit.ly/2geHBxR, Erişim tarihi: 18.08.2017

Anonim, (2017), Yapay Zekânın Modern Tarihi, SocialBusiness TR, Erişim: http://bit.ly/2xKVrn6, Erişim tarihi: 18.08.2017

Colmerauer, A., (1990), An introduction to Prolog III, Communications of the ACM, Volume 33 Issue 7, July 1990, pp: 69-90

Copeland, B.J., (2017), Artificial intelligence (AI), Encyclopedia Britannica, Erişim tarihi: 31.Ağustos.2017, Erişim: http://bit.ly/2gdF3Aj,

ÇMB (Evrim Ağacı), (Şubat 2015), Hayatlarımıza Giren 5 Yapay Zeka Ürünü: Sizin Yapay Zekanız Ne Kadar Zeki?, http://bit.ly/2yscpVX

Durakcan, Y.C. (22.06.2015), Yapay Zekanın Kısa Tarihçesi, Bilimfili, http://bit.ly/2geK8rS

Firebaugh, M.W. (1988). Artificial Intelligence: A Knowledge-Based Approach. Boston: PWS-Kent Publishing Co. ISBN-13: 978-0878353255, ISBN-10: 0878353259

Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: A case study using the DENDRAL program. In Machine Intelligence (Vol. 6).

Grosz, B., Davis, R., (1994), A Report to ARPA on Twenty First Century Intelligent Systems. AI Magazine, Fall 1994, s. 10-20.

Kale, L.V., Ramkumar, B., (1990), Implementation of a parallel Prolog interpreter on multiprocessors, PPOPP '90 of the second ACM SIGPLAN symposium on Principles & practice of parallel programming, ACM SIGPLAN Notices, Volume 25 Issue 3, Mar. 1990, pp: 99 - 108

Kara, M., (01.10.2014), Amelia: İnsan gibi düşünebilen sanal asistan, Erişim tarihi: 15.08.2017, Erişim: http://bit.ly/2kRB0Pf.

Kocabaş, Ş., (09.03.2013), Yapay Zeka | Amacı ve Tarihçesi | Gelecek'te Yapay Zeka, Erişim: http://bit.ly/2kTW1ss, Erişim tarihi: 15.08.2017

Kocabas, Ş. & Langley, P. (2000). Computer generation of process explanations in nuclear astrophysics. International Journal of Human-Computer Studies, 53, s.1149-1164.

Lenat, D.B., Feigenbaum, E.A., (1991), On the thresholds of knowledge, Proceedings IJCAI-87, Milan, Italy pp. 1173-1182, published in Artificial Intelligence, Volume 47, Issues 1–3, January 1991, pp 185-250, Elsevier.

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A. & Lederberg, J., (1980), Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project,New York: McGraw Hill, ISBN O-07-037895-9

McCarthy, J., (1960), Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I , Communications of the ACM, pp:184-195, Erişim: http://bit.ly/2ynpTmt, Erişim tarihi: 18.08.2017

Poundstone, W., (n.d.), John von Neumann AMERICAN MATHEMATICIAN, Erişim tarihi: 04.09.2017, Erişim: http://bit.ly/2gJWclS,

Russel, S.J., Norvig, P., (1995), "Artificial Intelligence A Modern Approach", Contributing writers: John F. Canny, Jitendra M. Malik, Douglas D. Edwards , Prentice Hall, Englewood Cliffs, New Jersey 07632, ISBN 0-13-103805-2, Erişim: http://bit.ly/2fbdJ50, Erişim tarihi: 04.09.2017

Sönmez, Ş., (n.d.), Yapay Zeka (ders notları), Erişim: http://bit.ly/2ieeRcT , Erişim tarihi: 18.08.2017

TDK, (n.d.), 2017, Zeka, Erişim: http://bit.ly/1CcStkR, Erişim tarihi: 04.09.2017

Uğur, A., Kınacı, A.C., (2006), Yapay Zeka Teknikleri ve Yapay Sinir Ağları Kullanılarak Web Sayfalarının Sınıflandırılması, inet-tr'06 - XI. Türkiye'de İnternet Konferansı, 21 – 23.Aralık.2006, TOBB Ekonomi ve Teknoloji Üniversitesi Ankara, Konferans Bildiri Kitabı, sayfa: 345-349

Waterman, D.A. (1986). A Guide to Expert Systems. Addison-Wesley, Reading, MA Webber, (n.d.), Prolog - a little more history 1, Erişim: http://bit.ly/2xJBlol , Erişim tarihi: 04.09.2017

Wiener, N., (1948), CYBERNETICS or control and communication in the animal and the machine, THE M.I.T. PRESS Cambridge, Massachusetts, ISBN:0-262-23007-0

1.8 Yazar Hakkında

Tüm öğrenim ve çalışma hayatı İstanbul Üniversitesinde geçmiştir. 1987 yılında Fen Fakültesi, 1990 yılında Sosyal Bilimler Enstitüsü Sayısal Yöntemler Anabilim Dalı'nda master, 1999 yılında Fen Bilimleri Enstitüsü Bilgisayar Bilimleri Yazılım Mühendisliği Anabilim Dalı'nda doktora derecesini almıştır. Halen Enformatik Bölümü ve Hasan Ali Yücel Eğitim Fakültesi BÖTE Bölümü'nde öğretim üyesi olarak çalışmaktadır. İNETD – İnternet Teknolojileri Derneği Yönetim Kurulu Üyesi ve TBD – Türkiye Bilişim Derneği Üyesidir.

2. Güvenlik Öncelikli Zeki Sistemler

Büyük Ölçekli Güvenliğin Öncelikli Olduğu Sistemlerde Gömülü Zeki Gerçek-Zamanlı Sistemlerin Rolü ve Karmaşıklık Ölçümü

Prof. Dr. Erman COŞKUN, Arş. Gör. Büşra ALMA

2.1 Giriş

1960'lardan başlayarak bilgisayarların işletmeler ve iş hayatı tarafından; önce hesaplama yapmak, ve veri saklamak, daha sonra veri işlemek ve veriyi bilgiye dönüştürmek ve daha sonra da karar alıcılar tarafından daha doğru ve daha hızlı karar alınmasını desteklemek amacıyla kullanılması bilgisayarların ve bilişim sistemlerinin işletme için önemini tarihsel gelişimde sürekli arttırmıştır. Günümüzde ise bilişim sisteminin ana bileşenleri olan teknoloji, donanım, yazılım, ağ ve network alanındaki ulaşılan nokta ile artık bilişim sistemleri sadece rutin karar vermeyi desteklemekle kalmamakta, diğer ve en önemli bileşen olan insan unsurunun yerini alma yolunda hızla ilerlemektedir. Günümüzde sistemler rutin veri işleyerek bilgi üreten sistemlerden, akıllı ya da zeki sistemler olarak isimlendirilen ve insanın karar alma sürecini ve yöntemlerinin karmaşık yapısını belli derecede kullanabilen, kendi kendine öğrenip sürekli iyileştirme yapmayı hedefleyen ve en önemlisi insansız veya insan unsurunun minimum katkısıyla karar almayı gerçekleştirebilen sistemlere dönüşmektedir. Bunların en güzel örnekleri kendi kendine sürüş yapabilen otomobillerin kontrol sistemleri, endüstri 4.0 uygulamalarını mümkün kılan yönetim ve planlama sistemleri, nesnelerin interneti sayesinde yönetilebilen akıllı bina ve ev yönetim sistemleri, insansız savaş gemisi kontrol ve yönetim sistemi, farklı amaçlarla üretilmiş olan robotlar olarak hayatın farklı alanlarından verilebilir.

Gömülü gerçek zamanlı sistemler dediğimizde genellikle büyük bir mekanik, sosyal, bilişsel, fiziki bir sistemin içine gömülmüş ve gelişmelere göre anlık olarak işlem yaparak sonuç üretebilen ve genellikle büyük sistemi kontrol etmeyi yada yönetmeyi amaçlayanbilişim sistemleri anlaşılmalıdır. Örneğin bir gemide gemi kaptanına yardımcı olacak bir bilişim sistemi vardır. Bu geminin, mekanik,

elektronik ve çevresel alt sistemlerinin verilerini toplar ve bir ekranda kaptana gösterir. Bu sistem radarlardan veri alır, sensörlerden gelen rüzgar, hız, dalga boyu, derinlik, diğer gemilerin pozisyonu gibi bilgileri kaptana gösterir ve genellikle navigasyon sistemi olarak isimlendirilir. Gemideki gömülü gerçek zamanlı sistem ise, navigasyon sistemi tarafından geminin farklı alt sistemlerinden toplanan verilerin hepsini birlikte kullanıp işleyerek gemi kaptanına tavsiyelerde bulunacak ya da gemiyi doğrudan kendisi yönetip kullanabilecek sistemdir. Gömülü sistemler bir nevi diğer mekanik, fiziksel, sosyal sistemlerin verilerini toplayıp onları yöneten birer kumanda veya yönetim merkezi şeklinde düşünülebilir.

Bu çalışmadaki amacımız uygulamaları ve sayıları hızla artan ve artık ciddi anlamda özellikle stresli ortamlarda insan karar vericilerin işini ve hata ihtimalini azaltan çoğu zamanda onların yerine kullanılmaya başlanan Gömülü Zeki Gerçek-Zamanlı Sistemler ve bu sistemlerin karmaşıklığının incelenmesidir. Bu sistemler doğası, çevresi, yapısı ve yaptıkları iş gereği karmaşık sistemlerdir. Bunlarda meydana gelebilecek en ufak bir aksama yada problem içinde gömülü bulundukları tüm sistemi riske atacaktır. Bu nedenle bu sistemlerinin karmaşıklığının çalışılması, bu sistemlerde karmaşıklığın nasıl ve hangi boyutlar ile ölçülebileceğinin belirlenmesi ve sistem için çalışmama yada hata yapma riskinin minimize edileceği karmaşıklık düzeyinin belirlenmesi gerekmektedir. Çalışmanın amacı öncelikle Gömülü Zeki Gerçek-Zamanlı Sistem tanımı yapmak, bunların uygulama alanlarını sunmak ve bu sistemlerin karmaşıklığını oluşturan faktörlerin ve karmaşıklık türlerinin sistematik bir yapıda incelenmesidir.

2.2 Büyük Ölçekli Güvenliğin Öncelikli Olduğu Sistemler

Büyük Ölçekli Güvenliğin Öncelikli Olduğu Sistemler (Safety-Critical Large Scale Systems) sistem güvenliği ve güvenilirliğinin ön plana çıktığı olası kaza durumlarında çok büyük çaplı felaketlere ve toplumsal olaylara neden olabilen sistemlerdir. Büyük Ölçekli Sistemler literatürde birbiriyle çatışan hedeflere sahip olan ve her biri belirli bir seviyede kaza riski barındıran birçok farklı kişi, mekan, kaynak ve teknik parçalardan meydana gelmiş sistemler olarak tanımlanmaktadır. Böylelikle sosyo-teknik sistemler olarak da ifade edilen bu sistemler sistem parçalarının ve aralarındaki ilişkinin karmaşık olması neticesinde olası kazaların etkilerine karşı savunmasız bir hale gelmektedir. Büyük ölçekli sistemlerin ortak özelliklerinden bazıları çok fazla karar değişkeni, dış kaynaklı değişken, durum değişkeni ve farklı parçalar (alt-sistemlere) sahip olmalarıdır (Yacov, 1982). Fonksiyonellik, güvenlik ve güvenilirliğin ön plana çıktığı bu sistemlerle havacılık, trafik kontrol sistemleri, medikal sistemler ve enerji kontrol sistemlerinde sıklıkla karşılaşmaktayız (Jaalinoja, 2004). 1986 yılında büyük ölçekli güvenliğin önde olduğu sisteme örnek olabilecek Çernobil Nükleer Santrali'nde meydana gelen kazadan sonra bu sistemler olası kazalarda çevreye ve insan hayatına verdikleri büyük zararlar nedeniyle oldukça dikkat çekmiştir (Read 1993'den akt. Coşkun ve Grabowski, 2005). Bir başka örnek ise Exxon Valdez kazasıdır. Alaska yakınlarında sahile oturan petrol yüklü tankerin taşıdığı ham petrol tüm sahile yayılarak o bölgede biyolojik yaşamı sonlandıran, geçimini balıkçılık ve deniz ürünleri ile sağlayan yöre halkına ekonomik zarar veren en büyük çevre felaketlerinden birine neden olmuştur. Büyük ölçekli güvenliğin öncelikli olduğu sistemlerde meydana gelen bir diğer büyük kaza ise 1988 yılında ABD tarafından düşürülen İran Hava Yolları'na ait yolcu uçağı kazasıdır. Bu uçak Basra Körfezi üzerindeyken ABD donanma gemisi tarafından savaş uçağı olduğu düşünülerek füzeyle vurulmuştur. Bu kazanın temel nedeni belirsiz verinin mevcudiyetinde yanlış karar alınmasıdır. Radarda donanma gemisi kaptanı tarafından görülen yolcu uçağının kimliği tanımlanamamış, zaman baskısı olduğundan ve detaylı veri bulunmadığından uçağı vurma emri verilmiştir. Bu tür kazaların dikkat çekmesindeki en büyük etkenlerden biri de bunların

"Safety-Critical" (Güvenliğin Öncelikli Olduğu) olarak ifade edilen sistemler içerisinde meydana gelmiş olmasıdır. Bu hususla ilişkili olarak bu sistemlerde güvenlik ve emniyet gibi konular oldukça önem kazanmaktadır. Bu sistemlerin güvenilirliğini artırmak, kaza risklerini azaltmak veya sistem kullanıcılarına karar verme, izleme ve otomasyon gibi sistem fonksiyonlarında yardımcı olarak insan hatasını minimize etmek amacıyla bu sistemler "Zeki Sistemlerle" desteklenmektedir (Coşkun ve Gabowski, 2005). Aşağıdaki sistemler bunlara örnek olarak verilebilir:

- Gemi ve Uzay Kontrol Sistemleri
- Hava Trafiği Kontrol Sistemleri
- Nükleer Santral Kontrol Sistemleri
- Zeki Otoyol Kontrol Sistemleri
- Askeri Savunma Sistemleri

Kaza kavramının tanımı geleneksel olarak ölüm, yaralanma, sakatlık, hastalık veya ekonomik kayıplara neden olan beklenmedik olay ya da bir dizi olaylar olarak yapılmaktadır. Kazalar genellikle istenmeyen ve beklenmeyen enerji veya tehlikeli madde salınımlarıyla , insan hayatını ve çevreyi etkileyecek şekilde sonuçlanabilmektedir. Bu açıdan bakıldığında bilgisayarlar göreceli olarak güvenli aygıtlarken, potansiyel olarak tehlikeli bir sistemin içerisinde alt bir sistem olarak bulunduklarında büyük kazalara neden olabilmektedirler (Leveson, 1991). Bu nedenle nükleer enerji santrallerini, hava taşıtı veya diğer ulaşım araçlarını, sağlık sistemlerini, üretim süreçlerini, havacılık ve savunma sistemlerini izlemek ve kontrol etmek için kullanılan bilgisayarların yol açabileceği potansiyel problemler tüm sistem güvenliği bağlamında ele alınmalıdır. Sistem güvenliği mühendisleri güvenliği tehlike ve riskler açısından ele almaktadırlar. Çünkü genel olarak birçok faktör tarafından oluşan kazalara karşı güvenlik, kazalardan ziyade tehlikeler bakımından tanımlanmalı ve bu sayede de sistem tasarımında tehlike teşkil edebilecek unsurların tümüne dikkat edilmelidir (Leveson, 1991).

2.3 Gömülü Zeki Gerçek-Zamanlı Sistemler (Embedded Intelligent Real-Time Systems-EIRTS)

Tüm bunlar göz önüne alındığında büyük çaplı ve hayati tehlike arz eden sistemler söz konusu olduğunda kaza riskini minimize etmek, sistem bileşenleri ve aralarındaki ilişkileri izlemek adına kullanıcıların bu aktiviteleri yerine getirebilmesi ve karar almalarını desteklemek amacıyla bu sistemlerin içerisine genellikle "Gömülü Zeki Gerçek-Zamanlı Sistemler" yerleştirilmektedir. Bunun gerçekleştirilme biçimi tamamen sistem özelliklerine dayanmakla birlikte insan etkileşimini gerektiren durumlarda kullanıcıya bilgi ve karar alma yetkisi verilmektedir. Bazı durumlarda ise sistemin tamamen otomasyonu sağlanarak alt bir sistemin izleme ve kontrol faaliyetlerini gerçekleştirmesi, problemli alanları belirlemesi, veri toplaması ve buna bağlı olarak karar vermesi gerçekleştirilmektedir (Coşkun, 2001). "Gömülü Zeki Gerçek-Zamanlı Sistemler" gömülü sistemler, zeki sistemler ve gerçek-zamanlı sistemlerin özelliklerini taşımaktadır. Gömülü Zeki Sistemler ise zeki sistemlerin sistem elemanlarını kontrol etmek, öğrenmeyi sağlamak, sistemi düzeltmek ve kullanılacak verilere karar vermek gibi özelliklerini hem de gömülü sistemlerin özelliklerini taşırlar. Gömülü Zeki Gerçek-Zamanlı Sistemler söz konusu olduğunda sistemin olaylara zamanında cevap verebilirliği önem kazanmaktadır. Çevreden veya sistemin kendisinden kaynaklanan olaylar zaman kısıtları dahilinde ve doğru bir şekilde bu sistemlerde cevap bulmalıdır. Gömülü Zeki Gerçek-Zamanlı

Sistemlerin sahip olması gereken temel fonksiyonlar Coşkun'un (2001) çalışmasında belirtildiğine göre şu şekildedir:

- Büyük bir sistemin parçası olarak işlev görmek
- Sistem bileşenlerini veya tüm sistemi izlemek
- Karar vermeyi desteklemek amacıyla mantıksal çıkarımda bulunmak
- Gerçek-Zamanlı bir sistemin parçası olmak

2.3.1 Gömülü Sistemler

Gömülü Sistemler (Embedded Systems) endüstrisi öncelikle makine kontrol uygulamalarında kullanılmaya başladıktan sonra uygulamaları farklı alanlara yayılmıştır (EETimes, 2010) ve genel amaçlı bilgisayar sistemleriyle paralel olarak değişmiştir. Geçmişte genellikle gerçek -zamanlı sistemler (real-time systems) ve endüstriyel sistemlerle tanımlanan gömülü araştırma ve geliştirme faaliyetleri günümüzde bilgisayarların yaygın kullanımıyla birlikte bu teknolojilerle daha sık ve her alanda karşılaşılmaktadır (Wordpress, 2012)

Gömülü Sistemler, dışarıdan görülemeyen ve genel olarak kullanıcı tarafından direkt olarak ulaşılamayan hesaplama mekanizmaları ve özel yazılım içeren elektronik araç-gereçler veya daha karmaşık sistemlerdir (IDC, 2012). Bu sistemler günümüzde birçok sektörde inovasyon ve büyümenin temelini teşkil etmektedir. Bu sektörlerden bazıları aşağıda listelenmiştir:

- Otomotiv
- Havacılık Elektroniği/ Uzay Endüstrisi
- Endüstriyel Otomasyon
- Ulaşım, Su ve Çevre Koruma
- Sağlık ve Medikal Sistemler
- Enerji Tüketim Teknolojileri (Ev/Bina)
- İletişim
- Tüketici Elektroniği
- Enerji

Gömülü Sistemler mekanik, yazılım ve donanımı birleştiren elektro-mekanik sistemlerdir. Gömülü bir sistemin büyüklüğü küçük bir termometreden kimyasal bir reaktörün kontrolüne kadar değişebilmektedir (Jaalinoja, 2004). Bu sistemlerin zaman kısıtlarına sahip olanları Gömülü Gerçek-Zamanlı Sistemler olarak bilinmekle birlikte bu sistemler aşağıdaki bölümde açıklanmaktadır.

2.3.2 Gerçek-Zamanlı Sistemler

Gerçek-zamanlı sistemler dış kaynaklı olaylara minimum süre içerisinde yanıt vermesi gereken ve zaman kısıtları altında işlev gören bilgisayarlaştırılmış sistemlerdir (Pathan, 2010; Stankovic, 1988). Sistemin doğruluğu hesaplamadaki mantıksal sonuçların yanında sonuçların üretildiği zamana da bağlıdır. Bunun temel nedeni ise sistemin değişen çevreyle sürekli etkileşim halinde olmasıdır (Bernat, Burns ve Llamosi, 2001). Bu sistemler gömülü bir sistemin içerisinde bulunan basit mikro-kontrolörlerden oluşabilmekle birlikte oldukça karmaşık ve dağıtılmış farklı mekanizmaları kapsayan sistemler de olabilmektedir. Sistemin karmaşık olma derecesi bu bakımdan oldukça geniş bir yelpazede değişmektedir. Kumanda ve kontrol sistemleri, süreç kontrol sistemleri, yüksek hızdaki iletişim sistemleri Gerçek-Zamanlı sistemlere verilebilecek örneklerdendir. Bunların yanı sıra Uzman Sistemler ve diğer Yapay Zeka (Artificial Intelligence-AI) teknolojilerine entegre edilebilen bu sistemler, bu teknolojilerle birleştirildiğinde sistem karmaşıklığını daha da artırmaktadır (Stankovic, 1996).

Bu sistemler bir görevin yerine getirilme süresindeki zaman aşımının yarattığı etkilere göre "hard" veya "soft" gerçek-zamanlı sistemler olarak sınıflandırılmaktadırlar. "Hard" gerçek-zamanlı sistemler söz konusu olduğunda bir olaya cevap süresinin geçmiş olmasının sonucu felaket olarak nitelendirilebilecekken, "soft" gerçek-zamanlı sistemlerde sistemin cevap süresini aşmış olmasının etkileri göreceli olarak daha hafiftir (Pathan, 2010).

Gelecek nesil gerçek-zamanlı sistemlerin büyük, karmaşık, dağıtılmış, zeki ve belirsizliğin giderek arttığı ortamlarda işlev görebilmesi gerekmektedir. Bu sistemler cevap verebilirlik için yeterince esnek, beklenmeyen sistem durumlarına karşı reaksiyon almak için oldukça hızlı, gereksinimler değiştikçe değişebilen ve bu sırada da geliştirme, test ve doğrulama gibi faaliyet maliyetlerinde tasarrufu gerçekleştirebilen sistemler olmalıdır. "Gömülü Zeki Sistemler" başka bir sistem içerisinde yerleşik bulunarak kaynakları paylaşır, çevre ve diğer sistemlerle iletişim kurar, mantıksal çıkarımlar için veri toplar, bilgi tabanı aracılığıyla veriyi işler ve sonuçlar üreterek bunların paylaşılmasını sağlar (Coşkun, 2001). Bu bağlamda çoğu gömülü yazılım gerçek zamanlıdır ve genellikle büyük karmaşık bir sistem içerisine yerleşik durumda olup bu sistemin kontrol edilmesini ve izlenmesini sağlar (Highland, 1994'den akt. Coşkun, 2001). Aşağıdaki bölümde zeki sistemlerin özellikleri daha detaylı olarak ele alınmaktadır.

2.3.3 Zeki Sistemler

Zeki bir sistemin en önemli özelliği kendi fonksiyonelliğini artırmak amacıyla çevreyle etkileşim haline bulunarak öğrenme kabiliyetine sahip olmasıdır (Poggio ve Stringa, 1992). Bu sistemler dış çevre ve farklı sistemlerden veri toplayarak veriyi doğrulamak amacıyla farklı kaynakları veya sistem zekasını kullanmaktadırlar. Bu doğrulanan veri ve bilgi sistem zekasına bağlı olarak karar vermeyi desteklemektedir. Bir karar verildikten sonra, sonuçlar sistem operatörlerinin veya diğer sistem bileşenlerinin son karar verme sürecini desteklemek amacıyla kullanıcıya veya zeki sistemin içerisinde yerleşik olduğu sisteme gönderilmektedir. Bu bağlamda, mantıksal çıkarım zeki sistemlerin en temel özelliğidir ve bu mantıksal çıkarım insan zekası ve insanların karar alma sürecine benzer bir mantıktaki fonksiyonlar sayesinde gerçekleştirilmektedir. Eğer sistemin hedefi kontrol etmek ise alt sistemler ve bileşenler beklenen performanstan sapmalara yol açabilecek olası kazalar ve tehlikeler için izlenerek, durumunun düzeltilmesi ve raporlanması sağlanmaktadır. Gömülü sistemler, zeki sistemler, gerçek-zamanlı sistemler ve gömülü zeki gerçek-zamanlı sistemlerin özellikleri ve aralarındaki farklılığa ilişkin özet bilgi aşağıdaki tabloda sunulmaktadır.

Tablo 2.1: Gömülü Zeki Gerçek-Zamanlı Sistemler ve Özellikleri (Coşkun, 2001)

Gömülü Sistemler	Zeki Sistemler	Gerçek-Zamanlı Sistemler	Gömülü Zeki Gerçek-Zamanlı Sistemler
Esas sistemle iletişim	Kullanılacak veri ve bilgiye karar vermek	Zamanında cevap	Kullanıcı ve sistem arasındaki iletişimi yönetmek
Karar vermek için veri toplamak, yorumlamak ve sonuçları paylaşmak	Sistemi düzeltmek	Doğruluk	Girdilerin doğruluk ve tutarlılığını kontrol etmek
İzleme faaliyetleri	Geçmiş hatalardan ders çıkartmak		Karar vermek için verilerin bütünleştirilmesi
	Her bir sistem bileşeninin çalışmasını kontrol etmek		Kullanılacak veri ve bilgiye karar vermek
			Geçmiş hatalardan ders çıkarmak
			Sistemi düzeltmek
			Zamanında cevap vermek
			Her bir sistem bileşeninin çalışmasını kontrol etmek

2.4 Gömülü Zeki Gerçek-Zamanlı Sistemlerin Karmaşıklığı

Bilgisayarlar yapı itibariyle güvenli olmayan araçlar değildir ve yazılım ciddi bir mantık hatası olmadıkça direkt olarak kazalara sebebiyet vermez. Örneğin uçuş esnasında tehlikeli bir durum söz konusu olduğunda veya kaza riski bulunduğunda sonuçlar pilotların tecrübe ve yetenekleri, çevikliği, şans, görüş alanı gibi kontrol sistemlerini tasarlayan mühendislerin kontrolü altında olmayan faktörlere de bağlıdır. Fakat sistem yazılımı tehlikeli olayların oluşumuna katkı sağlayabileceğinden yazılımla ilgili olan problemlerin azaltılması ve kontrol edilmesi riski azaltacaktır. Risk kavramı mühendisler tarafından tehlikeli bir olayın olma ihtimali olarak tanımlanmaktadır. Sistem mühendisliğinin bir alt dalı olarak sistem güvenliği mühendisleri uygun olan bilimsel, yönetsel ve mühendislik tekniklerini kullanarak sistemin bütünündeki risk faktörlerini elimine ederek kabul edilebilir bir risk seviyesini sağlamayı hedeflerler. Yazılım sistemsel tehlikeler üzerinde etki göstermek suretiyle riske olumlu veya olumsuz yönde etki yapmaktadır. Bu nedenle bazı yazılım güvenliği mühendisleri problemlerin sistemsel doğasına vurgu yapmak ve muhtemel çözümlerin sistem bağlamında ele alınmasını sağlamak amacıyla "yazılım güvenliği" teriminden ziyade "yazılım sistem güvenliği" terimini kullanmaktadırlar (Leveson, 1991). Literatürde "Karmaşıklık" kavramına ilişkin farklı disiplinlerin farklı tanımları bulunmaktadır. Tanımlar, ölçüm araçları ve metrikler birçok çalışmanın konusu olmakla birlikte matematik, psikololoji, sosyal bilimler, sistem bilimleri, ekonomi ve bilgisayar bilimleri gibi disiplinler tarafından farklı şekillerde yorumlanmıştır. Karmaşık sistemler parçalardan oluşan, parçaların fiziksel ve fonksiyonel olarak heterojen olduğu ve sistem fonksiyonuna hizmet edecek biçimde organize edildiği ve etkileşim halinde olduğu sistemlerdir. Zio (2014) tarafından bu özelliğin yapısal ve dinamik karmaşıklığa neden olduğu belirtilmektedir. Yazılım karmaşıklığı yazılım ve sistem mühendisleri tarafından uzun yıllar boyunca kod özelliğine dayanan koddaki hata sayısı, geliştirme zamanı ve maliyeti, kodun büyüklüğü ve satır sayısı gibi metriklerle ölçülmüştür (Coşkun, 2001). Coşkun (2001) çalışmasında "Gömülü Zeki Gerçek-Zamanlı Sistemlerde" karmaşıklık için bir model önerisinde bulunmuştur. Bu modele göre "yazılım karmaşıklığı" için belirlenen "karmaşıklık" çeşitleri şu şekildedir:

- Mimariye dayalı/Yapısal Karmaşıklık
- Veri işleme/Mantıksal çıkarım/Fonksiyonel Karmaşıklık
- Kullanıcı Arayüzü Karmaşıklığı
- Karar destek/Anlam Karmaşıklığı

2.4.1 Mimariye dayalı/Yapısal Karmaşıklık

Gömülü Zeki Gerçek-Zamanlı Sistemler (EIRTS) büyük ölçüde yazılıma dayalı olan sistemlerdir. Kazman ve Burth (2016) büyük yazılım sistemlerinin başarısında temel unsurlardan birinin yapısal sadelik olduğunu ve bu bağlamda bu sistemlerde mimariye dayalı/yapısal karmaşıklığın ölçümünde

kullanılacak olan metriklerin önemini vurgulamaktadırlar. Bilgisayar bilimleri ve yazılım mühendisliği disiplinlerine dayanan bu metrikler matematiksel ve ekonomik modeller aracılığıyla kod karmaşıklığını ölçmeyi hedeflemektedir (Coşkun, 2001).

Shao ve Wang (2003) bu metriklerin temel olarak program büyüklüğü, program akış grafikleri veya Halstead'ın "yazılım bilimi metrikleri" (software science metrics) gibi modül arayüzlerine dayandıklarını belirtmişlerdir. Yaygın olarak kulanılan bir diğer karmaşıklık ölçümü ise McCabe tarafından geliştirilen yazılım modülünün karar yapısınının karmaşıklığını ölçmeyi hedefleyen "siklometik karmaşıklıktır" (cyclomatic complexity) (Coşkun, 2001).

2.4.2 Veri işleme/Mantıksal çıkarım/Fonksiyonel Karmaşıklık

EIRTS farklı kaynaklardan elde ettiği veriyi işleyerek analiz eder, ve çeşitli mantıksal çıkarım algoritmaları aracılığıyla sonuçlar üreterek karar vericilerin veya kullanıcıların kararlarını desteklemek amacıyla onlara iletilmesini sağlar. Bu bağlamda EIRTS karmaşıklığının bir diğer boyutu bu sistemlerin sistem zekası veya veri işleme mekanizmasına yönelik olan "Veri işleme/Mantıksal çıkarım/Fonksiyonelik" karmaşıklığıdır.

2.4.3 Kullanıcı Arayüzü Karmaşıklığı

Kullanıcılar Gömülü Zeki Gerçek-Zamanlı Sistemlerin ve Güvenlik Öncelikli Sistemlerin bir diğer önemli bileşenidir. Otomasyona en çok dayalı kontrol sistemleri uygulamalarında dahi sistemin en az bir yöneticisi vardır. Karar destek sistemlerinde ise kullanıcı bir kişi veya bir grup karar vericiden oluşabilmektedir. Bu sistemlerde EIRTS ve kullanıcıları arasındaki iletişim kullanıcı arayüzleri aracılığıyla gerçekleştirilmektedir. Sistem fonksiyonlarınının ve özelliklerinin etkin bir şekilde kullanılabilmesi için sistem arayüzündeki bilginin (ekran) kullanıcılar tarafından kolay görülür, anlaşılır ve yorumlanabilir durumda olması gerekmektedir. Bu sebeple "Kullanıcı Arayüzü Karmaşıklığı" da Coşkun (2001) tarafından EIRTS karmaşıklığı için geliştirilen modele dahil edilmektedir.

2.4.4 Karar destek/Anlam Karmaşıklığı

Gömülü Zeki Gerçek-Zamanlı Sistemler, kullanıcılar ve karar vericilere bilgi ve karar desteği sağlamaktadırlar. Karar vericilerin etkin ve iyi kararlar verebilmesi amacıyla bu sistemlerin durumla ilgili doğru bilgi vermesi, alternatif kararları belirleyerek her bir alternatifin sonuçlarını sunması gerekmektedir. Bu nedenle, verdikleri kararların kalitesini artırmak amacıyla kullanıcıya alternatiflerin belirlenmesi, test edilmesi ve sonuçların tahmini gibi hususlarda destek veren bu sistemlerin karmaşıklığının bir diğer bileşeni de "Karar destek/Anlam" Karmaşıklığıdır.

2.5 Gömülü Zeki Gerçek-Zamanlı Sistemlerin Tasarımı

Hayati tehlike arz eden sistemler söz konusu olduğunda güvenlik ve emniyeti sağlamak amacıyla bu sistemleri kontrol eden ve yöneten yazılımların büyük bir dikkatle tasarlanıp geliştirilmesi gerekmektedir. Çevrede, insan yaşamında ve finansal kaynaklarda yapacağı potansiyel hasarlara göre sınıflandırılabilen sistemlerde, risk seviyesi en yüksek grupta bulunan sistemler tüm hata türlerine ve başarısızlık risklerine göre değerlendirilmeli ve test edilmelidir (Coşkun, 2001).

Günümüzde bu sistemlerle ev eşyaları, otomotiv gömülü sistemleri, taşınabilir aygıtlar, elektrikli aletler, akıllı telefonlar, gömülü medikal araç-gereçler endüstriyel ürünler ve aletler başta olmak üzere yaşamımızın her alanında karşılaşmaktayız. Internet teknolojilerinin yaygınlaşmasıyla birlikte

Endüstri 4.0 ve Nesnelerin İnterneti gibi bilişim alanındaki güncel gelişmeler ve uygulamalara paralel olarak gömülü sistemlerin hayatımızdaki yerini daha fazla artıracağı aşikardır. Bilgi ve iletişim teknolojilerindeki gelişmeler bu sistemleri giderek daha karmaşık hale getirerek daha güçlü, etkin, performans özellikleri bakımından üstün sistemlerin geliştirilmesi ve pazarlanması bakımından hem yeni fırsatlar hem de başa çıkılması gereken yeni güçlükler doğurmaktadır (EETimes, 2010). Bu sistemlerin yaygınlaşması sistemlerin sahip olması gereken güvenlik, güvenilirlik ve emniyetli olma gibi özelliklere odaklanılmayı gerektirmektedir. Ayrıca yeni gömülü sistemler uygulamalarının yanı sıra birçok sektörde ürünler bağlamında talep edilen etkinlik ve performansı karşılamak amacıyla bilgisayar kontrolüne olan gereksinim artmaktadır. Geliştirilen yeni Gömülü Sistemler uygulamalarıyla birlikte mevcut gömülü sistemler mimarilerine de artan işlem gücü ve internet bant genişliği gibi talepleri karşılamak için gerekli geliştirmeler yapılmalıdır (Hallmans, tarihsiz).

Birçok yaygın görüşün aksine Gömülü Sistemler salt elektronik mekanizmalardan ibaret değildir. Dijital ve analog parçalar, özel amaçlı sensörler, aktüatörler, yazılım, çeşitli mekanik parçalar gömülü sistemlerin bileşenleri olabilirken bu sistemlerin tasarlanmasında ağırlık, maliyet, güç tüketimi, gerçek zamanlı cevap verebilirlik (real-time responsiveness), termal enerji yayılımı, radyasyon / emisyon, sağlamlık ve radyasyona dayanıklılık gibi gereksinimlerin dikkate alınması gerekmektedir (EETimes, 2010).

Gömülü Sistemler giderek artan bir biçimde cep telefonları, kişisel yardımcı aygıtlar, akıllı kartlar, gömülü sensörler ve giyilebilir teknolojiler gibi topluma yayılan uygulamalar haline geldikçe insanlar ve bilgisayarlar arasındaki etkileşim de giderek artmaktadır. Bu teknolojilerin kişisel ve ticari altyapılara entegre edilmesi güvenlik konusunu daha da önemli bir hale getirmiştir. Örnek olarak kablosuz ağ üzerinden doktorlara veri gönderen giyilebilir bir kalp-izleme cihazında bulunan gömülü bir sistem gönderdiği veriyi güvenli ve bozulmamış bir şekilde ulaştırmalıdır. Ayrıca, Gömülü Sistemler genellikle işlemci tabanlı aygıtlar olup kaynakların sınırlı olduğu durumlarda iş görmektedir. Bu bağlamda bu sistemlerin tasarımları geleneksel sistem tasarımı ve güvenli tasarım konularından farklılaşmaktadır (Hwang, Schaumont ve Verbauwhede, 2006).

2.5.1 Yazılım Güvenliği ve Güvenli Sistem Tasarımı

Yazılım güvenliğine ilişkin hususların bilgisayarların büyük ve potansiyel olarak tehlikeli olan sistemlerin içerisine kısmi veya bütünsel kontrolü sağlamak amacıyla gömülü olduklarında ortaya çıktığından daha önce bahsedilmişti. Gömülü bir sistemin sağladığı kontrol fonksiyonu sistemin gerçekleştirmesi gereken kontrol fonksiyonundan ve sistemin çalışmasına ilişkin kısıtların bütünleştirilmesinden oluşmaktadır. Çoğunlukla güvenlik kısıtlarıyla talep edilen fonksiyonellik arasında karar vermek etik, ahlaki, yasal, finansal ve sosyal boyutları dikkate almayı gerektirmektedir. Güvenlik kısıtları bilgisayar yazılımıyla yakından ilişkilidir. Bir bilgisayar, kontrol sisteminde aşağıdaki üç temel fonksiyonu gerçekleştirebilmektedir;

1. Veri girişi
2. Direkt dijital kontrolü içeren veya kontrol edilen değişkenlerle olayların sıralamasına ilişkin temel kontrol fonksiyonlarının gerçekleştirilmesi
3. Güvenlik şartlarının gerçekleştirilmesi (güvenlik kilidi olarak görev yapma)

Bu fonksiyonlardan ilk ikisinde bilgisayarların sistem içerisindeki rolleri direkt olarak güvenlikle ilişkili değildir. Fakat diğer durumda bilgisayarların görevi yerleşke sistem içerisinde sistemin güvenli durumunu sürdürmek ve sistem-güvenlik kontrol fonksiyonunun desteklenmesini sağlamaktır (Leveson, 1991). Herhangi bir sistemde sistemin alt parçaları bu fonksiyonlardan bir veya birkaçını gerçekleştirebilmektedir. Örnek olarak, nükleer enerji santralinde bir bilgisayar yönetsel

ve düzenleyici faaliyetler için sistemin işlemsel verisini toplayabilirken bir yandan da güç üretimi ünitesinin kontrol fonksiyonlarını yerine getirebilmektedir. Ayrıca tehlike durumları algılandığında sistemin emniyetli duruma geçmesini sağlayacak emniyet supabı olarak ta görev yapabilmektedirler. Fakat güvenlik açısından bakıldığında bilgisayarların kontrolör ve güvenlik sağlama amaçlı olarak kullanıldıkları durumlar sistem güvenliğiyle yakından ilişkilidir (Leveson, 1991).

2.6 Gelecekte Gömülü Sistemler

Nesnelerin İnterneti (IoT) uygulamalarının temelinde gün geçtikçe daha zeki hale gelen gömülü sistemler yer almaktadır. Bu sistemler ürünlerdeki fonksiyonelliği artırma potansiyeline sahip olmakla birlikte yeni hizmet olanakları ve işletmeler için daha fazla kar sağlama imkanı tanımaktadırlar. Daha zeki sistemlerin talep edilmesi ve bu sistemlerden beklentiler bunların içerisinde yer alan gömülü yazılımlara yapılan yatırımları artırmıştır. IoT' den beklenen değeri maksimize etmek amacıyla yazılım geliştirmede kullanılan süreçler ve teknolojiler tekrar gözden geçirilmelidir (Rommel ve Walek, 2017).

Ürün yapılarının artan bir biçimde daha karmaşık hale gelmesi ve yazılımdan talep edilen daha fazla fonksiyonellik ürün kalitesi, ürün geliştirme süreleri ve ARGE bütçesi açısından mevcut durumun yeterli olmamasına neden olmaktadır. Bu bağlamda geleneksel yazılım geliştirme metodolojilerinden çok daha sofistike ve entegre yönetim çözümlerinin uygulanma gereksinimi ortaya çıkmaktadır. Karmaşık sistem gereksinimleriyle başa çıkabilecek proje yönetimi tekniklerinin ve süreçlerin yazılım temelli ürünlerin yaşam döngüsüne adapte edilmesi gerekmektedir.

Gereksinim yönetimi, sistem mimarisi ve simülasyonu için entegre edilmiş araçlar ve ürün hattı mühendisliğinin yazılımın yaratılması ve yeniden kullanılabilirliği bakımından katkısı büyüktür. Ayrıca bu süreçler ve mekanizmaların kullanılması özellikle güvenliğin öncelikli olduğu sistemler bakımından tasarım döngüsü boyunca test edilmesi ve değişim yönetimine imkan tanıması bakımından değer katma potansiyeline sahiptir (Rommel ve Walek, 2017).

Geleneksel Bilgi Teknolojileri (BT) harcamalarından daha hızlı büyüme gösteren Gömülü Sistemler teknolojileri 2010 ve 2015 yılları arasında Avrupa için yıllık yaklaşık olarak %12 büyüme göstermiştir. 2020 yılına ilişkin senaryo ise Gömülü Sistemlerin daha zeki sistemler halini alacağı yönündedir. Bu dönüşüm için gerekli olan bir takım şartlar ve gereksinimler IDC (2012) raporunda şu şekilde özetlenmiştir:

Tablo 2.2: Gömülü Sistemlerin Evrimleşmesinde Gereksinimler (IDC, 2012).

Politik & Yasal Gereksinimler	**Kullanıcı Gereksinimleri**	**Ekonomik Gereksinimler**	**Teknolojik Gereksinimler**
Güvenlik	İnsan Bilgisayar Etkileşimi	Dağılmış Mimari Yönetimi	Çok Çekirdekli İşlemciler
Emniyet	Kusursuz Bağlantı	Sistem Otonomisi	Sanallaştırma Yazılımları
Düzenlemeler	Birlikte İşlerlik		Küçük Aygıtların Enerji Yönetimi
Sertifikasyon			

2.6.1 Gömülü Sistemlerin Geleceği ve "Systems of Systems" (SoS) Yaklaşımı

Yeni nesil Gömülü Sistemler uygulamaları beraberinde 2 inovasyon alanını da getirmektedir. IDC (2012) raporuna göre bunlardan biri yüksek performanslı ve programlanabilme kapasitesi yüksek mikro-işlemciler, internet bağlantısı, yüksek seviye işletim sistemleri ve ara yazılım uygulamalarıdır. Diğeri ise veri ve hizmetlerden oluşan küresel dijital ağlardır. Neticede Gömülü Sistemlerin "System

of Systems (SoS)" konseptine evrilmesi söz konusudur. Bu aşamada ise artan karmaşıklık ve dayandığı temel etkenler şu şekilde özetlenebilir:

- Verilerin çoğalması
- Fonksiyonlar ve servislerin çeşitliliği ve zenginliği
- Birlikte işlerlik ve ağ oluşturma

2020 yılına ait öngörüler kişi başına düşecek olan yaklaşık 100 işlemci ve çok sayıdaki aygıtın yüksek bir hesaplama yoğunluğuna yol açacağı şeklindedir. Bu araç gereçlerin çoğalması BT varlıklarının kontrol ve izleme, geliştirme ve konfigürasyon yönetimi için sofistike sistem yönetimi yazılımlarına gereksinimi artıracaktır. "Systems of Systems" yaklaşımıyla Gömülü Sistemlerin birbirleriyle bağlantısı ve sensör entegrasyonu yardımıyla sistemlerin veri alışverişi yapması mümkün olacaktır. Gömülü Sistemler her sistemin diğer sistemlerden ve çevreden veri toplamasını sağlayacaktır. Bununla birlikte artan hesaplama hacmi, yeni yazılım teknolojileri ve İnternet sistemlerin birlikte çalışabilirliğini artıracaktır.

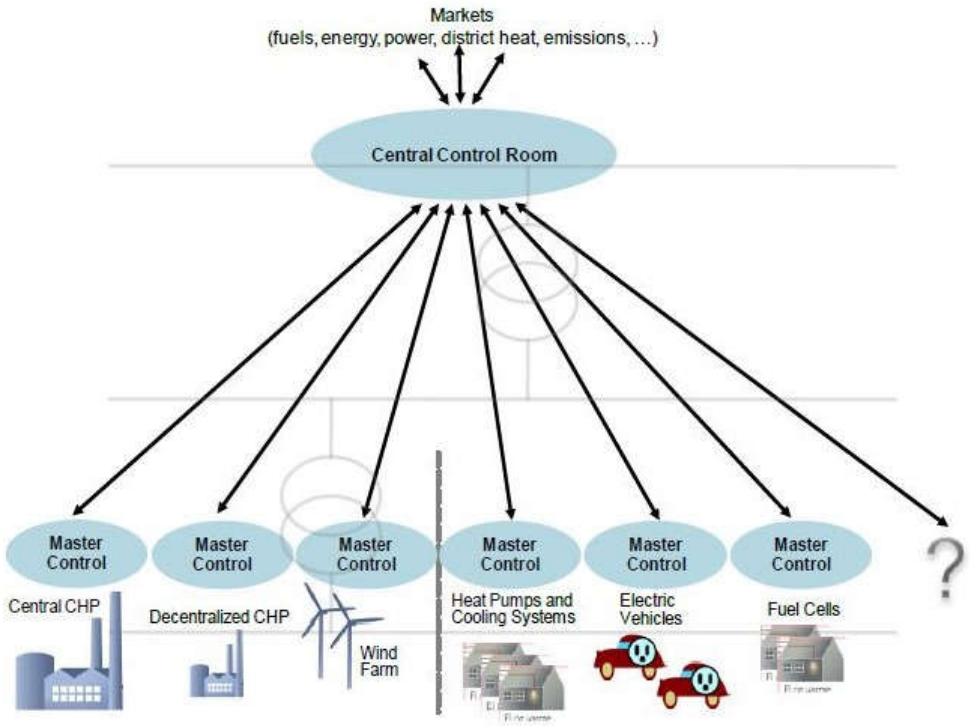
Gömülü Sistemler için 2020 yılı tahminleri bu sistemlerin geleneksel Gömülü Sistemlerden daha zeki sistemlere doğru evrimleşeceği yönündedir (IDC, 2012). Akıllı sistemlerin sayılarının artmasıyla birlikte her bir sistemin internet bağlantısı neticesinde üretilen veri miktarı daha fazla artış gösterecektir. Akıllı sistemler bulut ve mekan-bazlı sistemler ve sosyal ağlarla bağlantı kurarak müşteriler için yeni hizmetler ve ürünler sunulmasına imkan tanıyacaktır (IDC, 2012).

Gömülü Sistemlerin İnternetle buluşması toplumda ve endüstride kullanılan sistemlerin birbirleriyle entegrasyon ve bağlantı kurma yetisini artırmaktadır (EU Comission, 2012). Bu bağlamda tüm bu faktörler göz önüne alındığında gömülü sistemlerin karmaşıklık derecesi daha önce hiç olmadığı kadar yüksektir ve bu karmaşıklik derecesi "Karmaşık Sistem Mühendisliğinin" klasik bakış açısından "Systems of Systems" (SoS) Mühendisliğine doğru bir paradigma kaymasını meydana getirmiştir (EU Comission, 2012).

Bu yeni sistemlerle birlikte sistem mühendisliği topluluğu tarafından zeki şebekeler, entegre edilmiş tedarik zinciri uygulamaları, işbirlikçi organizasyonlar ve yeni nesil hava trafiği kontrolüne ilişkin uygulamalara odaklanan "Systems of Systems" yaklaşımı ortaya atılmıştır (Jamshidi 2009'dan akt. Samad ve Parisini, 2011). Lukkien (2015) bu yaklaşımın günümüzde bağımsız (otonom) alt sistemlerin birleşmesinden meydana gelen sistemleri ifade etmek için kullanılan modern bir terim olduğunu belirtmektedir.

Zeki Şebekeler günümüzde giderek artan bir ilgi çekmektedir. Elektrik ve bilgi akışını sağlayarak günümüzün klasik güç şebekelerinden üstünlüklerini ortaya koymaktadırlar. Bu şebekeler modernize edilmiş sistem altyapısına dayalı iletişim ve kontrol mekanizmalarına dayalıdır. Enerji dağıtımı, yenilenebilir enerji uygulamaları, dağıtımın optimize edilmiş şekilde gerçekleştirilmesi, sistem durumunun izlenmesi bu sistemlerin görevleri arasındadır. Bu bağlamda SoS olarak zeki şebekeler birçok fonksiyonu gerçekleştirecek otonom kontrol birimlerinden oluşmak zorundadır (Samad ve Parisini, 2011). Aşağıdaki şekilde yerel ihtiyaçları karşılayan kontrol sistemleri ve bunların etkin bir güç sistemi sağlamak adına birlikte fonksiyon göstermesini tasvir etmektedir.

Zeki Şebekelerin yanında Üretim Tedarik Zinciri, Gömülü Otomotiv Sistemleri "Systems of Systems" kapsamı dahilindedir. Samad ve Parisini (2011) günümüz otomobillerini tekerlek üzerinde giden "Gömülü Sistemler" topluluğu olarak tanımlamışlardır. Bu endüstride son yıllardaki inovasyonların büyük kısmı yerleşik bilgisayarlar, kontrol ve iletişim alanlarında meydana gelerek güvenlik,



Şekil 2.1: "Systems of Systems" olarak Zeki Şebekeler (Samad ve Parisini, 2011)

yakıt tasarrufu ve güvenilirliği büyük ölçüde artırmıştır. Güvenlikle ilgili olan "Gömülü Sistemlerden" bazıları şu şekildedir: çarpma etki-uyarı, zeki hız programlayıcısı, çekiş-stabilite kontrolü, hava yastığı açılma ve kilitlemeyen diferansiyel fren sistemleri. Bu sistemler ayrı tasarlanmasına rağmen otomobilden ve çevresinden bağımsız değillerdir. Bu nedenle araçta meydana gelen çeşitli arızaların kaynakları farklı sistemler olabilmektedir. Sürücünün motor çalışır vaziyette aracı terk etmesi durumunda aracın kendi kendisini kilitlemesi bu duruma örnek olarak verilebilir. Bu sebeple otomotiv sistemlerini tasarlarken çözüm SoS bakış açısıyla hareket etmektir. Gömülü Sistemlerin gelecekteki sınırlarını çizmek ve manzarayı ortaya koymak adına birbiriyle bağlantılı olan tüm sistemlerin bu bakış açısıyla ele alınması gerekmektedir. "System of Systems" bakış açısından farklı Gömülü Sistemler uygulamaları aşağıdaki tabloda sunulmuştur (IDC, 2012).

Tablo 3'te de görüldüğü üzere kontrol uygulamalarının genişleyen kapsamı ve karmaşıklığının artması kontrol sistemleri teknolojilerinde yeni talepleri doğurmuştur. Yeni uygulamalar geniş-kapsamlı ve karmaşık olma özelliğinin yanında ayrıca merkezi olmayan, dağılmış, heterojen ve yarı-otonom bileşenlerin ağ tabanlı olarak bir araya getirilmesinden oluşmaktadır.

2.6.2 SoS ve Karmaşıklık

"Systems of Systems" teknoloji, operasyon, coğrafi lokasyon ve kavramsal çerçeve bakımından çok çeşitli olan birçok bağımsız gömülü sistemden meydana gelen birbiriyle bağımlı sistemler topluluğudur (Karcanias, 2015). Fakat bu kavram literatürde farklı araştırmacılar tarafından farklı şekillerde ifade edilmektedir. Buna bağlı olarak ta araştırmacıların bakış açısına göre bu sistemlerin farklı özellikleri üzerinde durulmaktadır. Maier (1998) ve De Laurentis' den (2005) akt. Dersin ve Transport (2014) tarafından tipik özellikler şu şekilde belirtilmiştir:

Tablo 2.3: SoS Bakış Açısından Gömülü Sistemler (IDC, 2012)

Sektör	Etkileyen Etmenler	Sistemler/Teknolojiler
Ulaşım Hizmetleri-	· Karbon ayak izi/sürdürülebilirlik	**Zeki Şarj Sistemleri**
Otomobil 2.0 & Zeki Taşımacılık Sistemleri	· Mobilite	**Vehicle to Grid (V2G)**
	· Hizmet kalitesine ilişkin müşteri beklentisi	**Araç içi Teknolojiler (Yazılım-İletişim)**
	· Ürün/Hizmet farklılaştırma	· Şarj Altyapı Teknolojileri
		· Geniş Alan Ağları/GPS
		Sürücü Servisleri Portalları
		· Sosyal Medya
Sağlık	· Yaşlanan nüfus	**Teletıp ve Uzaktan Hasta İzleme**
	· Artan maliyetler	· Zeki/İnternete bağlı tıbbi cihazlar (glukometre, puls oksimetre, tansiyon)
	· Kronik hastalıkların yaygınlığı	**Bakım Yönetim Sistemleri (Uzaktan Bakım)**
	· Alandaki uzman eksikliği	· Hasta Portalları
		· Elektronik Medikal
		Kayıtlar
Enerji	· Karbon ayak izi/sürdürülebilirlik	**Zeki Şebekeler**
	· Enerji Maliyeti	· "Smart Meters"
		· Şebeke Sensörleri
		· "Meter Data Management"
		· Şebeke Yönetim Sistemleri
		Enerji Verimliliği ve Talep Karşılama
		· Zeki Ev Uygulamaları
		· CRM sistemleri, Analitik ve Müşteri Portalları
Perakende	· Artan enerji-ulaşım maliyetleri	**Gerçek-Zamanlı Mağaza içi Sistemler**
	· Karbon ayak izi/sürdürülebilirlik	· PoS/Self-Checkout
	· Müşteri odaklılık	· Kiosk
		· Mobil/Kişisel alışveriş asistanları
		Merkezi Arka Ofis Sistemleri
		· E-Ticaret, Sosyal Medya
		· İş Zekası/Müşteri Analitiği
		Enerji Yönetim Sistemleri
		· HVAC
		· Işıklandırma-Soğutma
Finansal Hizmetler	· Az sayıda ve küçük şubelerin çok aktivite yerine getirme gerekliliği	**Zeki Bankacılık Sistemleri**
	· Şubelerde etkin olarak dağıtılmamış uzmanlar (sağlık, kredi, sigorta vb.)	· "Personal Teller Machine" (Uzmanlarla gerçek zamanlı yüz yüze iletişim-self-service kiosk)
		· Mobil Bankacılık (Zeki Uyarı, ATM ve ev bilgisayarlarıyla entegrasyon)
		· Arka Ofis Sistemleri (Dinamik fiyatlandırma, Sahtecilik analitiği)
		· Güvenlik

- Operasyonel ve yönetsel bağımsızlık
- Coğrafi dağılım
- Sonradan ortaya çıkan özellikler (emergence)
- Evrimsel gelişim
- Bileşen sistemlerin heterojen olması

Maier (1998) tarafından belirtilen bu beş kriter "Systems of Systems" konseptinin temel özellikleri olarak belirlenmiştir. Bunlardan "Emergent behaviour" olarak tanımlanan özellik sistemlerin bir araya gelmesinden sonra ortaya çıkan özellikler olarak ifade edilmektedir. Sommerville (2014) bu özelliklere veri yoğunluğunu da eklemektedir. SoS yazılımları büyüklük olarak bileşen sistem yazılımlarından çok daha büyük olmakla birlikte bu yazılımlar çok büyük hacimdeki verilere dayanmakta ve bunları yönetmektedirler. Dersin ve Transport (2015) tarafından ise kaotik davranış, birbirine bağlı ve ardışık arıza potansiyelleri, kendi kendini organize edebilme özelliği, zayıf bağlaşım ve karmaşık süreçler bu sistemlerle ilişkili dikkat edilmesi gereken hususlar ve sistem özellikleri olarak belirtilmektedir. Bu bağlamda tüm bu özellikler "Systems of Systems" konsepti içerisindeki sistemlerin çok daha karmaşık olduğunu ve geçmişte sistem karmaşıklığını ölçmek ve

değerlendirmek için kullanılan metriklerin artık bu sistemler bağlamında değerlendirilerek baştan yorumlanması gerekliliğini ortaya koymaktadır.

Sistemlerin birbiriyle ilişkisi dinamik karmaşıklığı doğurmaktadır. Sistemler kullanılmaya başlandıktan sonra sistem karmaşıklığının yanında süreçlerin karmaşık olma durumu da yönetilmesi gereken ayrı bir husustur. Somerville (2014) bu sosyo-teknik sistemlere ait farklı karmaşıklık türlerinden bahsetmektedir:

- Teknik Karmaşıklık
- Yönetsel Karmaşıklık
- Yönetişim (Governance Complexity) Karmaşıklığı

Yönetim ve yönetişime ilişkin karmaşık olma durumu benzer olmasına rağmen aynı kavramlar değildir. Yönetim karmaşıklığı operasyoneldir ve sistemlerle ilgili ne yapılıp yapılamayacağıyla ilgilenir. Yönetişim ise organizasyonlardaki üst düzey karar alma süreçleriyle ilgilenir (Somerville, 2014).

Karmaşıklık kavramına sistem mühendisliği perspektifinden bakıldığında çok sayıda farklı karmaşıklık türleri olduğu görülmektedir. SoS yaklaşımında karmaşıklığa ilişkin incelemeler yapmak adına öncelikle sistem mühendisliği bakış açısından yorumlanan bu karmaşıklık türlerini incelemek ve Gömülü Zeki Sistemlerin oluşturduğu SoS ile ne şekilde paralellik gösterdiğini ortaya koymak önemlidir. Bu yaklaşım karmaşıklığa kategorik olarak yaklaşmak ve gelecekte bu sistemlerin karmaşıklığının ölçülmesi için metrikler geliştirilmesi adına önemlidir.

Sheard (2013) tanımlanan farklı karmaşıklık kavramlarını organize ederek karmaşıklık topolojisine daha organize bir yaklaşım getirmiştir. Bu bağlamda karmaşıklık kavramına 5 boyuttan yaklaşılarak, sistem ve çevresinin karmaşık öğeleri; Sistemin kendisi, sistemleri gerçekleştiren projeler, çevre (sistemin etkileşimde olduğu yazılım, donanım ve diğer yapılar), sosyo-politik sistem ve bilişsellik (cognition) olarak belirlenmiştir. Bunlardan sistem öğesi sistemi oluşturan bileşenler ve onların etkileşiminden meydana gelmektedir. Birbiriyle bağlantılı olan elemanlar söz konusu olduğunda sisteme ait yeni davranış biçimleri ortaya çıkmaktadır. Belirsizliğin olduğu bu tarz durumlarda kavrama ve insan faktörü devreye girerek sübjektif veya bilişsel karmaşıklık ortaya çıkmaktadır. İnsanların kavrama ve muhakeme yetisinin sınırlı olmasından yola çıkıldığında çevredeki olaylar veya beklenmedik sistem davranışlarını anlama veya yorumlamada güçlükler görülebilmektedir. Belirsizliğin arttığı bu durumlar tahmin edilebilirliği azaltarak risklere neden olmaktadır.

Tüm bu karmaşıklık kavramları göz önüne alındığında gömülü zeki sistemler ve onların giderek birbiriyle bağlantılı sistemler haline gelmesi bu sistemlere ilişkin farklı karmaşıklık kavramlarına yönelik bütüncül bir bakış açısı gerektirmektedir. Bu sistemlerin bir araya gelmesi ve yeni teknolojiler yeni problemleri de ortaya çıkartmıştır. Bu sebeple sadece bir sistemi meydana getiren elemanların karmaşıklığını anlamaya ve ölçmeye yönelik yaklaşımlar günümüzde yeterli kalmamaktadır.

2.7 Gelecekteki Gömülü Sistemlerde Ön Plana Çıkan Kritik Konular

Bu inceleme ve açıklamalardan sonra bizce önümüzdeki 5 yıl içerisinde görülmesi muhtemel etkiler ve dikkat edilmesi gereken temel hususlar şu şekilde sunulabilir. Endüstriyel ve toplumsal talep sistemlerden talep edilen fonksiyonların ve servislerin çeşitliliğini artırmaya devam edecektir. Özellikle yazılım açısından sistem tasarımı ve geliştirmede karmaşıklık kavramıyla başa çıkabilecek ve bunun

kontrolünü sağlayacak güncel teknolojiler günümüz itibariyle sınırlı kalmaktadır. Gelecekteki sistemlerin tasarımı ve karmaşık sistemleri geliştirmek için yenilikçi yaklaşımlara ihtiyaç duyulacaktır (IDC, 2012).

Gelecekteki gömülü sistemler karakteristiği ve bu sistemlerde karmaşıklıkla başa çıkılması anlamında ön plana çıkan güncel kritik konular aşağıdaki tabloda özetlenmektedir.

Tablo 2.4: Gelecekte Gömülü Sistemler ve Kritik Araştırma Konuları (IDC, 2012)

Teknolojik ve Bilimsel Zorluklar	**Kritik Araştırma Konuları**
Kritiklik zorunlulukları (güvenlik, emniyet ve sertifikasyon)	Güvenli ve emniyetli sistemler için kod yazılımı & doğrulama, simülasyon & sertifikasyon
Dağıtılmış mimari ve otonom sistemler	Sistem analizi ve programlama araçları, dağıtılmış mimari tasarımı
Kusursuz bağlantı	Servis odaklı mimari geliştirme ve internet bağlantısı
İnsan makine ara yüzü	HMI ara yüz geliştirme mekanizmaları ve gerçek-zamanlı simülasyon
Çok çekirdekli yapı ve sanallaştırma	Paralel mimariye geçiş, mimari tasarımı ve sertifikasyon
Enerji yönetimi	Enerji yönetimi için yazılım geliştirme araçları, sistemlere enerji yönetiminin entegre edilmesi

Sistem modelleme, kontrol, doğrulama, sanallaştırma, veri madenciliği gibi teknikler ve çok disiplinli uygulamalar için ortak bir dil geliştirme gereksinimi ortaya çıkmaktadır. Toplumsal yönden bakıldığında ise toplumsal kullanıcı kabulü konularına dikkat etmekte fayda vardır. Bu amaçla veri güvenliği ve entegrasyonu, güvenliğin öncelikli olduğu (safety- critical systems) için güvenliğin garantilenmesi ve standartlara göre sertifikasyonun mümkün kılınması gibi hususlara verilmesi gereken önem artmaktadır (EU Comission, 2012). Ayrıca, sistemler birçok farklı fonksiyonu gerçekleştirdiğinde ve farklı hizmetler sunduğunda muhtemel sistem açıkları kritik bir öneme sahip olmaktadır (IDC, 2012). Sonuç olarak, SoS için gerekli olan iş modellerinin, mimari platformlarının, teorilerin ve endüstrinin potansiyel SoS faydalarını yorumlayabilecekleri karar destek mekanizmalarının geliştirilmesi gelecekte bu sistemlerin potansiyellerini gerçekleştirebilmeleri ve önemli sorunların çözüm bulması adına ele alınmalıdır.

Günümüzde sistem güvenilirliğiyle ilgili analizlerin birçoğu bileşenlerin birbirinden bağımsız olduğu varsayımına dayanmaktadır. Sistem güvenilirliği ve emniyet analizlerinde birbirine bağlı ve ardışık sistematik hataların ele alınması gerekmektedir ve özellikle hata kaynaklarının modellenmesi gereksinimi ortaya çıkmaktadır. Bu bağlamda nükleer enerji santralleri alanında çeşitli çalışmalar mevcutken, "systems of systems" kapsamındaki yeni ortaya çıkan sistem davranışlarının modellenmesi, güvenilirlik ve emniyet analizleri için yeni yöntem ve yaklaşımlar ortaya atılamamıştır (Dersin ve Transport, 2015). Özetle, karmaşıklaşan iş dünyası ve sosyal dünya, teknolojideki gelişmeler hem daha büyük ölçekli ve güvenliğin öncelikli olduğu sistem uygulamalarını hem de zeki sistemler ile bunları yönetme ihtiyacını arttırmaktadır. Bu nedenle bu sistemlerde karmaşıklığı ortaya çıkaran faktörler SoS ve diğer interdisipliner yaklaşımlarla detaylı olarak irdelenmeli, karmaşıklığın kök nedenleri ve yol açtığı sonuçlar anlaşılmalı ve optimum karmaşıklık düzeyinin tespiti ve bunun uygulanabilmesi için gerekli olan yöntem ve tekniklere yönelinmelidir. Aksi takdirde hiç düşünülemeyen ve birbirini tetikleyebilen yeni kazaların meydana gelmesi kaçınılmaz olacak ve sistemlerin her geçen gün büyüyüen boyutu nedeniyle kayıplarda artacaktır.

2.8 Kaynakça

Bernat, G., Burns, A., & Liamosi, A. (2001). Weakly hard real-time systems. IEEE transactions on Computers, 50(4), 308-321.

Coskun, E. (2001). Complexity and Its Impacts in Embedded Intelligent Real-Time Sistem. A Thesis Submitted to Renssealer Polytechnic Institute, New York

Coskun, E., & Grabowski, M. (2005). Software complexity and its impacts in embedded intelligent real time systems. Journal of Systems and Software, 78(2), 128-145.

Dersin, P. & Transport, A. (2014). Systems of Systems. IEEE-Reliablity Society. Technical Committee on 'Sistem of Systems'-WHITE PAPER. Available at http://bit.ly/2l0HFH4

EETimes, 2010. Emerging trends in embedded systems and their applications. Available at http://ubm.io/2kV8tbr

European Commission (2012). Direction in Systems of Systems Engineering. Report from THA Workshop on Synergies among Projects and Directions in Advanced Systems Engineering held on 04th and 05th July 2012 in Brussels, Belgium. Available at http://bit.ly/2yRglAO

Hallmans, D. On evolution of an embedded system architecture. Hwang, D., Schaumont, P. & Verbauwhede, I. (2006). Securing Embedded Systems. EIII Securiy & Privacy, 4(2), 40-49.

IDC (2012). Final Study Report: Design of Future Embedded Systems (SMART 2009/0063). Available at http://www.cordis.europa.eu/fp7/ict/embedded-systems-engineering/documents/idc-study-final-report.pdf

Jaalinoja, J. (2004). Requirements implementation in Embedded Software Development. VTT Publications 526. Available at http://www.vtt.fi/inf/pdf/publications/2004/P526.pdf

Karcanias, N. (2015). Systems Complexity: The Paradigm of Systems of Systems. Complexity Science Workshop 18, 19 June 2015 City University. Available at http://bit.ly/2xN9qZE

Kazman, R. (1998, March). Assessing architectural complexity. In Software Maintenance and Reengineering, 1998. Proceedings of the Second Euromicro Conference on (pp. 104-112). IEEE.

Leveson, N. (1991). Software safety in embedded computer systems. Communications of the ACM, 34(2), 34-46. Lukkien, J. (2015). A Summary on Systems of Systems Engineering. Available at http://www.win.tue.nl/ johanl/docs/SoS

Maier, W. (1998). Architecting principles for systems of systems, Systems Engineering, 1(4), 267-284

Pathan, R. M. (2010). Scheduling Algorithms for Fault-Tolerant Real-Time Systems, Chalmers University of Technology.

Poggio, T. & Stringa, L. (1992). A Project for an Intelligent System: Vision and Learning. International Journal of Quantum Chemistry, 42, 727-739.

Rommel, C. & Wallek, J. (2017). Where and When. Webinar: IoT Evolution in Embedded Systems. Availabel at https://polarion.plm.automation.siemens.com/events/webinar/iot-evolution-in-embedded-systems

Samad, T., & Parisini, T. (2011). Systems of systems. The Impact of Control Technology, 12(1), 175-183.

Shao, J., & Wang, Y. (2003). A new measure of software complexity based on cognitive weights. Canadian Journal of Electrical and Computer Engineering, 28(2), 69-74.

Somerville, I. (2014). Systems of Systems, Chapter 20. Available at http://bit.ly/2yxAwTn

Stankovic, J. (1988). Misconceptions about real-time computing: A serious problem for next generation systems. IEEEComputer 21, 10, 10-19.

Stankovic, J.A. (1996). Real-Time and Embedded Systems. ACM Computing Surveys, 28(1). Available at http://delivery.acm.org Wordpress, 2012. Evolution of Embedded Systems. Available at https://purpleleap.wordpress.com/2012/04/27/evolution-of-embedded-systems/

Yacov, Y. (1982). "Modeling Of Large-scale Systems in a Hierarchical-Multiobjective Framework" Studies in Management Science and Systems v. 7 pp: 1-17, IEEE Comm on Large-scale Systems, New York, NY, USA

Zio, E. (2014). Integrated deterministic and probabilistic safety assessment: concepts, challenges, research directions. Nuclear Engineering and Design, 280, 413-419.

2.9 Yazarlar Hakkında

Prof.Dr. Erman Coşkun Mühendislik Bilimleri ve Bilişim Sistemleri alanındaki doktora ve Mühendislik Yönetimi alanındaki yüksek lisans derecelerini ABD New York eyaletinde bulunan Rensselaer Polytechnic Institute'dan elde etmiştir. Ayrıca Pace University'den Yönetim Bilimleri ve Sayısal Yöntemler alanında MBA derecesi olan Prof.Dr. Erman Coşkun Sakarya Üniversitesi'nde Yönetim Bilişim Sistemleri Bölüm Başkanlığı'nı yürütmektedir. Yönetim Bilişim Sistemleri ve Yöneylem Araştırması alanlarında profesörlüğü olan Prof.Dr. Erman Coşkun'un, 17 yıllık akademik kariyeri Birleşik Devletler, Türkiye, Kıbrıs ve Suudi Arabistan gibi ülkelerdeki çeşitli görevleri kapsamaktadır. Türkiye ve Birleşik Devletler'de Yönetim Bilişim Sistemleri, Tedarik Zinciri ve Lojistik Yönetimi ile Sayısal Yöntemler ve Yönetim Bilimleri alanlarında dersler vermiş, LeMoyne College ve Sakarya Üniversitesi'nden birçok araştırma ve ders verme başarı ödülü almıştır. Araştırmaları İş Analitiği, İş Zekası, Kurumsal Kaynak Planlama, Gömülü Zeki Sistemler ile Afet ve Kriz Yönetimi alanlarına odaklanarak kamu ve özel sektör için birçok alanda danışmanlık ve proje yürütücülüğü yapmıştır.

Arş. Gör. Büşra Alma Elektrik-Elektronik Mühendisliği alanındaki lisansını Bilkent Üniversitesi'nde tamamladıktan sonra İngiltere'deki Kingston University'de Engineering Projects & Systems Management alanında yüksek lisans yapmıştır. Yüksek lisansını distinction derecesiyle tamamlayarak akademik kariyerine Sakarya Üniversitesi'nde devam etmiştir. Mevcut durumda Sakarya Üniversitesi Yönetim Bilişim Sistemleri Bölümü'nde Yönetim Bilişim Sistemleri alanındaki doktorasını sürdürmekte ve Araştırma Görevlisi olarak çalışmaktadır. Araştırmada IT Proje Yönetimi, İş Analitiği, Karar Destek Sistemleri, İş Zekası, Gömülü Zeki Sistemler ve İnsan Bilgisayar Etkileşimi alanlarına odaklanarak çeşitli TÜBİTAK projelerinde görev almıştır.

3. Bulanık Mantık ve Matlab Uygulamaları

Bulanık Mantık ve Matlab Uygulamaları

Yrd. Doç. Dr. Alper KİRAZ

3.1 Giriş

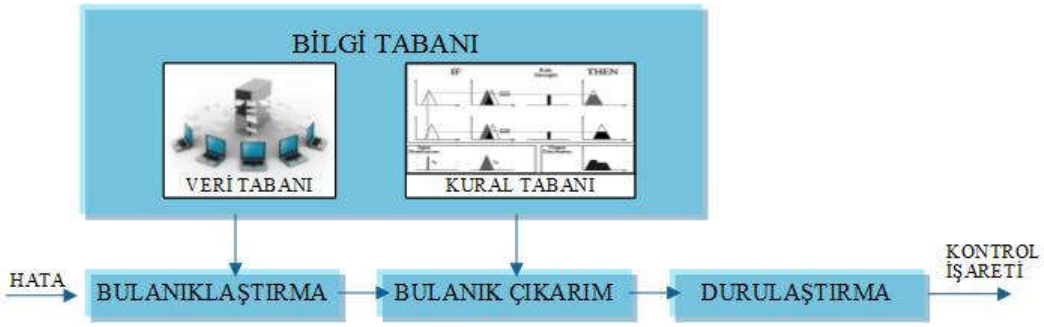
Bulanık mantık insan düşüncesini temel alan ve bu düşünceyi matematiksel fonksiyonlara dönüştürerek işlem yapan, hesaplamalı bir yaklaşımdır. Lotfi Asker Zadeh (Lütfi Aliasker Zade) ilk olarak 1965 yılındaki makalesiyle bu yaklaşımın temelini oluşturmuş bir bilim adamıdır. İkili mantık olarak da bilinen Aristo mantığı (klasik mantık) 0 ve 1'lerden oluşmakta ve bir elemanın ya o kümeye ait olduğunu ya da olmadığını göstermektedir. Bulanık mantıkta ise her bir eleman [0,1] arasında üyelik dereceleri alarak aynı anda birden fazla kümeye ait olabilmektedir.

Bulanık mantık ile modeller kurulurken girdilerin tümünü çıktıların tümüyle ilişkilendirerek küme ve kuralların tanımlaması yapılmaktadır. Bu nedenle de bulanık modellerin çalışma sistematiği matematiksel bir neden-sonuç fonksiyonunun çalışma şekline benzemektedir.

Bulanık mantığın genel özelliklerine bakıldığında;

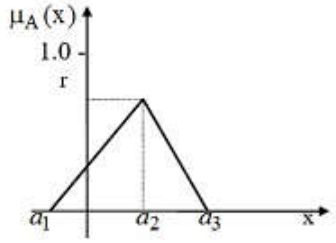
- Kesin değerlere dayanan düşünce yerine, yaklaşık düşünme kullanılmaktadır,
- Bilgi sözel ifadeler (az, çok, sıcak, soğuk vb.) şeklindedir,
- Kümelerin ağırlıkları 0-1 aralığında ifade edilir,
- Durulaştırma işlemi, sözel ifadelerin birbiri ile arasında tanımlanan kurallar ile gerçekleşir,
- Matematiksel modelin kurulmasının zor ve karmaşık olduğu sistemlerde kullanılabilir [1].

Bulanık mantık denetleyici, bulanıklaştırma, bulanık çıkarım, durulama ve bilgi tabanı olmak üzere dört temel bileşenden oluşmakta olup temel bir bulanık mantık denetleyici yapısı Şekil 1'de sunulmaktadır.



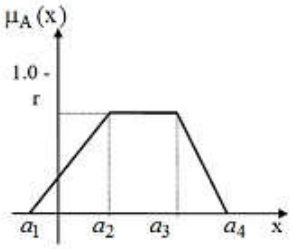
Şekil 3.1: Sinyal işleme akış diyagramı

Bulanık mantık modeli kurulacak probleme özgü giriş bilgilerinin dilsel değişkenler kullanılarak ifade edilmesi ve bulanık mantık bilgisine dönüştürülmesi işlemine bulanıklaştırma adı verilmektedir. Bulanıklaştırma işlemi sonrasında oluşan dilsel değişkenler, üyelik dereceleri alarak üçgensel, yamuk, S şekilli, çan şekilli ve problemin yapısına özgü daha pek çok geometrik şekillerle temsil edilmektedir. Üçgensel bir üyelik fonksiyonu örneği Şekil 2'de, ikizkenar yamuk üyelik fonksiyonu örneği Şekil 3'te, S şekilli bir üyelik fonksiyonu örneği Şekil 4'te ve çan şekilli bir üyelik fonksiyonu örneği Şekil 5'te formülleri ile birlikte sunulmaktadır [4].



$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1} r, & \text{eğer } a_1 \le x \le a_2 \\ \dfrac{a_3-x}{a_3-a_2} r, & \text{eğer } a_2 \le x \le a_3 \\ 0, & \text{diğer hallerde} \end{cases}$$

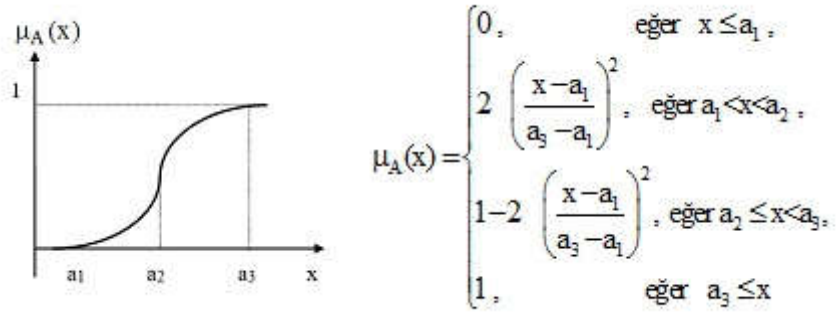
Şekil 3.2: Üçgensel üyelik fonksiyonu örneği



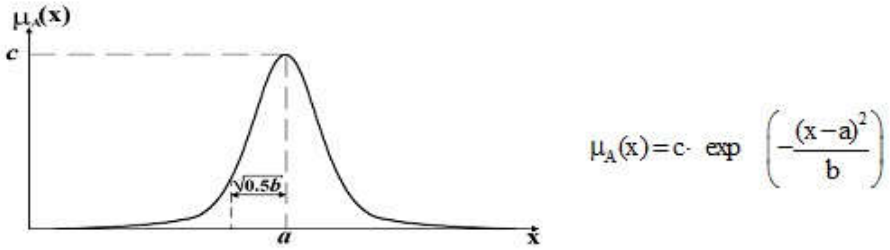
$$\mu_A(x) = \begin{cases} \dfrac{x-a_1}{a_2-a_1} r, & \text{eğer } a_1 \le x \le a_2 \\ r, & \text{eğer } a_2 \le x \le a_3 \\ \dfrac{a_4-x}{a_4-a_3} r, & \text{eğer } a_3 \le x \le a_4 \\ 0, & \text{diğer hallerde} \end{cases}$$

Şekil 3.3: İkizkenar yamuk üyelik fonksiyonu örneği

Çıkarım ünitesi karar verme işlemlerinde, bilgi tabanına gidip, veri tabanından üyelik fonksiyonlarıyla ilgili bilgileri, kural tabanından ise değişik giriş değerleri için tespit edilmiş olan kontrol çıkışları bilgisini alır. Bu bakımdan bilgi tabanı ve çıkarım ünitesi sürekli ilişki halindedir [5].



Şekil 3.4: S şekilli üyelik fonksiyonu örneği



Şekil 3.5: Çan şekilli üyelik fonksiyonu örneği

Veri tabanı, üyelik fonksiyonlarının tespit edilmesi için yapılan ön çalışmalar ile son hali belli olmuş üyelik fonksiyonlarının sınır ve eğim bilgilerini içerir. Kural tabanı, kontrol kurallarının saklandığı veri tabanıdır. Bir sistem için kural tabanı geliştirilirken, sistem çıkışını etkileyebilecek giriş değerleri tespit edilmelidir. Bulanık kontrol kuralları genellikle bir uzman bilgisinden türetilir. Bulanık kurallar oluşturulurken sistem giriş ve çıkışı "ve/veya" işlemleri kullanılarak ve "eğer-ise-o halde" biçimine sahip koşul cümleleriyle birbirine bağlanır. Bu koşul cümlelerinin her biri bir kural olarak isimlendirilir. Bulanıklaştırma biriminden gelen değerler, kural tabanındaki kurallar üzerinde uygulanarak bulanık sonuçlar üretilmektedir. İlk olarak, her bir giriş değerinin ne oranda hangi üyelik kümesine ait olduğu saptanmaktadır. Bu değerler kural tablosuna yerleştirilerek uygun çıkışlar elde edilmektedir. Bulanık mantık kuralları kural içerisindeki bileştiricilerin anlamlarının yorumlanması ile hesaplanmaktadır.

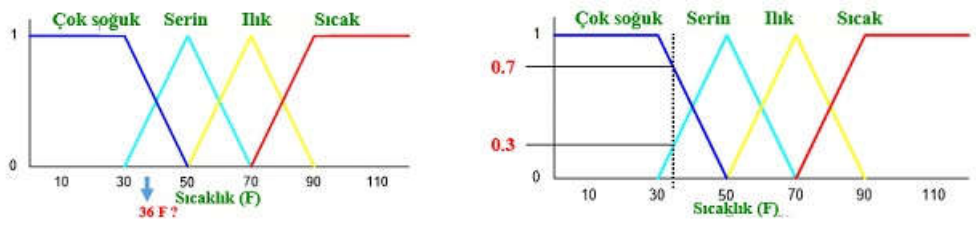
Son olarak durulaştırma, olasılık dağılımını en iyi gösteren, bulanık olmayan denetim etkinliği elde etme süreci olarak tanımlanır. Bu sayede kontrol ünitesinden gelen sayılar fiziksel ve kesin sayılara dönüştürülmektedir [6]. Uygulamanın özelliklerine dayalı olarak durulaştırma sürecinde ortalama maksimum, ilk maksimum, son maksimum, ağırlıklı ortalama ve ağırlık merkezi gibi yöntemler kullanılmaktadır.

- Uzmanlar sistemi tam olarak anlayamadığında,
- Farklı uzmanların görüş ayrılığına düştüğü durumlarda,
- Bilgi dilsel değişkenlerle ifade edilemediğinde,

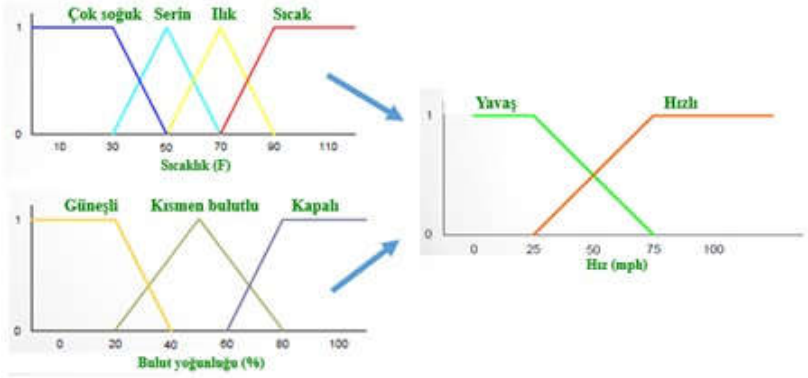
- Objektif analizlerin yapılamadığı karmaşık içgüdüsel durumlarda bulanık mantık yaklaşımının tercih edilmemesi tavsiye edilmektedir.

Bulanık mantık ile kontrol konusundaki ilk uygulama 1974'de Mamdani [7] tarafından buhar makinesinin kontrolü ile gerçekleştirilmiştir. Bu uygulama sayesinde Zadeh'in dilbilimsel kural yaklaşımının bilgisayarlar tarafından kolaylıkla işlenebilen bir formda sağlanabileceğini Mamdani ortaya koymuştur. Bulanık mantık ile kontrol ilk kez Danimarka'da endüstriyel bir prosese olan çimento fırının kontrolüne 1982'de uygulanmıştır [8]. Japon araştırmacıların yeni teknolojiler üzerine olan yaklaşımları sayesinde bulanık mantık çok hızlı bir şekilde gelişme göstermiştir. Bulanık mantık günümüzde elektronik kontrol sistemleri, otomotiv endüstrisi fren sistemleri, proses kontrol, proses planlama ve ev elektroniği gibi birçok alanda uygulanmıştır. Her gün kullanılan ev aletlerinin kontrol edilmesinde bulanık mantık modellerinin uygulanması ile birlikte önemli ölçüde enerji ve zaman tasarrufu sağlanmaktadır [9].

Sıcaklık ile ilgili aşağıdaki örneği inceleyelim. Burada üçgensel ve yamuk üyelik fonksiyonları tipi ile 4 adet dilsel değişken kullanılarak bir sıcaklık üyelik fonksiyonu tanımlanmıştır.



Şekil 3.6: Sıcaklık üyelik fonksiyonu



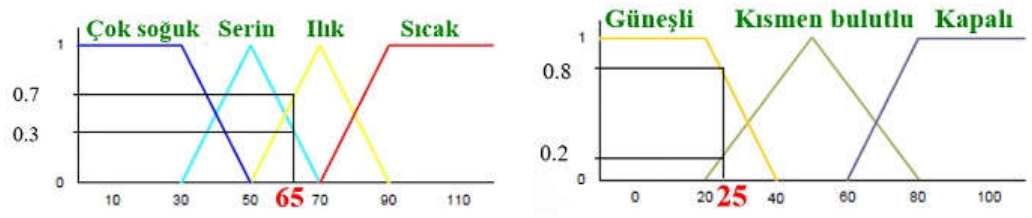
Şekil 3.7: İki girdili bir çıktılı hız kontrol bulanık modeli

Sıcaklığın 65F olduğu ve Bulut yoğunluğunun %25 olduğu durumda ve

Kural 1:Eğer hava ılık ve güneşli ise hızlı sür

Kural 2:Eğer hava serin ve kısmen bulutlu ise yavaş sür kuralları kapsamında ilk maksimum durulaştırma metodu ile aracın hızını belirleyelim. Burada öncelikle 65F sıcaklıkta ve %25 bulut yoğunluğunda girdilerin üyelik fonksiyonlarını nasıl temsil ettiğini inceleyelim.

Şekil 8'den de anlaşılacağı üzere 65F sıcaklık girdisi; 0,3 üyelik derecesi ile "Serin", 0,7 üyelik derecesi ile "Ilık" havayı temsil etmektedir. %25 Bulut yoğunluğu girdisi ise; 0,2 üyelik derecesi ile



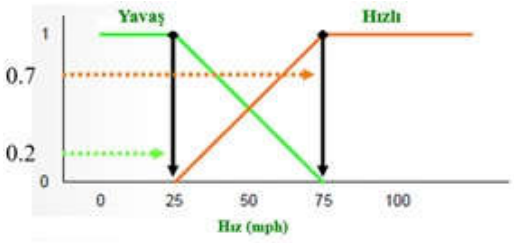
Şekil 3.8: 65F Sıcaklık ve %25 Bulut yoğunluğu üyelik dereceleri

"Kısmen bulutlu", 0,8 üyelik derecesi ile "Güneşli" havayı temsil etmektedir. Bu durumda;

Kural 1: Eğer hava ılık ve güneşli ise hızlı sür (0,7 ^0,8 = 0,7) kuralı "ve" operatörü ile yazıldığından küçük olan değer hız değerine (0,7) eşit olacaktır.

Kural 2:Eğer hava serin ve kısmen bulutlu ise yavaş sür (0,2 ^0,3 = 0,2) kuralı "ve" operatörü ile yazıldığından küçük olan değer hız değerine (0,2) eşit olacaktır. İlk maksimum kuralına göre Şekil 9'da belirtilen hız parametresinin matematiksel karşılığı olarak;

Hız= (0.2*25+0.7*75)/(0.2 + 0.7) =63.8 mph olarak hesaplanır.



Şekil 3.9: Hız çıktı değerinin durulaştırılması

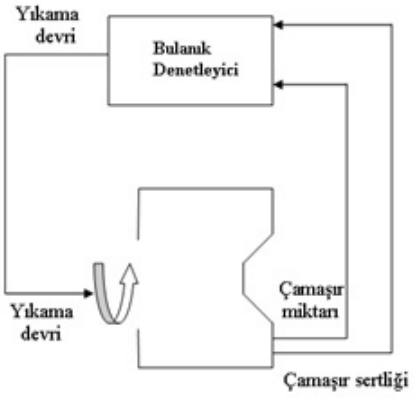
Buraya kadar olan anlatımda bulanık mantığın temel yapıtaşlarını oluşturan elemanlar açıklanarak ve basit iki örnekle desteklenerek ele alınmıştır. Tüm bu kavramların bir örnek üzerinden MATLAB yazılımı Bulanık Mantık Araç Kutusu (Toolbox) kullanılarak nasıl uygulandığını literatürde verilen örneklerde sıklıkla karşılaşılan Çamaşır Makinesi örneği ile açıklayalım.

3.2 Çamaşır Makinesi Yıkama Devri Kontrolünün Bulanık Modeli ve Matlab Uygulaması

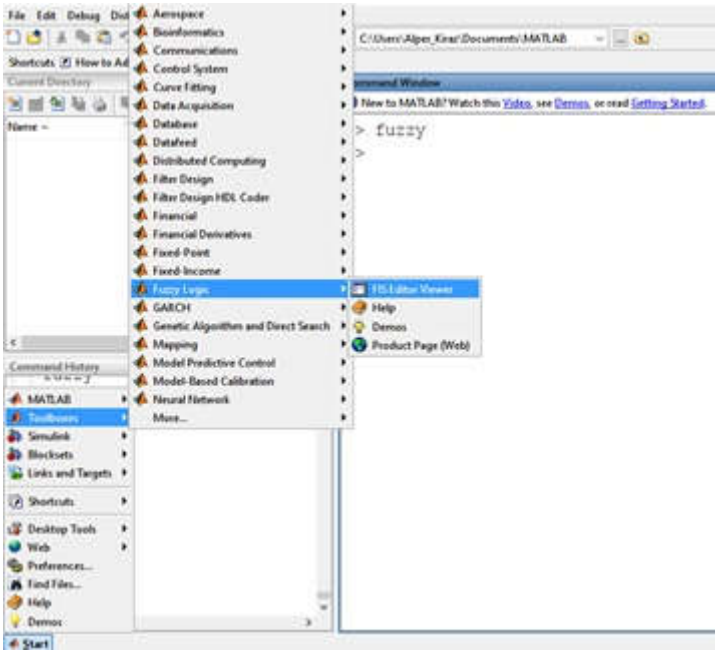
Çamaşır makinesinin yıkama devrini (YD) etkileyen iki girdi parametresinin çamaşır miktarı (ÇM) ve çamaşırın sertliği (ÇS) olduğunu varsayalım. Burada amaç zamandan ve enerjiden tasarruf sağlanacak optimum yıkama devrinin belirlenmesidir. Geliştirilecek bulanık mantık modeli için yıkama devri bulanık denetleyici tasarımı Şekil 10'da sunulmaktadır.

Matlab'ta bulanık mantık modeli oluşturmak için "Command Window" ekranına "fuzzy" yazılabilir veya sol alt kısımdaki "Start" butonu tıklanarak "Toolboxes" sekmesinden "Fuzzy logic" araç kutusu seçilerek "FIS Editor Viewer" işlemleri sırasıyla yapılabilir (Şekil 11).

Bulanık mantık araç kutusu ile öncelikle ÇS, ÇM ve YD değişkenleri girdi ve çıktı olarak tanımlanıp üyelik fonksiyonları oluşturulur. Araç kutusunda varsayılan olarak 1 girdi ve 1 çıktı değişkeni bulunmaktadır, çamaşır makinesi bulanık modeli için 1 girdi parametresi ilave etmemiz gerekmektedir (Şekil 12). Bu pencerede ayrıca ve/veya işlemcilerinin bulanık kuralları için hangi

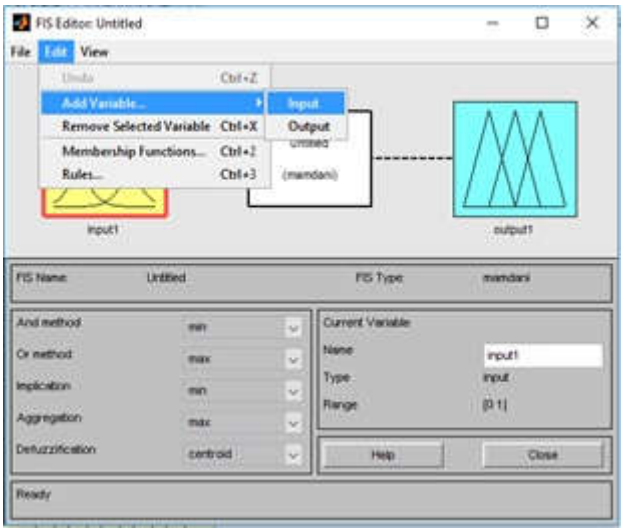


Şekil 3.10: Yıkama devri bulanık denetleyici tasarımı



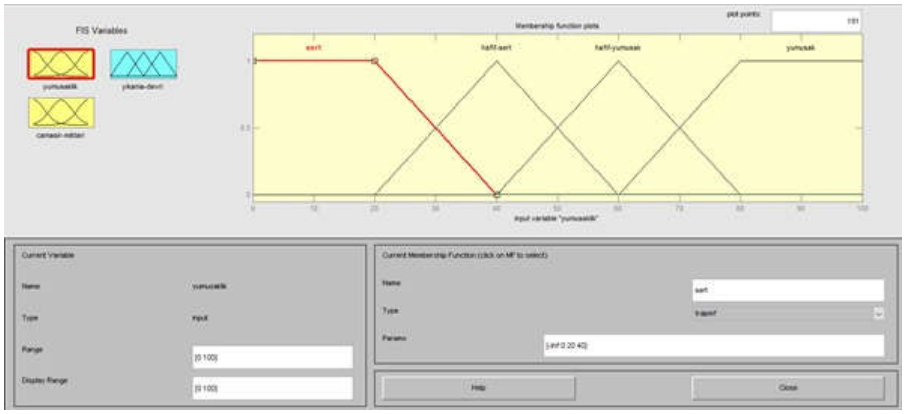
Şekil 3.11: Matlab bulanık mantık araç kutusu

fonksiyonların kullanılacağı (min, max vb.), bulanık sonuçların birleştirilmesinde kullanılacak fonksiyon ve durulaştırma işleminin tipi gibi parametreler seçilebilmektedir.



Şekil 3.12: Değişken ilave edilmesi ve bulanık modelin özellikleri

Daha sonra ÇY, ÇM ve YD değişkenleri için üyelik fonksiyonları tanımlanır. Eğer öncesinde değişkenlerle ilgili veriler var ise SPSS, NCSS, Matlab vb. yazılımlar aracılığı ile bu veri setlerinin üyelik fonksiyonlarının adedi ve üçgensel, yamuk, sigmoid vb. fonksiyonları belirlenebilir. Veri yok ise uzman görüşüne ile bu değişkenler tanımlanır. Şekil 13'ten de anlaşılacağı üzere ÇY değişkeni için uzman görüşüne dayalı olarak 5 adet üyelik fonksiyonu tanımlanmıştır. Bu üyelik fonksiyonu sert, hafif sert, hafif yumuşak ve yumuşak olmak üzere [0 100] aralığında dilsel değişkenler kullanılarak oluşturuluştur. Sert ve yumuşak üyelik fonksiyonları için yamuk (trapmf), hafif sert ve hafif yumuşak üyelik fonksiyonları için ise üçgensel (trimf) tercih edilmiştir. Her bir değişken için seçilen üyelik fonksiyonları ve bunların değişim aralıkları Tablo 1'de sunulmaktadır.



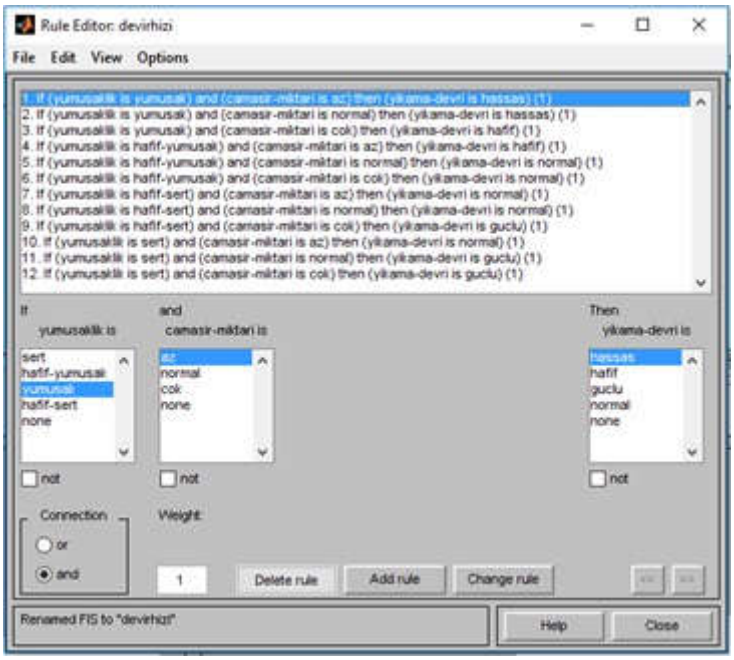
Şekil 3.13: ÇY değişkeni için üyelik fonksiyonu tanımlama

Üyelik fonksiyonları tanımlandıktan sonra Edit → Rules seçeneği tıklanarak bulanık mantık modelinin kuralları girilmektedir. Bu model için uzman görüşü alınarak 12 adet kural girilmiştir.

Tablo 3.1: ÇY değişkeni için üyelik fonksiyonu tanımlama

Değişkenler	ÇY	ÇM	YD
Üyelik fonksiyonları ve aralıkları	sert [-Inf 0 20 40] [yamuk] hafif sert [20 40 60] [üçgensel] hafif yumuşak [40 60 80] [üçgensel] yumuşak [60 80 100 Inf] [yamuk]	az [-Inf 0 1 3] [yamuk] normal [1 3 5] [üçgensel] çok [3 5 6 Inf] [yamuk]	hassas [-Inf 0 400 600] [yamuk] hafif [400 600 800] [üçgensel] normal [600 800 1000] [üçgensel] güçlü [801 1000 1200 Inf] [yamuk]

Örneğin ilk kural "Eğer yumuşaklık yumuşak ise ve çamaşır miktarı az ise, o halde yıkama devri hassas" şeklindedir. Diğer kurallar da bu Şekil 14'ün alt kısmında kalan alan sayesinde kullanıcı tarafından kolayca seçilerek ve/veya operatörleri de kullanılarak "Add rule" seçeneği ile sistemde tanımlanmaktadır. Kuralların tamamı girildikten sonra View → Surface seçeneği ile iki girdi değişkeninin YD çıktı değişkenini nasıl etkilediği (girdi çıktı ilişkisi) 3 boyutlu grafik ile gözlemlenebilmektedir. Bu grafikle ilgili bir görsel Şekil 15'te sunulmaktadır.

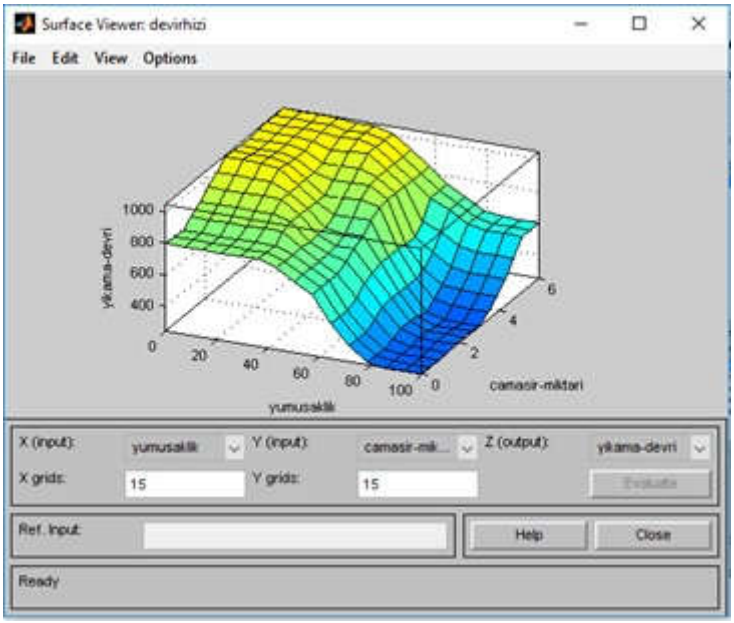


Şekil 3.14: Bulanık kural tabanı

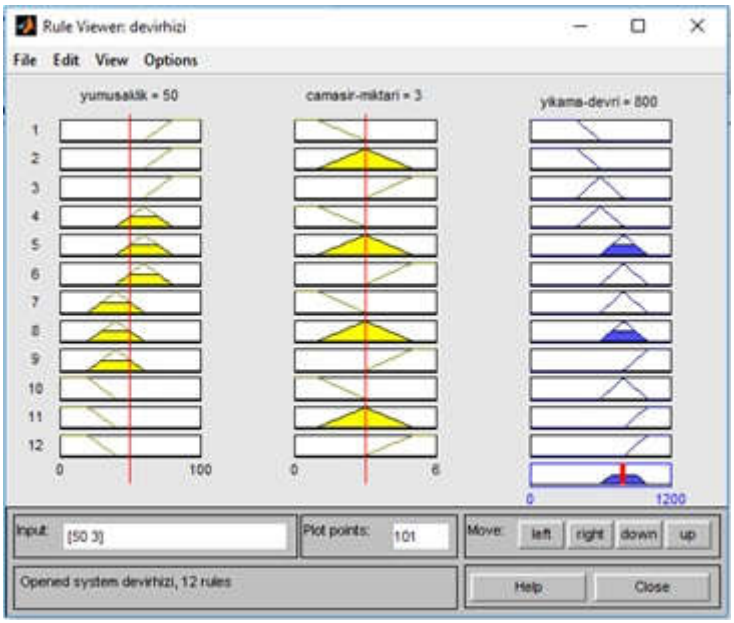
Durulaştırılmış çıktı değerlerini gözlemlemek, girdi parametrelerindeki değişimin çıktı parametresini nasıl etkilediğini sayısal olarak elde edebilmek için yine View seçeneğinden Rules sekmesi seçilerek gözlemlenebilir. Örneğin Şekil 16'daki örnek gösterimde "Çamaşır Yumuşaklığı 50, Çamaşır Miktarı 3 iken Yıkama Devrinin 800 olması gerektiğini gözlemlemekteyiz.

Her defasında girdi değerlerinin kullanıcı tarafından tek tek girilmesi zaman alacağından dolayı kurulan bu model öncelikle Şekil 17'de gösterildiği üzere "Workspace"e aktarılmalıdır.

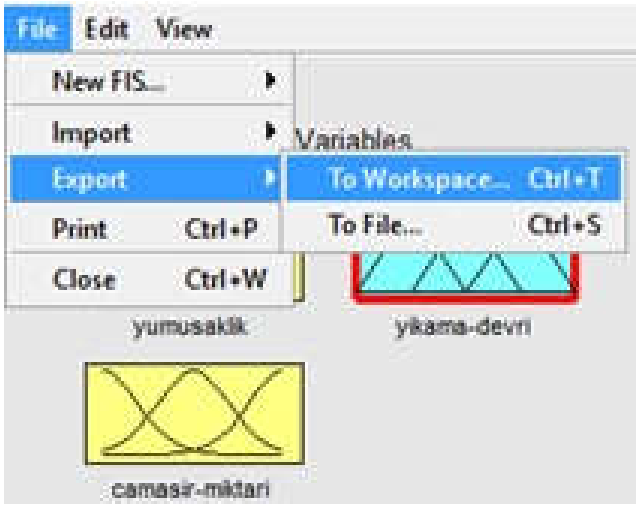
Daha sonra Command Window ekranında aşağıdaki kodlar çalıştırılarak istenilen girdi değerlerine karşılık olarak Yıkama Devri değerleri matris işlemleri yoluyla kolayca elde edilebilmektedir.



Şekil 3.15: Girdi çıktı ilişkisinin 3 boyutlu grafik ile gösterimi



Şekil 3.16: Durulaştırılmış sonuçların elde edilmesi



Şekil 3.17: Modelin Workspace'e aktarılması

```
1        CY=[15;20;35;50;80];
2        CM=[5;3;2;3;4];
3         giris=[CY CM];
4        YD=evalfis(giris,devirhizi)
```

Kodlar yazıldığında Yıkama Devrini temsil eden YD değişkeni (YD= 1048.2 1048.2 880.7 800.2 372.6) olarak hesaplanmış olacaktır.

3.3 Kaynakça

[1] Zadeh, L. A., Fuzzy Sets, Information and Control, Vol. 8, pp. 338-353, 1965.

[2] Dadone, P., Design Optimization of Fuzzy Logic Systems, Doctor of Philosophy in Electrical Engineering, Virginia Polytechnic Institute and State University, 2001.

[3] Mendel, J. M., Fuzzy Logic Systems for Engineering: A Tutorial, Proceedings of the IEEE, 83(3), 1995.

[4] Hızıroğlu, A., Cebeci, H.İ., Taşkın, H., İpek, M., Selvi, İ.H., Kiraz, A., Şişci, M., Codal, K.S., Esnek Hesaplama: İşletme ve Ekonomide Uygulamaları, Çeviri Kitap, Sakarya Üniversitesi Rektörlüğü Basımevi, ISBN: 978-605-4735-80-8, 2017.

[5] Şenol, F., Bulanık Mantık Kontrolcüsü, Gazi Üniversitesi Lisans Tezi, 2000.

[6] Hacımurtazaoğlu, M., Bulanık mantık ile manyetik kilit uygulaması, XIV. Akademik Bilişim Konferansı, Mersin, 2014.

[7] Mamdani, E. H., Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, Proc. IEEE, 121(12), pp. 1585-1588, 1974.

[8] Munakata, T., Jani, Y., Fuzzy Systems: An Overview, Communications of the ACM, 37(3), pp. 69-76, 1994.

[9] Tiryaki, A. E., Kazan, R., Bulaşık Makinesinin Bulanık Mantık ile Modellenmesi, Mühendis ve Makina Dergisi, TMMOB Makina Mühendisleri Odası, Cilt 48, Sayı 565, s. 3-8, 2007.

3.4 Yazar Hakkında

Alper Kiraz, 25.02.1985'te Sakarya'da doğdu. İlk, orta ve lise eğitimini Sakarya'da tamamladı. 2003 yılında Figen Sakallıoğlu Anadolu Lisesi'nden mezun oldu. 2003 yılında lisans öğrenimine başladığı Sakarya Üniversitesi Endüstri Mühendisliği Bölümü'nden 2007 yılında mezun oldu. 2007 yılında Sakarya Üniversitesi Endüstri Mühendisliği Anabilim dalında yüksek lisansa başlayarak, aynı yıl Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nde Araştırma Görevlisi olarak göreve başladı. "Yapay Sinir Ağları ile Sanal Laboratuvar Tasarımı" isimli yüksek lisans tezini tamamlayarak 2010 Ocak ayında doktoraya başladı. Erasmus Öğrenim Hareketliliği kapsamında İrlanda Institute of Technology Tralee Üniversitesi'nde doktora eğitimi aldı. 2015 Ocak ayında "Yapay Zeka Destekli Sanal Laboratuvar Tasarımı: Çekme Deneyi Uygulaması" isimli doktora tezini tamamladı. 2012 yılında "MATLAB: Yapay Zeka ve Mühendislik Uygulamaları" kitabında bölüm yazarlığı ve 2017 yılında "Esnek Hesaplama: İşletme ve Ekonomide Uygulamaları" çeviri kitabı yazarlarındandır. Şuan halen Sakarya Üniversitesi Endüstri Mühendisliği bölümünde Yrd. Doç. Dr. olarak görev yapmaktadır. Detaylı özgemişe http://www.kiraz.sakarya.edu.tr/ adresinden erişilebilir.

4. Makine Öğrenmesi Kütüphaneleri

Açık Kaynaklı Makine Öğrenmesi Kütüphaneleri

Öğr. Gör. Dr. Murat GEZER, Sefa SAYLAN

4.1 Giriş

Günümüzde veri madenciliği alanında kullanılan birçok farklı programlama dili ve analiz aracı mevcuttur. Veri madenciliği alanında kullanılan ücretli araçlar; genelde işletmeler tarafından rağbet görseler de akademik dünyada sınırlı bütçe ve kaynak yetersizliğinden dolayı rağbet görmemektedirler. Bu durum beraberinde geliştirmeyi ve gelişmeyi getirmektedir. Python programlama dili gibi birçok açık kaynaklı programlama dillerinde makine öğrenmesi algoritmalarını sunan farklı açık kaynaklı kütüphaneler mevcuttur. Bu kütüphaneler aracılığıyla farklı alanlarda makine öğrenmesi algoritmaları kullanarak tahmin modelleri, kümeleme, sınıflandırma gibi işlemler insan müdahalesi olmadan yüksek doğrulukla gerçekleştirilebilir. Bu bölümde açık kaynaklı yazılımların ne anlama geldiği ifade edildikten sonra python programlama dilinde sunulan açık kaynaklı kütüphaneler tanıtılacaktır.

4.2 Açık Kaynak Nedir?

"Açık kaynak kodlu yazılımı temel olarak kaynak kodu herkes tarafından incelenebilen, üzerinde değişiklik yapılabildiği ve yeniden dağıtılabilir yazılım olarak tanımlayabiliriz." (GEZER, 2005)

"Açık kaynak kodlu yazılım, kullanıcıya yazılımı değiştirme olanağı sağlaması demektir. Daha açık bir ifade ile kullanıcıya kodun derlenmemiş, ham halinin sunulmasıdır. Bu sayede kullanıcı/geliştirici kodu ihtiyacına uygun şekilde tekrar derleyebilir." (Şen, 2010)

"Açık kaynak kod, ürünün kaynağına rahatça erişebilme imkânı sunan bir uygulama geliştirme yöntemi olarak tanımlanabilir." (Altıparmak, 2011)

Yukarıdaki tanımlardan da açıklandığı üzere açık kaynaklı yazılımlar; kaynak kodları kullanıcılar tarafından incelenebilen ve üreticisinin gerçekleştirmiş olduğu lisanslama türüne bağlı olarak farklı amaçlarla kullanılabilen yazılım türüdür.

Günümüzde açık kaynak denildiğinde aklımıza ilk gelen işletim sistemlerinden birisi olan Linux'ün 341 adet farklı dağıtımı bulunmaktadır (distrowatch.com).

4.3 Açık Kaynaklı Yazılımların Lisanslanma Yöntemleri Nelerdir ?

Kullanılan açık kaynaklı yazılımlarında üreticilerin haklarının koruması için sunulan birçok farklı lisanslama yöntemi mevcuttur. Open Source Initiative (OSI) 1998 yılında kamu yararına Kaliforniya'da kurulan, vergi muafiyetine sahip , kar amacı gütmeyen, açık kaynağın yararlarını savunan, öğreten ve farklı bölgelerdeki açık kaynak toplulukları ile köprüler oluşturmayı hedefleyen küresel bir kuruluştur (About the Open Source Initiative | Open Source Initiative) .

Üretilmiş olan yazılımın açık kaynaklı bir yazılım olarak tanımlanabilmesi için aşağıdaki kriterleri sağlaması gerekmektedir (The Open Source Definition (Annotated) | Open Source Initiative).

1. Ücretsiz yeniden dağıtılabilir olmalıdır.
2. Kaynak kodu erişilebilir olmalıdır.
3. Kaynak kodun lisans hakları korunarak yeni ürünler türetilebilir olmalıdır.
4. Oluşturulan üründe farklı açık kaynaklardan yararlanması durumunda bu kaynakların lisanslarına sağdık kalmalıdır.
5. Kişilere veya gruplara karşı ayrımcılık yapılmamalıdır.
6. Kullanım alanına yönelik bir sınırlamada bulunulmamalıdır.
7. Kaynak kodun lisansı elde edilen kişiler için de aynen geçerli olmalıdır.
8. Lisans yalnızca tek bir ürüne özgü olmamalıdır.
9. Lisans diğer yazılımları kısıtlayamaz.
10. Lisans teknoloji tarafsız olmalıdır.

Lisans için gerekli olan kuralların açıklamaları için lütfen web sayfasını ziyaret ediniz. Açık kaynak dünyasında en popüler kullanılan lisanslama türleri aşağıda sunulmuştur
(Open Source Licenses by Category | Open Source Initiative).

- Apache License 2.0 (Apache-2.0)
- 3-clause BSD license (BSD-3-Clause)
- 2-clause BSD license (BSD-2-Clause)
- GNU General Public License (GPL)
- GNU Lesser General Public License (LGPL)
- MIT license (MIT)
- Mozilla Public License 2.0 (MPL-2.0)
- Common Development and Distribution License version 1.0 (CDDL-1.0)
- Eclipse Public License version 2.0

4.4 Python Nedir?

Python, Guido Van Rossum adlı Hollandalı bir programcı tarafından 1990 yılında geliştirilmeye başlanan bir programlama dilidir. Python ismi İngiliz komedi grubu Monty Python' dan gelmektedir (Özgür, 2010).

Geliştiriciler tarafından açık kaynaklı olarak sunulan birçok kütüphanesi bulunan python programlama dilinde uygulama geliştirmek oldukça kolaydır. C ve benzeri programlama dilleri gibi programın sonlandığını bir (";" gibi) operatör yardımıyla bildirmenize gerek yoktur. Değişken tanımlarken türünü kendisi otomatik olarak girilen değere göre belirleyebilir.

İçerisinde lineer cebir işlemlerini gerçekleştirebileceğimiz " numpy " kütüphanesi mevcuttur. Çok boyutlu dizilerle işlem yapılması mümkün olan bu kütüphane sayesinde birçok lineer cebir işlemini sunulan fonksiyonlarla gerçekleştirebilirsiniz (NumPy).

Mühendisler, matematikçiler ve bilim insanlarının kullanımına sunulan bir başka kütüphane ise "scipy" dır. Fourier transform , interpolasyon, optimizasyon, ve istatistiksel algoritmaları barındıran bir kütüphanedir (SciPy.org).

Yine geliştiriciler tarafından açık kaynaklı olarak sunulan ve bilimsel makalelerde de sıklıkla kullanılan bir kütüphane olan "matplotlib" sayesinde çalışmalarda grafik çizimi için ücreti araçları kullanmak yerine ücretsiz olarak istediğiniz grafikleri çizdirebilir ve bu grafikleri kullanabilirsiniz (Matplotlib: Python plotting - Matplotlib 2.0.2 documentation)

Özetle python dilinde tıp alanından astronomiye kadar birçok farklı alanda kütüphaneler geliştirilmiştir. Günümüzde popüler olarak makine öğrenmesi algoritmaları da yine python programlama dilinde açık kaynaklı olarak hizmete sunulmuştur. Bu bölüm içerisinde makine öğrenmesi kütüphanelerine ve makine öğrenmesi algoritmalarına değinilecektir.

4.5 Makine Öğrenmesi Nedir?

İnsanlar günlük hayatta karşılaşılan problemin çözümü için birbirinden farklı yöntemleri akıl süzgecinden geçirerek bir takım karmaşık ve mantıksal işlemlere tabi tutar ve sonuçlar üretirler. Karşılaşılan bir problemle eğer ilk defa karşılaşılmadıysa çözümü yine ilkine benzeyen bir yöntemle gerçekleşecektir.

Makine öğrenmesi; mantıksal işlemler gerçekleştirebilen makinelerin, gözlem ve ölçüm yöntemleriyle elde edilen verileri tecrübe olarak kabul etmesi ve bu tecrübelerden matematiksel algoritmalar aracılığıyla anlamlı ilişkiler üretmesi süreci olarak tanımlanabilir.

Makine öğrenmesinde veri kümeleri geçmiş tecrübelerimize karşılık gelmektedir, öğrenme algoritmaları sayesinde veriler arasındaki ilişkilerden çözülmesi istenilen probleme uygun bir model oluşturulur ve problemlerin çözümü için modele başvurarak sonuçlar üretilir.

Nasıl ki belirli bir alanda uzmanlaşmış ve bu alanda geçmişte çok fazla tecrübesi bulunan kişinin karşılaştığı problemi kolayca çözüme ulaştırması mümkünse, belirli bir alana özgü daha büyük veri kümelerinden tecrübe edinen öğrenme algoritmalarının da kolayca doğru çözüme ulaşması mümkündür.

Makine öğrenmesi yöntemleri kendi içerisinde ikiye ayrılır. Bunlar :

- Denetimli Öğrenme
- Denetimsiz Öğrenme

yöntemleridir.

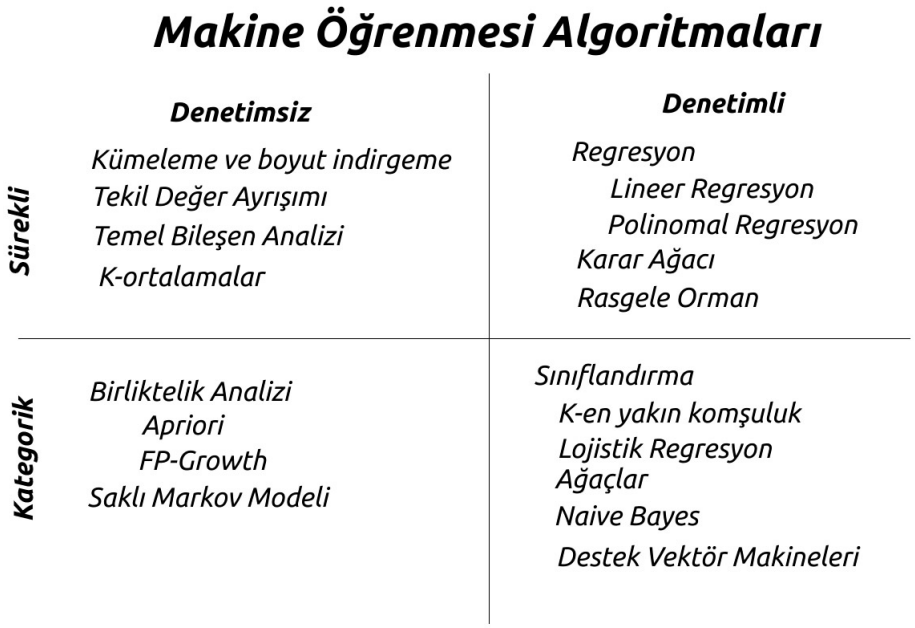
4.5.1 Denetimli Öğrenme Nedir?

"Denetimli öğrenme, önceden gözlemlenmiş ve sonuçları bilinen (etiketlenmiş) verileri kullanarak bu verileri ve sonuçlarını kapsayan bir fonksiyon oluşturmayı amaçlayan makine öğrenimi metodudur" (Nizam, 2014).

"Denetimli öğrenmede hedef, girdi ölçülerinin sayısını temel alarak çıktı ölçüsünün değerini tahmin etmektir" (Koyuncugil, 2009).

4.5.2 Denetimsiz Öğrenme Nedir?

"Denetimsiz öğrenme, etiketlenmemiş verideki gizli yapıyı bulma işlemidir. Yani, veriler arasında var olan ama gözle görülmeyen bağıntının açığa çıkarılması işlemidir" (Nizam, 2014).



Şekil 4.1: Makine öğrenmesi algoritmaları

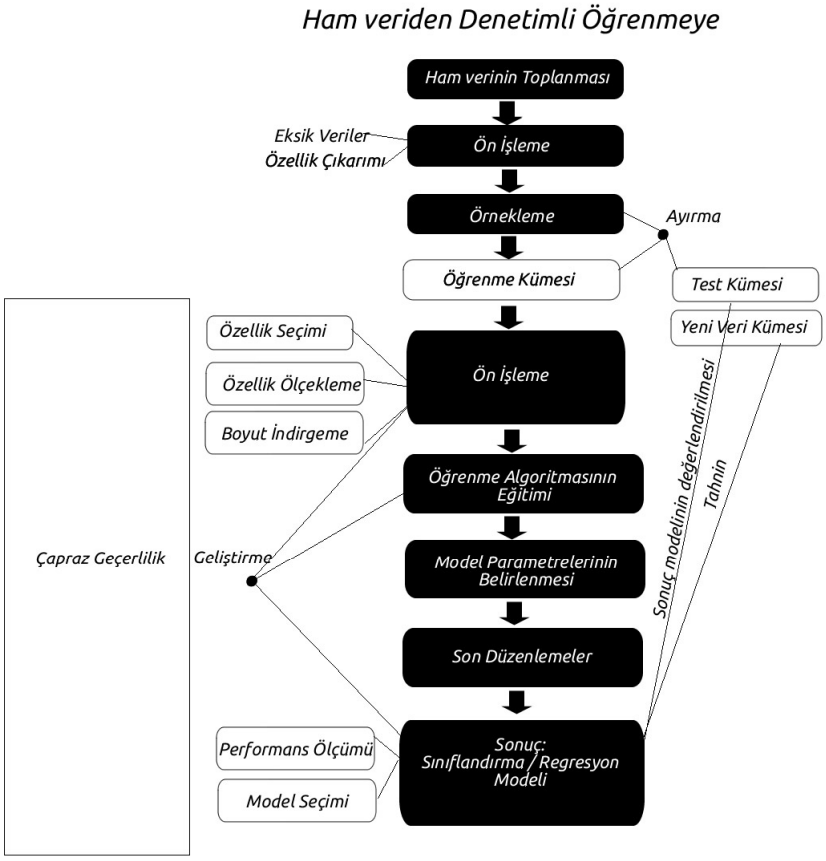
Şekil 4.1 görselinde denetimli ve denetimsiz öğrenme yöntemlerinde veri tipine göre(kategorik, sürekli) kullanılabilecek algoritmalar verilmiştir. Kaynak: ((PyCon 2014 Video) How To Get Started with Machine Learning - Melanie Warrick's PyCon 2014 Talk | Hackbright Academy, 2014) Kaynaktaki görsel Türkçeleştirilmiştir.

Şekil 4.2 Kaynak: (Predictive modeling, supervised machine learning, and pattern classification, 2014). Kaynaktaki görsel Türkçeleştirilmiştir.

Denetimli ve denetimsiz öğrenme konusunu bir örnek üzerinden ele alırsak;

1000 çalışanlı bir firmanın bünyesine 50 yeni çalışan daha alacağını ve kişilerin maaşlarını belirlemek için bir makine öğrenmesi algoritması kullanacağını varsayalım.

Bu problemi çözmek için 1000 adet çalışanın bilgileri kullanılarak öğrenme ve test veri kümeleri oluşturulur. Denetimli öğrenme yöntemi algoritmalarından birisini kullanılarak öğrenme veri kümesinden yaş, mezun olunan okul, iş tecrübesi ve mevcut maaş bilgilerini öğrenme algoritmasına girdi olarak veririz ve algoritma elindeki verilerle anlamlı ilişkiler kurarak bir maaş tahmini gerçekleştirir. Daha sonra bu yöntem test edilerek başarısı hesaplanır ve 50 kişinin maaşı oluşturulan



Şekil 4.2: Denetimli öğrenme ile makine öğrenmesi işleminin akış diyagramı

model üzerinden bulunur.

Eğer biz denetimsiz öğrenme yaklaşımının algoritmalarından birisini kullanırsak girdi parametresi olarak kişilerin özgeçmişlerindeki tüm veriyi algoritmaya veririz ve algoritma bu veriler arasında ilişkiler oluşturmaya çalışır. Örneğin yaş, cinsiyet ve doğduğu yıl , yaş ve iş tecrübesi gibi birçok farklı ilişki oluşturabilir. Bunun sonucunda bir model oluşur ve bu model üzerinden tahmin işlemi gerçekleştirilir. Denetimli öğrenme yaklaşımında hangi veriler arasında ilişki kuracağını belirlerken denetimsiz öğrenme yaklaşımında ilişkilerin seçimini algoritmaya bırakmış oluyoruz.

4.6 Makine Öğrenmesi Kütüphaneleri Nelerdir?

Bu başlık altında klasik öğrenme kütüphanelerini ve derin öğrenme kütüphaneleri açıklanacaktır.

4.6.1 Klasik makine öğrenmesi kütüphaneleri

SciKit-Learn

Makine öğrenmesine ait birçok algoritmayı bizlere sunan python kütüphanesidir. Bu kütüphaneyi kullanarak; k-en yakın komşuluk, destek vektör makineleri, naive bayes, karar ağaçları, rasgele orman, yapay sinir ağları, k-ortalamalar algoritmalarını kullanabilirsiniz.

PyLearn2

Theno kütüphanesinin üzerine geliştirilmiş bir makine öğrenme kütüphanesidir. Bu kütüphaneyi kullanarak k-ortalamalar, temel bileşen analizi, destek vektör makineleri algoritmalarını kullanabilirsiniz. Ayrıca bu kütüphane aracılığıyla işlemlerinizi grafik kartı işlemcisi üzerinde de gerçekleştirebilirsiniz. Kaynak: (Models - Pylearn2 dev documentation)

NuPIC (The Numenta Platform for Intelligent Computing)

Hiyerarşik geçici bellek (HTM) öğrenme algoritmalarını kullanan bir kütüphanedir. Canlı verilerde ve anomali tespitinde kullanılmaktadır. Kaynak: (NuPIC 1.0.3 API Documentation - NuPIC 1.0.3)

NiLearn

Nöro-görüntüleme verilerini kullanarak kolayca istatistiksel yöntemleri uygulayan tahmin, sınıflandırma ve bağlantı analizlerini gerçekleştiren bir kütüphanedir. Daha çok beyin görüntüleri üzerine nöronlar arasındaki bağlantıların analizler ve tahmin modelleri için kullanılmaktadır. Kütüphanede destek vektör makinesi algoritmasını kullanılabilir. Kaynak: (Nilearn: Machine learning for NeuroImaging in Python - Machine learning for NeuroImaging)

PyBrain

Bu kütüphane içerisinde standart ve gelişmiş birçok farklı algoritma mevcuttur. Bunlar; destek vektör makineleri, k ortalamalar, temel bileşen analizi, ve derin öğrenme algoritmalarıdır. Kaynak: (PyBrain)

Pattern

Bir web madenciliği kütüphanesidir. Veri madenciliği araçları içermektedir, internet üzerinden veri toplayan botlar, Html etiketlerini parçalayan fonksiyonlar, doğal dil işleme araçları, kümeleme, sınıflandırma, destek vektör makineleri, network analizi ve analizi görselleştirme araçları mevcuttur. Ayrıca içerisinde 50'den fazla örnek mevcuttur. Kaynak: (Pattern | CLiPS)

Bob

Ücretsiz olarak sunulan sinyal işleme ve makine öğrenmesi aracıdır. Hem python hem de c++ kodları ile tasarlanmış bir kütüphanedir. Makine öğrenmesi algoritmalarından destek vektör makineleri ve çok katmanlı algılayıcıları mevcuttur. Kaynak: (Bob - A framework for signal processing and machine learning)

MILK (Machine Learning Toolkit)

Python'un makine öğrenmesi aracı olarak daha çok denetimli öğrenme algoritmalarına odaklanmış olduğu bilinen bir kütüphanedir. K ortalamalar, destek vektör makineleri, rasgele orman, temel bileşen analizi, karar ağaçları, k-en yakın komşuluk algoritmalarını bu kütüphane içerisinde bulabilirsiniz. Kaynak: (Milk: Machine Learning Toolkit for Python, 2017).

AIMA (Artificial Intelligence: A Modern Approach)

Stuart Russell ve Peter Norvig'ın "Artificial Intelligence: A Modern Approach" adlı eserindeki sözde kodları verilen algoritmaları bizlere sunan bir kütüphanedir. Doğal dil işleme alanında da çeşitli algoritmaları içinde barındırmaktadır. Kaynak : (GitHub - aimacode/aima-python: Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach")

Simple AI

AIMA python kütüphanesi içerisinde bulunan algoritmalar baz alınarak hazırlanmıştır. Artificial Intelligence: A Modern Approach adlı kitabın içerisindeki algoritmaları sunmuştur. AIMA'dan farkını algoritmaların kullanımında daha "pythonik" bir yaklaşımda bulunulduğundan ve istikrarlı, modern sürdürülebilir bir versiyon oluşturmaya önem verdiklerinden bahsedilmektedir. Kaynak: (GitHub - simpleai-team/simpleai: simple artificial intelligence utilities)

EasyAI

Python dilinde sunulan bir yapay zeka kütüphanesi olan easyAi genellikle oyunların yapay zekalarında kullanılmaktadır. Örneğin; Tic Tac Toe , Connect 4, Reversi gibi oyunlarda kullanılmaktadır Kaynak: (games) .

PyML (Machine Learning in Python)

İnteraktif nesne yönelimli bir çatı altında Python'da makine öğrenmesi için geliştirilmiş bir kütüphanedir. Destek vektör makinelerine ve onun diğer çekirdek fonksiyonları ile gerçekleştirilecek makine öğrenmesi algoritmalarına odaklanmış bir kütüphanedir. Linux ve Mac OS X işletim sistemlerinde çalışmaktadır (PyML - machine learning in Python — PyML v0.7.3 documentation).

MLPy (Machine Learning in Python)

Denetimli ve denetimsiz öğrenme alanında birçok makine öğrenmesi algoritmasını bizlere sunan bir kütüphanedir. Regresyon, sınıflandırma (destek vektör makineleri, k-en yakın komşuluk), k-ortalamalar gibi birçok farklı algoritmayı kullanıma sunmaktadır. Kaynak: (mlpy - Machine Learning Python)

Shogun

Soeren Sonnenburg ve Gunnar Raetsch tarafından 1999 yılında başlatılan bir projedir. Büyük ölçekli kernel fonksiyonlarına ve biyoinformatik alanında ki çalışmalar için geliştirilmiştir. Bir makine öğrenmesi kütüphanesi olarak birçok atıf alan bu kütüphane 2017 yılında numfocus kütüphaneleri (kar amacı gütmeyen açık kaynaklı bilimsel hesaplama kütüphaneleri) arasına katılmıştır. K-en

yakın komşuluk, yapay sinir ağları, karar ağaçları, k-ortalamalar, naive bayes, gibi birçok farklı makine öğrenmesini algoritmalarını kullanabileceğiniz bir kütüphanedir. (Shogun Machine Learning - Home) (About | NumFOCUS, 2017)

LibSVM

Destek vektör makineleri algoritmalarını bizlere sunan python kütüphanesidir. Farklı kernel fonksiyonlarını kullanabilirsiniz. Kaynak: (LIBSVM – A Library for Support Vector Machines, 2017).

NLTK (Natural Language Toolkit)

Doğal dil işleme kütüphanelerinde en önde gelenlerden birisi olan nltk içerisinde 50 farklı derlem ve sözlük bulundurmaktadır. Sınıflandırma, etiketleme, parçalama, semantik çıkarımlarda bulunma gibi birçok farklı işlemi bu kütüphaneyle gerçekleştirebilirsiniz. Kaynak: (Natural Language Toolkit - NLTK 3.2.5 documentation)

Gensim (Generate Similar)

Czech Digital Mathematics Library için farklı python kodlarını barındırarak 2008 yılında yayınlanmaya başlayan gensim kütüphanesi belirli bir makale için o makaleye benzer makalelerin listesini oluşturarak sunmayı amaçlamaktadır. Önceleri farklı yöntemler denenmişse de istenilen sonuçlara ulaşılamamış ve gensim bu sorunlara çözüm için üretilmiştir. İçerisinde semantik analiz ve benzerlik fonksiyonları mevcuttur. Kaynak: (gensim: Tutorials)

PyAnn

Python dili için geliştirilmiş basit ve temel yapay sinir ağı kütüphanesidir. Süreci olabildiğince basitleştirmeyi hedeflemektedir. Kaynak : (GitHub - racaljk/pyann: Artificial Neural Network Library for Python)

FFNet

Python için ileri beslemeli yapay sinir ağı kütüphanesidir. Farklı optimizasyon yöntemlerine izin verir. Kaynak: (Overview - ffnet 0.8.3 documentation)

NeuroLab

Yapay sinir ağlarını çalıştırabileceğimiz python kütüphanelerinden birisidir. Esnek ağ konfigürasyonun yapısına sahiptir. Kütüphanede numpy ve scipy kütüphanelerinin fonksiyonları kullanılmaktadır. Kaynak: (Welcome to NeuroLab's documentation! — NeuroLab 0.3.5 documentation)

Bu aşamaya kadar tanıtılan kütüphaneler klasik yöntemle gerçekleştirilen makine öğrenmesi algoritmalarını bizlere sunan kütüphanelerdir. Bu aşamadan sonra tanıtılacak olan kütüphaneler, makine öğrenmesi alanında yeni bir yaklaşım olan ve temelleri yapay sinir ağlarına dayanan derin öğrenme algoritmalarını bizlere sunan kütüphanelerdir. Bu aşamaya geçmeden önce derin öğrenme hakkında kısaca birkaç bilgi vermek gerekirse; klasik yaklaşımlardan en büyük farkı özellik çıkarımı işlemini kendi başına gerçekleştiriyor olmasıdır, bu özelliği sayesinde ön işleme aşamasında zaman kaybedilmemektedir. Büyük veri kümesi kullanılarak gerçekleştirilen derin öğrenme modelleri çok fazla işlemci gücüne ihtiyaç duymaktadır bunun için grafik kartlarının işlemcilerinden yararlanılmaktadır.

4.6.2 Derin öğrenme kütüphaneleri

Hebel

Python'da yapay sinir ağlarını hızlıca derlemek için PyCuda ile grafik işlemcisini kullanılan derin öğrenme kütüphanesidir. Farklı aktivasyon fonksiyonları, yapay sinir ağı modellerini kullanabilirsiniz ayrıca eğitim yöntemi olarak da farklı momentumları kullanmak mümkündür. (Bretschneider, 2014)

Theano

Numpy'a benzer çok boyutlu dizilerle işlem yapmamıza olanak sağlayan kütüphanedir, matematiksel ifadeler ve işlemleri barındırır. Tüm mimarilerde etkili olarak çalışır. Montreal Üniversitesinin makine öğrenme grubu tarafından tasarlanmıştır. Dikkat çeken bir başka özellik ise yükün yoğunluğuna göre yüksek hesaplamalarda CPU veya GPU ile veri işlemeyi optimize eder (Welcome - Theano 0.9.0 documentation) .

TensorFlow

Derin öğrenme kütüphanelerinden birisi olan ve Google geliştiricileri tarafından geliştirilen bu kütüphane Google tarafından görsel üzerinden nesne belirleme ve ses tanıma problemlerinde kullanılmaktadır. Büyük veri setlerinden hızlıca öğrenme işlemini gerçekleştirdiği söylenmektedir. Kaynak: (TensorFlow)

Keras

Yüksek seviyeli bir arayüz sunan ve derin öğrenme algoritmalarından birisi olan Keras'ın tasarımda minimalist yaklaşımla kompakt sistemlerin oluşturulması yoluyla hızlı ve kolay deneme yapılması amaçlandı. Katmak yapısına dayalı olan mimarisi ile giriş katmanında veri tensör olarak verilir ve öğrenme işlemi başlatılır (Keras Documentation).

Cafee

Berkley yapay zeka araştırma geliştirilen açık kaynaklı derin öğrenme kütüphanesidir, geliştirilebilir kod yapısına sahip bir kütüphane ve hızlı olarak çalışmaktadır. Kaynak: (Caffe | Deep Learning Framework)

4.7 Yardımcı Kütüphaneler

Bu bölümde makine öğrenmesi algoritmaları için kullanıma açık olan veri kümelerini bizlere sunan kütüphaneler ve ön işleme aşamasında kullanılan bazı kütüphaneler tanıtılmıştır.

4.7.1 Fuel

Makine öğrenmesi alanında algoritmalarımızı eğitmek için bizlere veriler sunan açık kaynaklı bir kütüphanedir, içerisinde; MNIST(görsel veri) , Google'ın bir milyar kelimesi (metin verisi), CIFAR 10 (görsel veri) veri kümelerini içermektedir. (GitHub - mila-udem/fuel: A data pipeline framework for machine learning)

4.7.2 Skdata

Makine öğrenmesi alanında görüntü işleme ve doğal dil işleme alanında kabul görülen problemlerin veri kümelerini içeren kütüphanedir. (GitHub - jaberg/skdata: Data sets for machine learning in Python)

4.7.3 EIPY

Bilgi çıkarımına odaklanmış olan bir python kütüphanesidir. Örneğin metin içerisinden bir ilişki çıkarmak istiyorsunuz bunun için bu kütüphaneyi inceleyin. (GitHub - machinalis/iepy: Information Extraction in Python)

Bu içeriğin hazırlanmasında ayrıca (Top 20 Python Machine Learning Open Source Projects) ve (Python'un En iyi 15 Veri Bilimi Aracını Karşılaştırdık - SilikonHaber.com) Kaynaklarından yararlanılmıştır.

4.8 Sonuç

Günümüzde Python dilinde geliştirilen birçok farklı açık kaynaklı makine öğrenmesi kütüphanesi tanıtılmıştır. Bu kütüphaneler sayesinde geliştiriciler algoritmaları kullanarak kendi uygulamalarını geliştirebilir ve problemlerine farklı çözümler bulabilirler.

4.9 Kaynakça

Simple artificial intelligence utilitiesi, https://github.com/simpleai-team/simpleai adresinden 10.1.2017 tarihinde alındı.

Shogun Machine Learning, www.shogun-toolbox.org adresinden 10.1.2017 tarihinde alındı.

hackbrightacademy.com: https://hackbrightacademy.com/blog/pycon-2014-melanie-warrick-machine-learning-talk/ adresinden alındı.

PyCon 2014 Video, How To Get Started with Machine Learning - Melanie Warrick's PyCon 2014 Talk | Hackbright Academy. www.hackbrightacademy.com/blog/pycon-2014-melanie-warrick-machine-learning-talk/ adresinden 9.29.2017 tarihinde alındı

Özgür, F. (2010). Python.

About | NumFOCUS. (2017.10.1). numfocus.org: https://www.numfocus.org/about/ adresinden alındı

About the Open Source Initiative | Open Source Initiative. https://opensource.org/about adresinden alındı

Altıparmak, M. K. (2011). E-öğrenme ve uzaktan eğitimde açık kaynak kod lu öğrenme yönetim sistemleri. Akademik Bilişim Kongresi.

Bob - A framework for signal processing and machine learning. www.idiap.ch/software/bob/ adresinden 10.1.2017 tarihinde alınmıştır.

Bretschneider, H. (2014, 5). Hebel - GPU-Accelerated Deep Learning Library in Python.

Caffe | Deep Learning Framework. http://caffe.berkeleyvision.org/ adresinden 10.1.2017 tarihinde alındı.

distrowatch.com. DistroWatch.com: Put the fun back into computing. Use Linux, BSD. distrowatch.com: http://distrowatch.com/dwres.php?resource=origin adresinden alındı

Games, Python artificial intelligence framework for games, www.github.com/Zulko/easyAI adresinden10.1.2017 tarihinde alındı.

Gensim: Tutorials. https://radimrehurek.com/gensim/tutorial.html adresinden 10.1.2017 tarihinde alındı.

GEZER, M. (2005). Açık Kaynak Yazılımlarında Astronomi. 3 (2), 119-125.

GitHub - Python implementation of algorithms from Russell And Norvig's "Artificial Intelligence - A Modern Approach". www.github.com/aimacode/aima-python adresinden alındı.

GitHub - jaberg/skdata: Data sets for machine learning in Python. www.github.com/jaberg/skdata, adresinden 10.1.2017 tarihinde alındı.

GitHub - machinalis/iepy: Information Extraction in Python, https://github.com/machinalis/iepy adresinden 10.1.2017 tarihinde alındı.

GitHub - mila-udem/fuel: A data pipeline framework for machine learning.10 1, 2017 tarihinde github.com: https://github.com/mila-udem/fuel adresinden alındı

GitHub - racaljk/pyann: Artificial Neural Network Library for Python 10 1, 2017 tarihinde https://github.com/racaljk/pyann: https://github.com/racaljk/pyann adresinden alındı

Keras Documentation.10 1, 2017 tarihinde keras.io: https://keras.io adresinden alındı

Koyuncugil, A. S. (2009, 5). Veri Madenciliği: Tıp ve Sağlık Hizmetlerinde Kullanımı ve Uygulamaları. INTERNATIONAL JOURNAL OF INFORMATICS TECHNOLOGIES , 2 (2).

LIBSVM – A Library for Support Vector Machines. www.csie.ntu.edu.tw/ cjlin/libsvm/ adresinden 2017.10.1 tarihinde alındı.

Matplotlib: Python plotting - Matplotlib 2.0.2 documentation. https://matplotlib.org/ adresinden alındı

Milk: Machine Learning Toolkit for Python. (2017.10.1). luispedro.org/software/milk/ adresinden alındı.

mlpy - Machine Learning Python. 10.1.2017 tarihinde http://mlpy.sourceforge.net/ adresinden alındı

Models - Pylearn2 dev documentation. http://bit.ly/2ytjyUt adresinden 9.30.2017 tarihinde alındı.

Natural Language Toolkit - NLTK 3.2.5 documentation. http://www.nltk.org/ adresinden 10.01.2017 tarihinde alındı.

Nilearn: Machine learning for NeuroImaging in Python - Machine learning for NeuroImaging.09 30, 2017 tarihinde nilearn.github.io: http://nilearn.github.io/ adresinden alındı

Nizam, H. &. (2014). Sosyal medyada makine öğrenmesi ile duygu analizinde dengeli ve dengesiz veri setlerinin performanslarının karşılaştırılması. XIX. Türkiye'de İnternet Konferansı.

NumPy.numpy.org: http://www.numpy.org/ adresinden alındı

NuPIC 1.0.3 API Documentation - NuPIC 1.0.3. www.nupic.docs.numenta.org/stable/index.html adresinden 09.30.2017 tarihinde alındı.

Open Source Licenses by Category | www.opensource.org/licenses/category adresinden alındı

Overview - ffnet 0.8.3 documentation. www.ffnet.sourceforge.net/overview.html adresinden 10.1.2017 tarihinde alındı.

Pattern | CLiPS. www.clips.uantwerpen.be/pages/pattern adresinden 10.1.2017 tarihinde alındı.

Predictive modeling, supervised machine learning, and pattern classification. (2014, 08 25). 09 29, 2017 tarihinde sebastianraschka.com: http://bit.ly/1B5GV0Z adresinden alındı

PyBrain.10 01, 2017 tarihinde pybrain.org: http://www.pybrain.org/pages/features adresinden alındı

PyML - machine learning in Python — PyML v0.7.3 documentation.pyml.sourceforge.net/: http://pyml.sourceforge.net/ adresinden alındı

Python'un En iyi 15 Veri Bilimi Aracını Karşılaştırdık - SilikonHaber.com.10 1, 2017 tarihinde silikonhaber.com: https://www.silikonhaber.com/python-veri-madenciligi-146/ adresinden alındı

scikit-learn: machine learning in Python - scikit-learn 0.19.0 documentation.9 30, 2017 tarihinde scikit-learn.org: http://scikit-learn.org/stable/ adresinden alındı SciPy.org.scipy.org: www.scipy.org adresinden alındı

Şen, B. A. (2010). Düşük maliyetli web tabanlı uzaktan eğitim sistemi uygulaması. (s. 10-12). Akademik Bilişim.

TensorFlow.10 1, 2017 tarihinde tensorflow.org: https://www.tensorflow.org adresinden alındı.

The Open Source Definition (Annotated) https://opensource.org/osd-annotated adresinden alındı.

Top 20 Python Machine Learning Open Source Projects.http://www.kdnuggets.com/2015/06/top-20-python-machine-learning-open-source-projects.html adresinden 10.01.2017 tarihinde alındı.

Welcome - Theano 0.9.0 documentation. www.deeplearning.net/software/theano/ adresinden 10.01.2017 tarihinde alındı.

Welcome to NeuroLab's documentation! — NeuroLab 0.3.5 documentation. 10.1.2017 tarihinde pythonhosted.org: https://pythonhosted.org/neurolab/ adresinden alındı.

4.10 Yazarlar Hakkında

Almanya doğumlu olan Murat Gezer ilk ve orta eğitiminin büyük kısmını Almanya da Lise eğitimini İstanbul'da tamamladı. Lisans eğitimini Ege Üniversitesi Fen Fakültesi Astronomi ve Uzay Bilimleri bölümünde 1997 yılında tamamlamıştır. Bir süre özel sektörde yazılım ve Ar-Ge departmanlarında çalıştıktan sonra 2007 yılında İstanbul Üniversitesi Enformatik Bölümünde Yüksek Lisansını tamamlamış ardından 2014 yılında İstanbul Üniversitesi Elektrik-Elektronik Mühendisliği bölümünden "Modern haberleşme sistemlerinde görüntü kodlaması ve sıkıştırmasında özgün matematik yöntemler" adlı tezi savunarak DR. unvanını almıştır. 2011 yılından itibaren İstanbul Üniversitesi Enformatik bölümünde Öğretim Görevlisi olarak çalışmaktadır. Çeşitli kuruluşlar tarafından düzenlenen etkinliklerde Linux, Görüntü İşleme ve Makine Öğrenmesi konusunda seminerler ve eğitimler vermektedir. Detaylı akademik çalışmalara http://bit.ly/2ytzTZa adresinden erişilebilir.

Sefa SAYLAN (İstanbul , 1992), lise eğitimini Halis Kutmangil Çok Programlı Anadolu Meslek Lisesi Bilişim Teknolojileri Web tasarımı ve Programlama Bölümü'nde tamamladıktan sonra (2011), İstanbul Üniversitesi Fen Fakültesi Astronomi ve Uzay Bilimleri Bölümü'nü bitirdi (2015). Eğitimine Marmara Üniversitesi İşletme Fakültesi Sayısal Yöntemler Tezli Yüksek Lisans programında tez aşamasında devam etmektedir.

5. Python ile Görüntü İşleme

print("Hello, world!")

Python ile Görüntü İşlemede Örnek Bir Uygulama

Öğr. Gör. Dr. Murat GEZER

5.1 Giriş

Günümüz büyük bir veri tufanının tam ortasındadır. 2012 rakamlarına göre dijital evrende 2.7 Zetabyte veri bulunmaktaydı ve her geçen gün 2.5 exabyte boyutunda veri bu evrene eklenmektedir [1]. Verinin bu denli büyük boyutta olması nedeniyle anlamlandırılması yani bilgi haline dönüştürülmesi çok önemlidir Bu konuda her geçen gün yeni çalışmalar yapılmaktadır. Bu çalışmalar için makine öğrenmesi teknikleri kullanılmaktadır. Görüntü işleme özellikle dijital teknolojilerin çok hızlı gelişim sağlaması nedeniyle hayatımızın birçok alanında yerini almaktadır. Böylelikle dijital evren için en önemli verinin bilgiye dönüştürme kaynaklarından biri haline gelmiştir. Günümüzde veri bilimciler görüntü sınıflandırma, video analizi gibi çalışmaları için makineyle öğrenmesi için GPU'ları (Grafik İşlemci Ünitesi) kullanmaktadır. Bu yöntemler ile çok büyük miktarlarda eğitim verisini kullanarak çok iyi sonuçlar üretmektedir. Bu çalışmada Raspberry Pi gibi düşük güç tüketimine sahip mini bilgisayarlar kullanılabilecek olan temel yöntemler kullanılmıştır. Bu yöntemler görüntülerin benzerlerinin bulunması için benzerlik ve uzaklık ölçülerinin kullanılması şeklindedir. Bu çalışmada tüm kodlama Python 3.6 kullanarak Anaconda dağıtımı üzerinde gerçekleştirilmiştir.

5.1.1 Temel Tanımlar

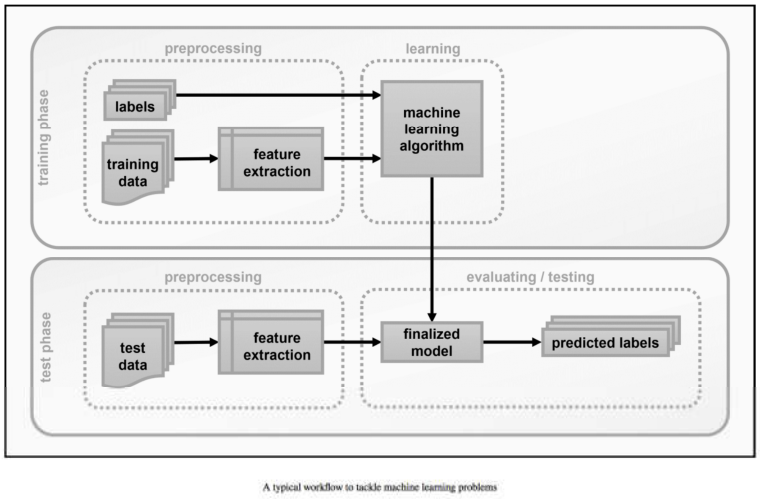
Görüntüler üzerinde gerçekleştirmeye başlamadan önce Makine Öğrenmesi, Görüntü İşleme ve Bilgisayarlı Görme tanımları üzerinde durmakta yarar vardır.

Makine Öğrenmesi:

Makine öğrenmesi bilgisayar bilimlerinin kuramsal alt dalıdır. Matematiksel ve istatistiksel yöntemler kullanarak mevcut verilerden çıkarımlar yapan, bu çıkarımlarla bilinmeyene dair tahminlerde bulunan yöntemler bütünü olarak adlandırabiliriz. Makine Öğrenmesinin öncülerinden olan Arthur

Samuel tanımı "bilgisayarın açık bir şekilde programlanmadan bir işi öğrenme yeteneği olarak" şeklinde vermiştir [2]. Üniversitelerde yaygın bir şekilde ders kitabı olarak kullanılan Introduction to Machine Learning adlı kitabında Alpaydın Makine öğrenmesinin temelini gözlenmiş bir örneklem kümesinden çıkarıp yapmak ve bu çıkarımlara uygun bir model oluşturmak için istatistik kullanılması olarak belirtmiştir [3]. Tom M. Mitchell tarafından verilen ve literatürde genellikle kullanılan tanım: "Bazı amaçları (T) gerçekleştirmek için P performansı ile çalışan bir program, deneyimleri sonucunda (E), P ile ölçülen performansını arttırarak amaçları gerçekleştiriyorsa öğrenebildiği söylenebilir." şeklindedir [4]. Makine öğrenmesi terimi, verilen tanımlardan anlaşılacağı gibi çok genel bir şekilde kullanılmaktadır. Ve genellikle büyük veri kümeleri içeresinden desen çıkarmak ya da mevcut verileri analiz ederek daha önce öğretilenleri temel alan yeni verilere ilişkin tahminler yapma kabiliyetine atıfta bulunmaktadır. Makine öğrenmesi ile istatistik bilimi, veri madenciliği, iş zekası, örüntü tanıma, yapay zekalı sistemler, doğal dil işleme, bilgisayarlı görme, biyoinformatik, robotik, görüntü işleme gibi birçok farklı alan arasında bir ilişki vardır Makine öğrenmesi algoritmaları genel olarak Danışmanlı Öğrenme (Supervised Learning), Danışmansız Öğrenme (Unsupervised Learning) , Yarı-Danışmanlı Öğrenme (Semi-Supervised Learning) ve Pekiştirmeli (Takviyeli) Öğrenme (Reinforcement Learning) olmak üzere dörde ayrılmaktadır. Bu çalışmada Danışmanlı Öğrenme ve Danışmanız Öğrenme algoritmaları ile işlem yapacağız. Danışmanlı Öğrenme algoritmaları sınıf verilerinin tanımlı olduğu durumlarda kullanılmaktadır. Danışmansız Öğrenme algoritmaları ise sınıf verilerinin tanımlı olmadığı durumlarda kullanılmaktadır.

Literatür incelendiğinde makine öğrenmesi süreçlerinde farklı yaklaşımlar bulunduğu görülmemektedir. Ancak en yaygın kullanılan makine öğrenmesi süreci Şekil 5.1 de görülmektedir.



Şekil 5.1: Örnek bir makine öğrenmesi akış şeması [5]

Burada görüldüğü üzere süreç iki aşamadan oluşmaktadır. Eğitim aşaması (training phase) ve test aşaması (test phase) şeklindedir. Makine öğrenmesindek bazı kavramlar şu şekildedir.

Gözlemler (Observations): öğrenmek ya da değerlendirmek için kullanılan her bir veri parçası. Örn: her bir e-posta bir gözlemdir.

Özellikler (Features): Bir gözlemi temsil eden (genelde sayısal) verilerdir. Örn: e-posta'nın uzunluğu, tarihi, bazı kelimelerin varlığı.

Etiketler (Labels): Gözlemlere atfedilen kategoriler. Örn: spam, spam-değil.

Eğitim Verisi (Training Data): Algoritmanın öğrenmesi için sunulan gözlemler dizisi. Algoritma bu veriye bakarak çıkarımlarda bulunur, kafasında model kurar. Örn: çok sayıda spam/spam-değil diye etiketlenmiş e-posta gözlemi.

Test Verisi (Test Data): Algoritmanın kafasında şekillendirdiği modelin ne kadar gerçeğe yakın olduğunu test etmek için kullanılan veri seti. Eğitim esnasında saklanır, eğitim bittikten sonra etiketsiz olarak algoritmaya verilerek algoritmanın (vermediğimiz etiketler hakkında) tahminlerde bulunması beklenir. Örn: spam olup olmadığı bilinen (ama gizlenen), eğitim verisindekilerden farklı çok sayıda e-posta gözlemi.

Görüntü İşleme:

Görüntü İşleme kavramına geçmeden önce sayısal görüntüyü tanımlamak gerekir. Sayısal görüntü genel olarak Görüntülerin sayısal ortam için uygun hale dönüştürülmüş şekilleri olarak tanımlanabilir. Görüntü, iki boyutlu m, n uzay koordinatlarında bir olarak tanımlanan ışık yoğunluk fonksiyonudur [6]. Burada x, y değerleri ve fonksiyon genlikleri sonlu ve tamsayı ise bu görüntü, sayısal görüntü olarak adlandırılır. Böylelikle görüntü işleme, görüntüyü sayısal hale getirmek ve bazı işlemleri gerçekleştirmek için geliştirilmiş, spesifik görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için kullanılan yöntemler olarak söyleyebiliriz. Görüntü işleme ile dijital görüntüler üzerinde çeşitli işlemler yapılarak yeni görüntülerin elde edilmesini amaçlanmaktadır. Bunun için ölçülmüş ya da kaydedilmiş olan dijital görüntü verilerine, bilgisayar da bulunan yazılımlar veya programla dilleri ile amaca uygun şekilde kullanılan matematiksel algoritmalar uygulanır. Genellikle Görüntü İşleme sistemi, önceden belirlenmiş sinyal işleme(Signal Processing)yöntemlerini uygularken görüntüleri iki boyutlu sinyaller olarak ele almaktadır.

Bilgisayar/Makine Görmesi:

Bilgisayar görmesi ya da Makine Görmesi, sayısal görüntülerden veya videolardan üst düzey bir anlam elde etmek için algoritmaların nasıl oluşturulabileceğini ele alan disiplinler arası bir alandır. Genel olarak insan görsel sisteminin yapabileceği görevleri otomatikleştirmeyi amaçlamaktadır. Gördüğünü anlayabilen akıllı bilgisayar sistemlerinin bilgisayar görmesinin ana amacıdır. Üzerinde çalışılan konular arasında, kamera görüntülerinden yüz tanıma, plaka tanıma, görüntüden 3-Boyutlu yüzey geometrisinin bulunması, ayrıt saptama bulunmaktadır.

Şekil 5.2: Görüntü işlemeden bilgisayarlı görmeye işlem zorluk seviyesi [7]

5.2 Python ile Görüntü İşleme

Python programlama dili 1980'lerin sonlarına doğru Guido van Rossum tarafından 1989 Aralık ayında geliştirildi. Adını bir yılandan değil Guido van Rossum'un çok sevdiği, Monty Python adlı altı kişilik bir İngiliz komedi grubunun Monty Python's Flying Circus adlı gösterisinden almıştır.

Günümüzde Python Yazılım Vakfı çevresinde toplanan Python topluluğu tarafından geliştirilmektedir.İlk kararlı sürümü olan Python 1.0 Ocak 1994 yayınlanmıştır. Son kararlı sürüm Eylül 2017 itibari ile 2.x serisinde Python 2.7.14 ve 3.x serisinde Python 3.6.2'dir. 3.x sürümü 3 Aralık 2008 yayınlanmaya başlamıştır; dikkat edilmesi gerek önemli bir nokta 3.x serisi 2.x serisi arasında tam bir geriye doğru uyumluluğun olmamasıdır [8]. Çalışma kapsamında kullandığımız Python 3.5 için her ne kadar doğrudan python.org adresine giderek en son python sürümünü indirmek mümkündür. Bu çalışmada kullanmak üzere https://www.anaconda.com/ adresinde bulunan Anaconda dağıtmını tercih edilmiştir. Anaconda piyasada bulunan bilimsel hesaplama için kullanılan ticari yazılımlara benzeyen açık kaynak kodlu bir platformdur. Python ile görüntü işleme ve makine öğrenmesi için bir çok modül bulunmaktadır. çalışmamızda OpenCV,sklearn, matplotlib ,pandas ve Numpy modülleri kullanılmıştır.

5.3 Benzerlik Algoritmaları ile Rakam Tanıma

5.3.1 Veri Kümesi

UCI Machine Learning Repository adresinde halka açık olan kalem tabanlı el yazısı karakterlerinin tanınması veri kümesi kullanılacaktır. Boğaziçi Üniversitesinden Ethem Alpaydın ve Fevzi Alimoğlu tarafından Wacom PL-100V tablet tarafından toplanmış olan bu veri kümesi 44 farklı kişinin yazmış olduğu 250 farklı rakamdan oluşmaktadır [9]. Toplam 5620 örnekten oluşmaktadır. Veri kümemiz sklearn kütüphanesinde hazır olarak gelmektedir.Veri kümesini yüklemek için öncelikle sklearn kütüphanesini içerisinde bulunan datasets nesnesini çağırdık. Bu işlem için **from sklearn import datasets** komutu kullanılır. datasets nesnesi içerisinde bulunan **load_digits()** metodu ile el yazısı veri kümesi otomatik olarak çevrimiçi olarak indirilmiştir.

5.3.2 Benzerlik ve uzaklık Ölçüleri

Makine öğrenmesinde kullanılan algoritmaların büyük kısmında nesnelerin birbirlerine olan benzerlikleri yada uzaklıkları özellikleri öznitelikleri değerleri vasıtası ile bulunmaktadır. Bu ölçütler problemin yapısı ve verinin türününe göre çeşitlidir .Tablo 5.1 da görüleceği gibi çeşitli Uzaklık ölçüleri bulunmaktadır ayrıca çeşitli benzerlik ölçüleride bulunmaktadır. Bu konuda detaylı bilgi istenirse türkçe kaynak olarak Haldun Akpınar tarafından yazılmış olan Data adlı kitabı gösterilebilir [10].

Aralık Ölçek	**Frekans**	**İkil**
Euclid	chi-square	Euclid
Kareli Euclid	Phi-square	Kareli Euclid
Minkowski		Büyüklük Farkı
Chebyskev		Örüntü Farkı
Manhattan		Varyans
Mahalobonis		Biçim
		Lance & Williams

Tablo 5.1: Uzaklık Ölçüleri [10]

Çalışmada Manhattan uzaklık metriği , Eucklid uzaklık metriği ve Kosinüs benzerliği ölçüleri kullanılmıştır. Vektörün arasındaki açı farkı kullanıldığında Kosinüs Benzerlik öçütü kullanılmış

Aralık Ölçek	**İkil**
Pearson Korelasyonu	Russell
Kosinüs benzerliği	Jackard
	Zar

Tablo 5.2: En Bilindik Benzerlik Ölçüleri [10]

olmaktadır (denklem 5.1).

$$cosinesimilarity = cos\theta = \frac{\vec{a} \cdot \vec{b}}{\| \vec{a} \| \| \vec{b} \|} \tag{5.1}$$

Kosinüş benzerlik ölçütü aradaki açıya bağlı olarak -1 ile 1 sonuçunu döndürmektedir. Sonuç 1'e yaklaştıkça görüntü benzerliği artmaktadır. Manhattan uzaklık ölçüsünde , gözlemler arasındaki mutlak uzaklıkların toplamı alınarak hesap yapılmaktadır [11].

$$manhattandistance = d(i,j) = \sum_{k=1}^{p} (|x_{ik} - x_{jk}|) i, j = 1,2,...,n; k = 1,2,...,p \tag{5.2}$$

Öklit uzaklığı ise iki nokta arasındaki doğrusal uzaklık olarak tanımlanabilir.

$$euclliddistance = d(i,j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2)} i, j = 1,2,...,n; k = 1,2,...,p \tag{5.3}$$

Uzaklık ve benzerlik ölçüleri için fonksiyonlar **sklearn.metrics.pairwise** nesnesinde bulunur ve bu konuda detaylı bilgi `http://scikit-learn.org` adresinde bulunmaktadır.

5.3.3 Uygulama

Veri kümemisin yüklenmesi *Spyder* editöründe

```
from sklearn import datasets
digits = datasets.load\_digits()
```

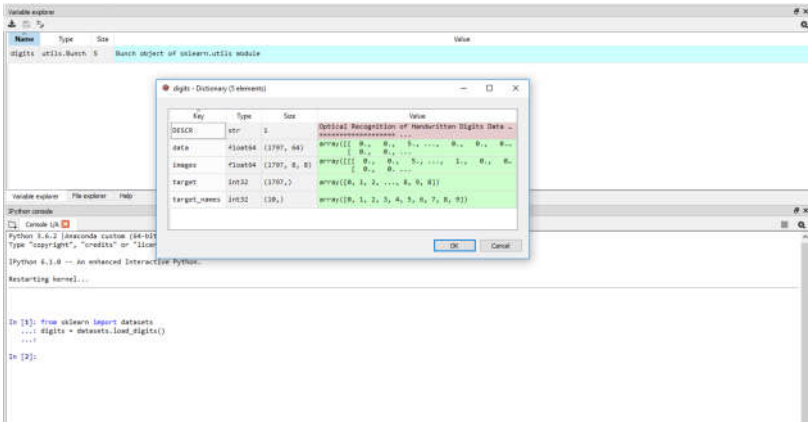
şeklinde girilerek sağlanmıştır. Şekil 5.3 'de veri kümesi yüklendikten spyder içersinde bulunan değişken tablosunda (variable explorer) 5 sözlük (dictionary) barındıran bir **digits** demeti (bunc) bulunmaktadır.

Digits veri kümesi 1797 elemandan oluşmaktadır (Tablo 5.3). Her bir eleman için data, images, target nitelikleri bulunmaktadır.

nitelik	**Açıklama**	**Tür**
images	Veri kümesindeki her bir elemanın matris halindeki 8x8 görüntüsü	float64
data	Veri kümesindeki her bir elemanın vektör haline getirilmiş hali	float64
target	her bir elemanın sınıfı	int32

Tablo 5.3: Digits veri kümesinin nitelikleri

Elemanları görüntü olarak göstermek için



Şekil 5.3: digits demeti

```
goruntuNo=1
import matplotlib.pyplot as plt
plt.imshow(digits.images[goruntuNo], cmap=plt.cm.gray_r,interpolation='nearest')
```

komutları kullanılmaktadır (Şekil 5.4).

Veri kümesi içersindeki matrix formundaki görüntülerin benzerliklerini hesaplamak için her bir elemanı için vektör haline dönüştürülmüş halini X dizisine (array)

```
X = digits.data
```

komutu ile aktarılır. Benzerini bulmak istediğimiz görüntünün vektor halini alıp işleme hazır hale getirmek için

```
goruntu=4
Y=X[goruntu].reshape(1,-1)
```

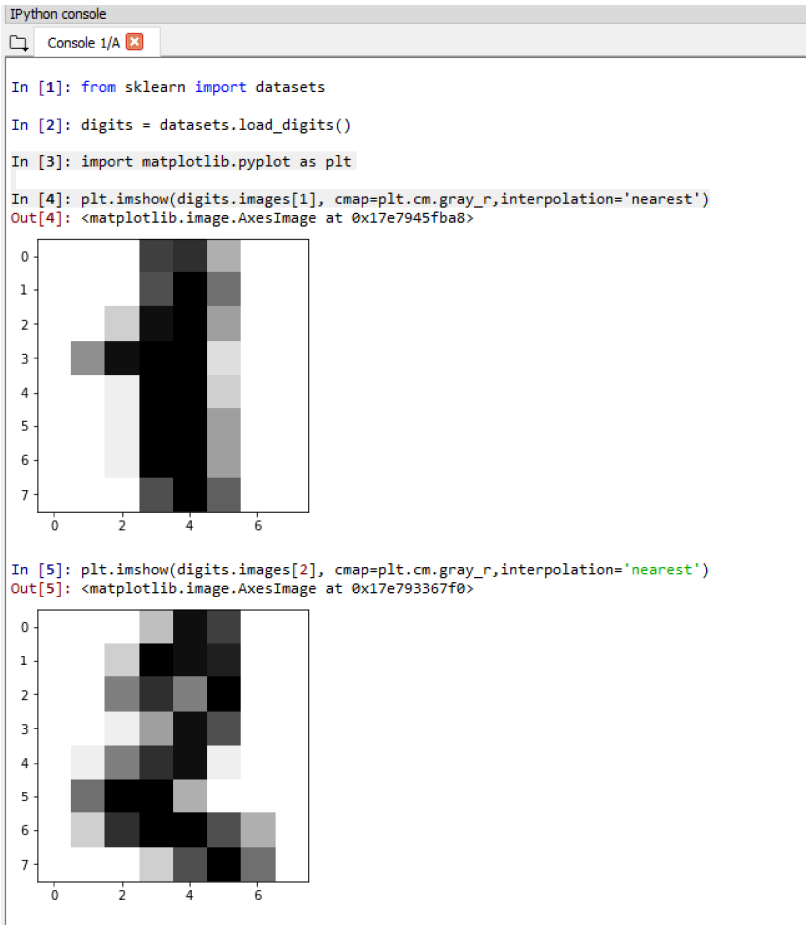
komutunu kullanılmaktadır. Örnek uygulamada 4 numaralı görüntü için işlem yapılmıştır. X dizisi bütün veri kümesinde bulunan elemanları vektör halinde bulundurmakta olup Y benzerlerini bulmak istediğimiz rakamı vektör halinde bulundurmaktadır. Kosinüs benzerlik denklemi 5.1 için

```
from sklearn.metrics.pairwise import cosine_similarity
coSim = cosine_similarity(Y, X)
```

kodu yazılabilir. Bu işlemin sonucunda elimizde benzerini bulmak istediğimiz görüntünün diğer görüntülere olana benzerlik değeri 0-1 arası değer olarak hesaplanacak ve *coSim* dizisine kaydedilmiştir. Benzerlik değerleri yüksekten düşüğe sıralamak ve ekranda göstermek için

```
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
sirali=cosf.sort_values('similarity', ascending=False)
sirali=sirali.reset_index()
```

komutları kulanılmaktadır. Şekil 5.5'den görüleceği üzere sıralanmış olan dizinin ilk elemanı görüntünün kendisi olup benzerliği (similarity) 1'e eşittir. Örnekte benzerliği bulunması istenen 4 numaralı



Şekil 5.4: Digits veri kümesindeki görüntüler

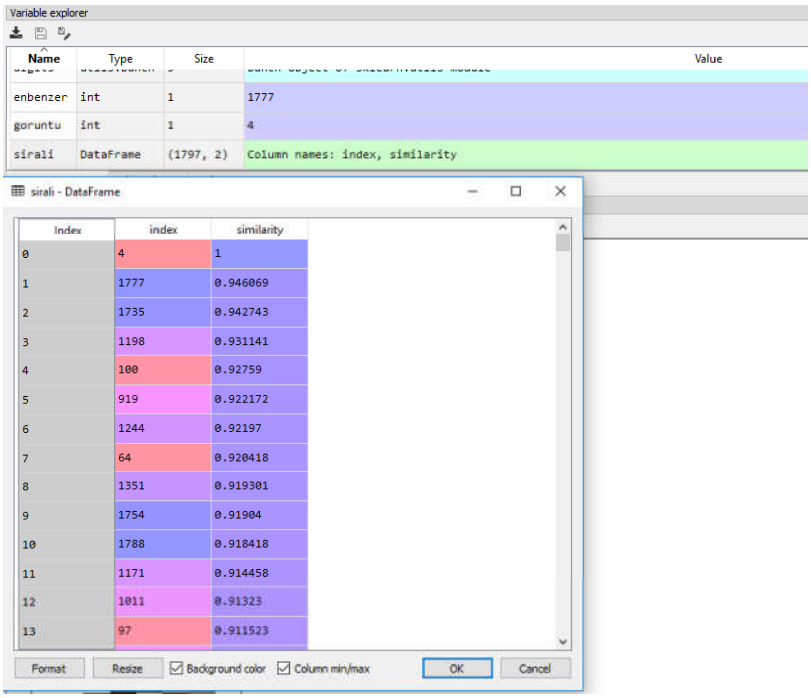
elemana en yakın örneğin benzerliği 0.946069 ile 1777 numaralı eleman olduğu görülmektedir.

Aynı şekilde uzaklık metrikleri için işlemler aynı şekilde yapılmaktadır. Öklit uzaklık metrği için kodumuz

```
from sklearn.metrics.pairwise import euclidean_distances
eucDis = euclidean_distances(Y, X)
eucDisf = pd.DataFrame(eucDism).T
eucDisf.columns = ['distance']
eucDissirali=eucDisf.sort_values('distance', ascending=True)
eucDissirali=sirali.reset_index()
```

Manhattan uzaklık metriği için kodumuz

```
from sklearn.metrics.pairwise import manhattan_distances
manDis = euclidean_distances(Y, X)
manDisf = pd.DataFrame(manDis).T
manDisf.columns = ['distance']
manDissirali=eucDisf.sort_values('distance', ascending=True)
manDissirali=sirali.reset_index()
```



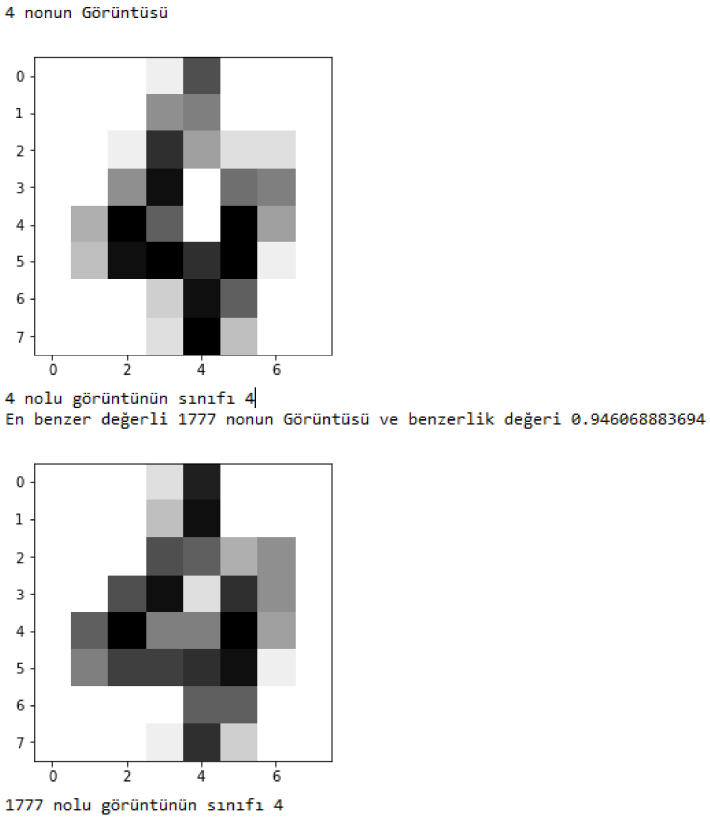
Şekil 5.5: Her bir görüntü indisine göre benzerlik değerlerinden bir kısım

şeklinde olmaktadır. Burada dikkat edilmesi gerek benzerlik ölçülerinde sıralama büyükten küçüğe olurken. Uzaklık ölçülerinde sıralama küçükten büyük olmasıdır. Bu bölümde anlatılan tüm kaynak kodlar ve fazlası yazarın github hesabından `https://github.com/mgezer` alınabilir.

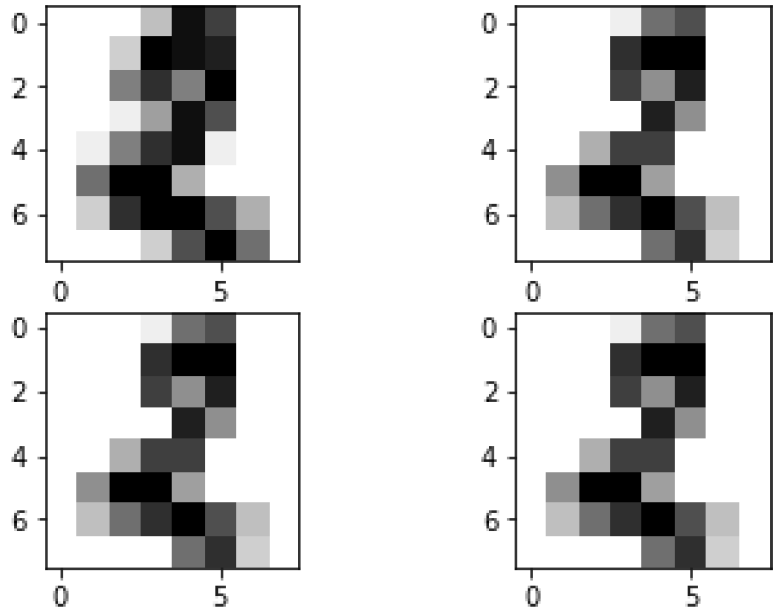
5.4 Sonuç

Bu bölüm kapsamında digit veri kümesi içersinde bulunan görüntülerdeki rakamların tanınması için benzerlik ve uzaklık ölçüleri kullanılmıştır. Şekil 5.7 de 2 nolu örnek için değişik metriklerde en iyi bulunan görüntülerin görünmektedir. şekil 5.8, şekil 5.9 ve şekil5.10 sırasıyla 5 rakam sınıfına ait olan 15 nolu görüntünün en iyi , 10. sıradaki ve 100. sıradaki benzerliklerine ait sonuçları göstermektedir. Bu şekillerden görülmektedir ki yapmış benzerlik ve uzaklık metrikleri ile yapılan sonuçlar ile uygun ve hızlı şekilde iyi sonuçlar alınmaktadır.

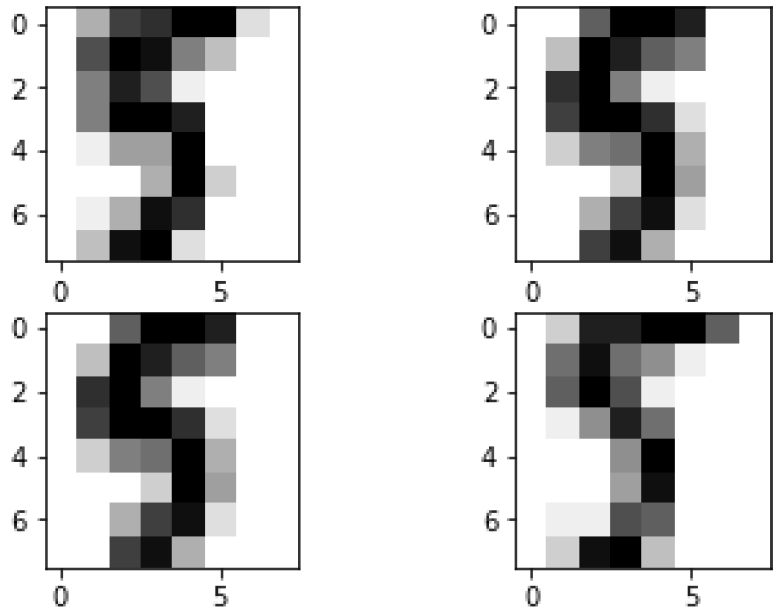
Okuyucunun kendisi geliştirmesi için kaynak kodlar sunulmuştur. Günümüzde Derin Öğrenme yöntemleri ile tanınma işlemleri çok boyutlu ve hacmi büyük veriler üzerinde çok daha etkin şekilde yapılabilmektedir.



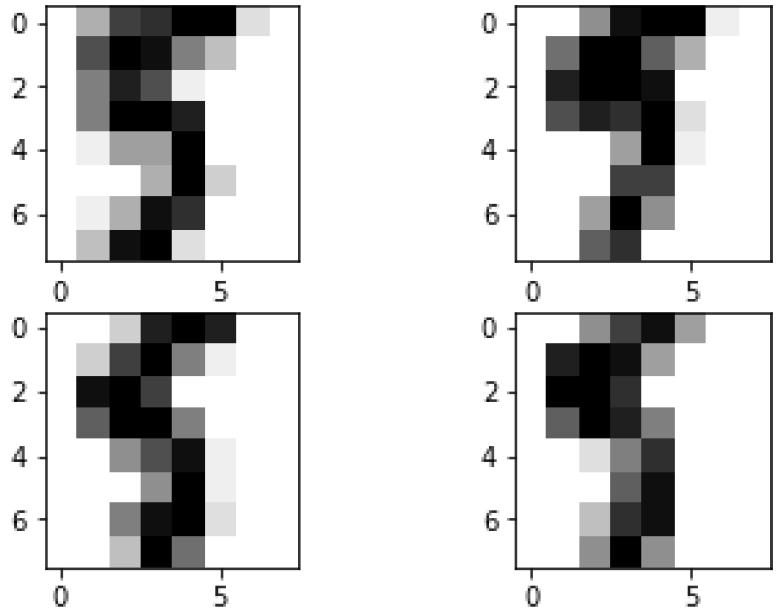
Şekil 5.6: Kosinüs benzerlik ölçütüne göre 4 numaralı görüntüye en yakın benzer görüntü



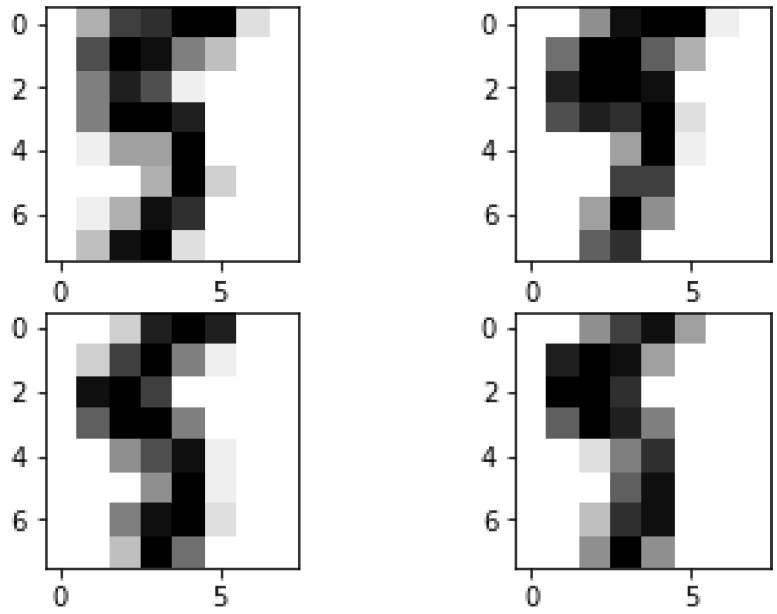
Şekil 5.7: Yukardan soldan sağa doğru 2 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan en yakın görüntü (nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (57 nolu eleman), manhattan uzaklık metriğine göre (57 nolu eleman)



Şekil 5.8: Yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan en yakın görüntü (1568 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (1568 nolu eleman), manhattan uzaklık metriğine göre (1192 nolu eleman)



Şekil 5.9: 5 rakamı olan yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan 10. sıradaki görüntü (1575 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (1659 nolu eleman), manhattan uzaklık metriğine göre (1643 nolu eleman)



Şekil 5.10: 5 rakamı olan yukardan soldan sağa doğru 15 nolu görüntü,Kosinüs benzerliğine göre bulunmuş olan 100. sıradaki görüntü (1185 nolu eleman), Aşağıdan soldan sağa doğru öklid uzaklığına göre görüntü (204 nolu eleman), manhattan uzaklık metriğine göre (1333 nolu eleman)

5.5 Kaynakça

[1] G. Oleś, "Revolution or evolution of traditional Business Intelligence concept," 2013.

[2] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," BM J. Res. Dev., vol. 3, no. 3, pp. 210–229, 1959.

[3] E. Alpaydın, Introduction to Machine Learning. Mit Press, 2014.

[4] T. M. Mitchell, Machine Learning. McGraw Hills, 1997.

[5] M. Beyeler, Machine Learning for OpenCV. Published by Packt Publishing Ltd., 2017.

[6] B. Jahne, Digital Image Processing: Concepts, Algorithms, and Scientific Applications. Springer, 1995.

[7] E. T. G. Oguz Güngör, "Dijital Görüntü İşleme." 2016.

[8] Python, "Python," Python, 2017. [Online]. Available: http://bit.ly/2x2rOcn. [Accessed: 17-Oct-2017].

[9] F. Alimoglu, "Combining Multiple Classifiers for Pen-Based Handwritten Digit Recognition," Institute of Graduate Studies in Science and Engineering Bogazici University, 1996.

[10] H. Akpınar, Data, Veri Madenciliği Veri Analizi. İstanbul: Papatya Yayıncılık, 2014.

[11] Ç. S. E. Yalçın Özkan, Biyoenformatik DNA Mikrodizi Veri Madenciliği. İstanbul: Papatya Yayıncılık, 2017.

5.6 Ekler

Kosinüs Benzerlik için yazar tarafından hazırlanmış olan kaynak kod

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  5 13:27:01 2017
@author: mgezer
"""
## Digits veritabani ile benzer goruntu bulma
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
# Veri Kumesini yukleyelim
digits = datasets.load_digits()
#display_img adinda goruntu gosterecek bir fonksiyon tanimlanmasi
def goruntuGoster(goruntuNo):
plt.imshow(digits.images[goruntuNo], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.show()
#  bazi veri kumesi elemanlarini ekranda gostereli
print("")
goruntuGoster(0)
goruntuGoster(1)
goruntuGoster(111)
# Her bir 8x8 lik goruntunun vektor haline X degiskenine aktar
X = digits.data
# Benzerlik Analizi yapilacak gorntu olan vektor sekilendiriyoruz
#satir vektorden sutuna ceviriyoruz
goruntu=4
Y=X[goruntu].reshape(1,-1)
# Cosine Benzerlik metrigini uyguluyoruz
coSim = cosine_similarity(Y, X)
"""
#sonucu Pandas Veri cercevresinin icine aliyoruz
ve en benzerden itibaren siralama yapiyoruz
"""
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
sirali=cosf.sort_values('similarity', ascending=False)
sirali=sirali.reset_index()
#ekrana bastiriyoruz
#print(sirali)
#enbenzerin indis degerini aliyoruz
print(goruntu,"nonun Goruntusu")
goruntuGoster(goruntu)
print(goruntu,"nolu goruntunun sinifi",digits.target[goruntu])
enbenzer=int(sirali.iloc[1]['index'])
bdegeri=sirali.iloc[1]['similarity']
print("En benzer degerli",enbenzer,"nonun Goruntusu ve benzerlik degeri",
bdegeri)
goruntuGoster(enbenzer)
print(enbenzer,"nolu goruntunun sinifi",digits.target[enbenzer])
```

Kaynak Kod: Değişik uzaklık ve benzerlik ölçülerine göre karşılaştırma kodu

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Sep  5 13:27:01 2017
@author: mgezer
"""
## Digits veritabani ile benzer goruntu bulma
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.metrics.pairwise import manhattan_distances
goruntu=1221
# Veri Kumesini yukleyelim
digits = datasets.load_digits()
X = digits.data
# Benzerlik Analizi yapilacak gorntu olan vektor sekilendiriyoruz
#satir vektorden sutuna ceviriyoruz
Y=X[goruntu].reshape(1,-1)
# Cosine Benzerlik metrigini uyguluyoruz
coSim = cosine_similarity(Y, X)
cosf = pd.DataFrame(coSim).T
cosf.columns = ['similarity']
cosfSirali=cosf.sort_values('similarity', ascending=False)
cosfSirali=cosfSirali.reset_index()
#ekrana bastiriyoruz
#print(sirali)
#enbenzerin indis degerini aliyoruz
enbenzer=int(cosfSirali.iloc[1]['index'])
# Oklid uzaklik metrigini uyguluyoruz
eucDis = euclidean_distances(Y, X)

eucf = pd.DataFrame(eucDis).T
eucf.columns = ['similarity']
eucfSirali=eucf.sort_values('similarity', ascending=True)
eucfSirali=eucfSirali.reset_index()
eucenbenzer=int(eucfSirali.iloc[1]['index'])
# Oklid uzaklik metrigini uyguluyoruz
manDis = manhattan_distances(Y, X)

manDisf = pd.DataFrame(manDis).T
manDisf.columns = ['similarity']
manDisfSirali=manDisf.sort_values('similarity', ascending=True)
manDisfSirali=manDisfSirali.reset_index()
manenbenzer=int(manDisfSirali.iloc[1]['index'])
plt.figure(1)
plt.subplot(221)
plt.imshow(digits.images[goruntu], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(222)
plt.imshow(digits.images[enbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(223)
plt.imshow(digits.images[eucenbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(224)
plt.imshow(digits.images[manenbenzer], cmap=plt.cm.gray_r,
interpolation='nearest')
plt.show()
```

5.7 Yazar Hakkında

Almanya doğumlu olan Murat Gezer ilk ve orta eğitiminin büyük kısmını Almanya da Lise eğitimini İstanbul'da tamamladı. Lisans eğitimini Ege Üniversitesi Fen Fakültesi Astronomi ve Uzay Bilimleri bölümünde 1997 yılında tamamlamıştır. Bir süre özel sektörde yazılım ve Ar-Ge departmanlarında çalıştıktan sonra 2007 yılında İstanbul Üniversitesi Enformatik Bölümünde Yüksek Lisansını tamamlamış ardından 2014 yılında İstanbul Üniversitesi Elektrik-Elektronik Mühendisliği bölümünden "Modern haberleşme sistemlerinde görüntü kodlaması ve sıkıştırmasında özgün matematik yöntemler" adlı tezi savunarak DR. unvanını almıştır. 2011 yılından itibaren İstanbul Üniversitesi Enformatik bölümünde Öğretim Görevlisi olarak çalışmaktadır. Çeşitli kuruluşlar tarafından düzenlenen etkinliklerde Linux, Görüntü İşleme ve Makine Öğrenmesi konusunda seminerler ve eğitimler vermektedir. Detaylı akademik çalışmalara http://bit.ly/2ytzTZa adresinden erişilebilir.

6. Akıllı Binalar

Akıllı Binalar

Doç. Dr. Tahsin TURGAY

6.1 Giriş

Dünya da teknolojik gelişimler ve ekonomik değişimler beraberinde birçok gelişimlere sebep olmuştur. 1990'lı yıllarda, bilgisayar ve telekominikasyon alt yapısında hızlı gelişmeler beraberinde bir çok alanda fayda sağlamıştır. Bilgi ve haberleşme teknolojilerilerindeki hızlı ilerleme birçok mühendislere ve tasarımcılara yeni olanaklar sunmuştur. Duyarlı çevre ve ekoloji için yapılarda akıllı bina kavramını ortaya çıkarmıştır.

Dünyamızda nüfus hızla artarken, kaynak kıtlığı, enerji krizi, göçün meydana getirdiği plansız kentleşme, yetersiz alt yapı beraberinde yaşamsal döngüde büyük tahribatlara neden olmuştur. Çevresel deformasyonlar, insan yaşamsal alanlarında bir çok kısıtlar meydana getirmiştir. Bu sorunların çözümü ise doğaya saygılı olmakla başlamaktadır. Önce çevremizi gözden geçirerek, sürdürülebilir kaynak yönetimini sağlamalıyız, kirliliği azaltmalıyız, enerji verimliliğini sağlamalıyız, doğal güzellikleri korumalıyız, çevre dostu sürdürülebilir, yeşil bina ve yaşam alanları oluşturmalıyız. Çevre dostu yaşam alanlarının oluşturulmasında en etken yöntemlerden biri akıllı bina sistemlerine önem verilmesiyle olmaktadır.

Bina tasarımı, yeni teknolojik olanaklarla beraber bir çok ayrıntaya çözüm getirecek şekilde şekillenmeye başlanmıştır. Bunlar; su, enerji, malzeme, atmosfere atılan zararlı atık tüketimini en aza indirme çerçevesinde şekillenmeye başlamıştır. Ülkemizde enerji tüketiminin yaklaşık yüzde 45'inin, kullanma suyunun ise yaklaşık yüzde 15'inin binalarda tüketildiği göz önüne alındığında yeni nesil yapılarda enerji kontrolü içeren sistemler içermelidir.

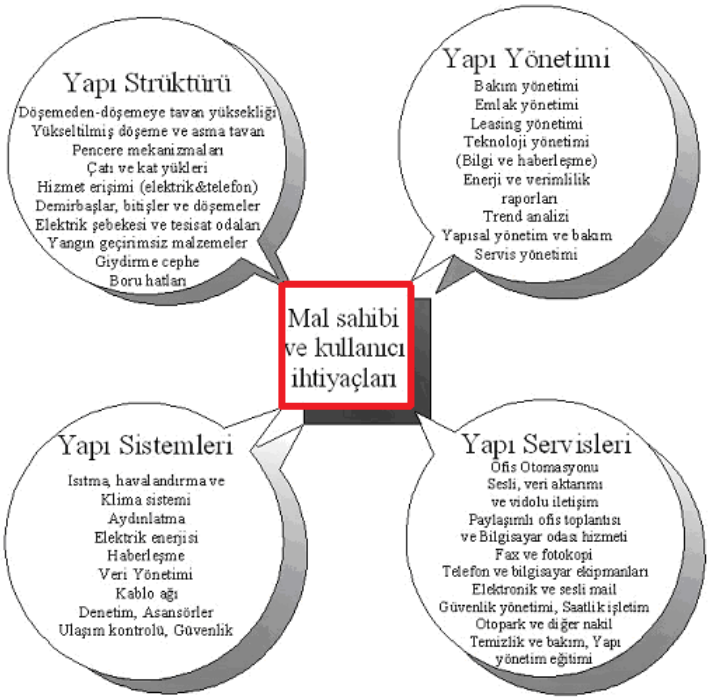
Etkin enerji kullanımının önem kazandığı günümüzde akıllı bina tasarımı, geleneksel bina tasarımından ayrı olarak birçok mesleki disiplinin bir araya gelerek binanın, hakim rüzgarlara göre konumu, güneşe olan yönünden başlayarak, iklimsel veriler gözetilerek, çevre ile olan uyumu, etkili enerji kullanımı, kullanıcıların yaşamsal konforu düşünülerek birçok kriter akıllı binanın temel kriterini oluşturur. Akıllı binalar, teknolojik bilgiyle donatılmış, kullanıcıların ihtiyaçlarını

karşılayan, farklı durumlara göre binanın önceden programlanmış bilgiler dahilinde veya yapay zeka yoluyla kendini yönetebilmesidir.

Binaya akıllılık özelliğini veren unsur binalarda kullanılan bilgisayar sistemleridir. Bu sistemler binanın iklimsel ve görsel konforunu algılayarak o çerçevede kontrol edebilen sistemlerdir. Bu sistemler kullanıcının isteğine göre programlanabilir veya uzaktan erişimle yönlendirilebilir. Bina bilgileri ve kullanıma dayalı bilgiler sistem içersinde temeli oluşturur. Bu sistemler enerji verimliliği ve ekonomik tutum sağlamakla beraber çevreye saygılıdır.

Washington ve Essex Akıllı Bina Enstitülerinin, yapmış oldukları detaylı bir tarif; birçok sistemi entegre eden; kullanıcı konforunu ve memnuniyetini arttırmak, yatırım ve işletim maliyetinde ekonomi, esneklik sağlamak amacıyla kaynakların düzenli şekilde yönetimini sağlayan, enerji tüketimini, güvenlik ve iş verimliliğini optimize etmek amacıyla, binayı yöneten ve bunun için bilgisayar teknolojisinden faydalanan binalar olarak tanımlanır.

6.2 Akıllı Bina Sistemleri



Şekil 6.1: Akıllı Bina sistem bütünü [1]

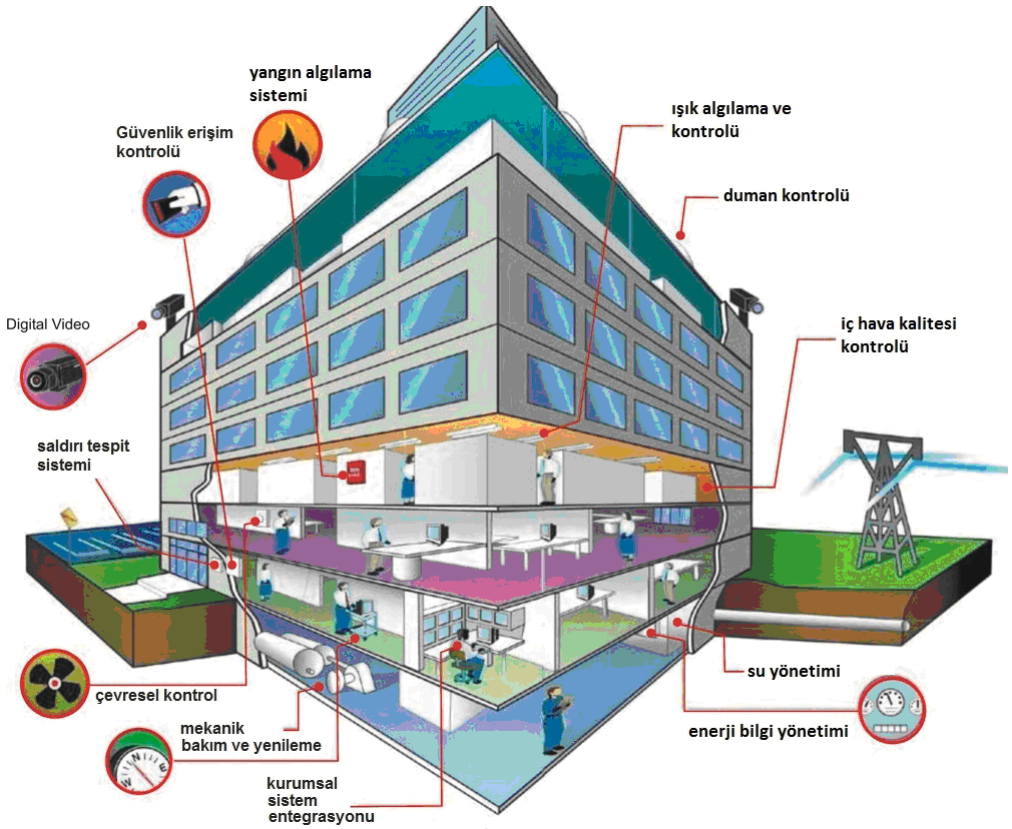
Avrupa Akıllı Bina Birliği, akıllı binayı, "binada yaşayanların etkinliğini en üst seviyeye getiren bir ortam oluştururken aynı zamanda en az donanım ve tesis maliyeti ile birlikte kaynakların en uygun şekilde kullanılmasını sağlayan bina" olarak tanımlamıştır [1].

Akıllı Binalar için CIB Çalışma Grubu WO98 raporunda (1995) şu şekilde tanım yapmıştır: Bir akıllı bina, değişen çevre şartlarına göre uyum gösterebilen bir mimari olup kullanıcısına verimli, çevresel olarak kabul görmüş koşulları sürekli bir şekilde dört temel elemanın birbirleriyle olan etkileşimi ile sağlamaktadır; otomasyon, strüktür, kontrol sistemi, kullanıcılar ve yönetimdir [2].

Asyada kullanılan akıllı bina tanımında şu koşullar yerine getirilmelidir. Bina sakinleri için en

uygun yaşam mekanı sunmak için havalandırma, güvenlik, ışık, ısı, yangın sistemlerini takip ederek kontrol eder. Binada katlar arası veri akışı sağlayan bir ağ alt yapısına sahiptir.

Tüm tanımlara bakıldığında akıllı binalarda istenen temel özellikler; çevre dostu olmalıdır, enerji tasarrufu sağlamalıdır. İnsan sağlığını ön planda tutmalıdır. Mekanı etkin ve esnek şekilde kullanabilmelidir. Bina ömrü ve maliyeti arasında optimum seviye olmalıdır. İnsan konforu ön planda olmalıdır. Güvenlik, yangın, deprem gibi tehlikelere karşı tedbir ön uyarı sistemleri olmalıdır.



Şekil 6.2: Akıllı Bina kontrol sistemleri

Akıllı ev sistemleri geliştirilirken göz önünde bulundurulan temel unsurlardan biri, bu sistemlerin kişisel bilgisayarlarla tam koordineli olarak çalışabilmesidir. Bilgisayarlar artık bir evin standartları arasına girerek, insanın haberleşme, eğlence gibi birçok alandaki alışkanlıklarına yenilikler sunmuştur. Evlerde birden fazla kişisel bilgisayar bulunması ve bunlar arasında bir ev içi bilgisayar ağı kurulması da bunun en etkili kanıtlarından biridir. Bu değişimler göz önüne alındığında, bir sonraki adım, bu bilgisayarların ev yaşantısına farklılıklar getirmesidir. En basit değişim örneği, evde bulunan ışıkların parlaklık oranları bilgisayar tarafından bir yazılım sayesinde kontrol edilebilmesidir[3].

6.3 Yapay zekaya sahip evler

Yapay zekanın önemli uygulamaları arasında olan zeki evler, dış dünyadaki değişikliklere tepki verebilen, yaşayanların davranışlarından ne zaman ne yapacağını bilen evlerdir. Zeki evlerin birçok tanımı yapılmasına karşın, buradaki zeki ev tanımı, öğrenebilen ve otomatik olarak eylemlerde

bulunabilen bir evdir. Akıllı evleri karakteristiklerine göre farklı kategorilere ayırmaya çalıştığımızda 3 ana kategoriye ayırabiliriz: kontrol edilebilir evler, programlanabilir evler, zeki evler.

Zeki evler, programlanabilir evler ile benzerlik gösterir, fakat programlanabilir evlere göre daha gelişmişlerdir. Programlanabilir evlerde senaryolar insan yardımı ile hazırlanmakta iken bu evlerde senaryo girişi yapılmaz. Bu evlerin öğrenme yeteneği vardır. Bu evler, kendi kendine inceleyip, buna göre kendi ayarlarını ve senaryolarını yaratabilen evlerdir. Bunun için öğrenme yeteneğine sahip yazılımlar, yani yapay zeka gereklidir. Evde yasayanların gün içindeki hareketlerini izlerler, tekrar eden hareketleri, ortaya çıkarırlar ve o durum için yapması gerekeni belirler ve bir daha o davranış ile karşılaşıldığında uygun ayarlamaları yapar. Bu evlerin dezavantajları vardır. Söyle ki, insan davranışlarına göre senaryo oluşturmaya çalışıldığından insan ruh halinin karmaşıklığı, her zaman aynı davranmayacağı göz ardı edilmiş olunur [4].

Farklı gelişmişlik sırasına sahip akıllı evler olduğu gibi, farklı amaca hitap eden akıllı evler de mevcuttur. Mesela, yaslı insanlar için veya fiziksel engeli olan insanlar için akıllı evler tasarlanmıştır [4].

Akıllı evlerde yazılım sistemin aynı zamanda donanım bağımsız olması gerkemektedir. Böylelikle yazılım mekanizmalarında yapılan değişiklikler üst seviyede kalır. Aynı zamanda bir kez hazırlanan yazılım, farklı farklı donanımlarla kullanılabilecektir. Zeki ev uygulamalarında kullanılan donanımlar da çok farklılık gösterebileceğinden, donanımdan bağımsız bir yazılım çok önemlidir.

Akıllı Ev Sistemlerindeki Yazılımların Genel Özellikleri:

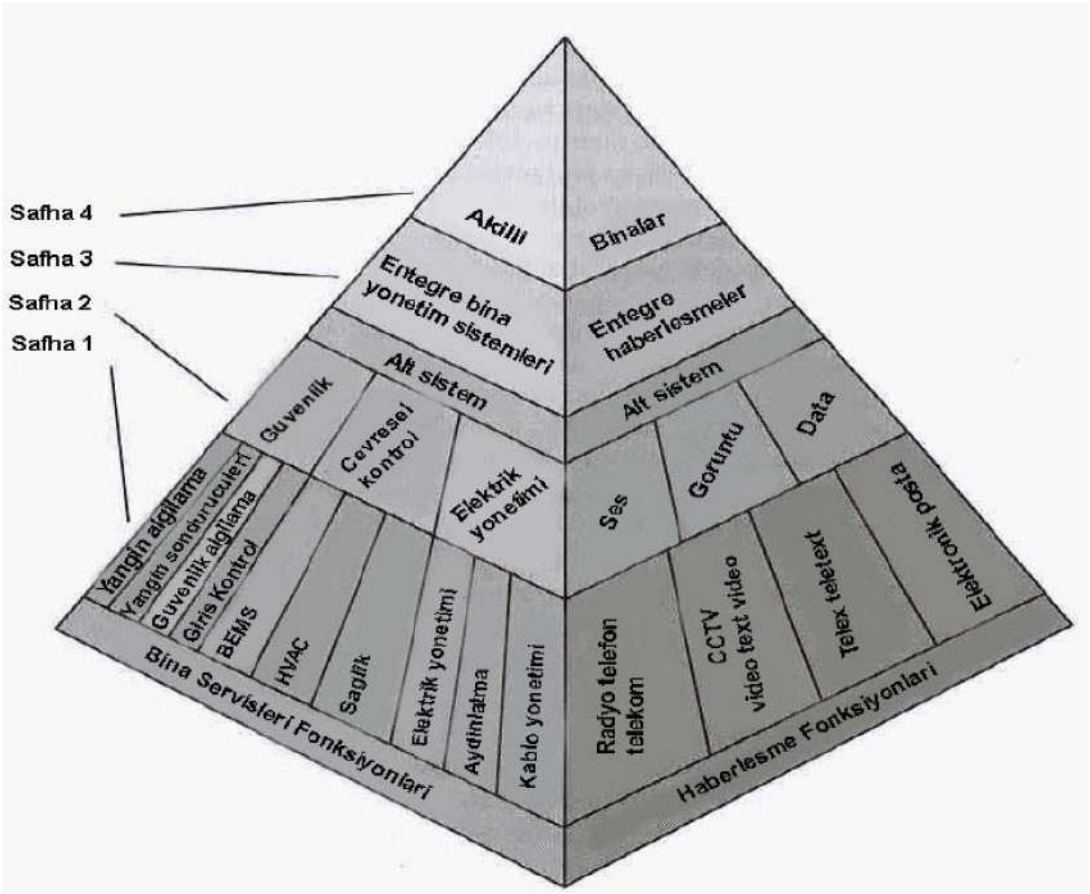
1. Gerçek zamanlı çalışabilme
2. Esnek
3. Hızlı yazılım geliştirebilme
4. Kolayca ve hızlıca yeni özellikler ekleyebilme, çıkarabilme, değiştirebilme
5. Kolayca yapay zeka kısımlarını, algoritmalarını değiştirebilme, yeni yapay zeka gerçekleştirimlerini ekleyebilme.
6. Donanım bağımsız
7. Çok sayıda sensor çeşidiyle çalışabilme [5].

6.4 Akıllı Binalarda Entegrasyon

Günümüzde binalarda kullanılan elektronik sistemlerin artması; bilişim teknolojilerindeki hızlı gelişim ve ucuzlayan maliyetler; internet kullanımının yaygınlaşması, bilgisayar sistemlerinin gelişimi, akıllı binaların teknolojik alt yapısını oluşturan sistemlerin uyum seviyesini artırarak sistemleri enerji en verimli şekilde etkileşimli çalışmasını zorunlu hale getirmektedir. Akıllı bina sistemlerinin entegrasyonu denildiğinde sadece elektro-mekanik sistemlerin entegrasyonun yanında, bina bir bütün olarak ele alınarak, binayı oluşturan pasif ve aktif bina sistemlerinin birbirlerini tamamlamaları gerekmektedir. Akıllı bina aktif alt sistemlerinin entegrasyon seviyelerinin entegrasyon safhalarının gelişiminin gösterimi Şekil 3.'de belirtilmiştir [2].

Binalarda asansörlerden kesintisiz güç kaynaklarına, yangın uyarı sisteminden jeneratörlere, normalde birbirinden kopuk, bağımsız çalışan bir dizi sistem ve cihaz ile tek bir sistem bütünlüğünde bilgi alışverişi yapılabilmesi, işletmeye farklı düzeylerde imkanlar sağllar. Binayı oluşturan tüm parçaların gerek anlık, gerek geçmiş verilerine ulaşabilen teknik yönetim, enerji optimizasyonu, periyodik bakım, arızalara (çoğu durumda oluşmadan) müdahale gibi konularda olağanüstü bir imkana kavuşur. Diğer önemli bir avantaj ise sistemler arası karmaşık otomatik senaryoların rahatlıkla kurgulanabilmesidir. Örneğin klasik olarak yangın ihbar sisteminden gelen uyarıya göre havalandırmada basit komutlar gerçekleşirken, entegrasyon sağlanmış bir sistemde ek olarak

aydınlatmanın tanımlanmış bir senaryoya geçmesi, LCD duyuru panolarında yangın bölgesine bağlı değişken kaçış duyurularının yayınlanması, otel IP TV sisteminden oda bazında uyarı yapılabilmesi, kartlı geçiş kontrollü kapılarda çıkış güvenlik seviyesinin düşürülmesi, asansörlerin zemin kata çağrılması, deprem anında ise en yakın katta kapıların açılması, gibi bir dizi ek işlem bir anda olası hale gelmektedir [6].



Şekil 6.3: Akıllı binalarda entegrasyon safhaları [2]

6.5 Akıllı Binalarda Enerji Etkin Tasarımın Sağlanması

Akıllı binaların tanımlarken önemli özelliklerin başında enerji korunumu gelir. Akıllı binalar enerjiyi %100'e yakın verimlilikle kullanmayı hedefleyen binalardır. Teknoloji sürecinin başlamasıyla enerji kaynaklarının önemi ortaya çıkmıştır. Temel hedef enerjiyi en etkin şekilde doğaya karşı saygılı şekilde kullanım olmalıdır. Nüfus artışına paralel artan sanayileşme ve teknolojik gelişmeye bağlı olarak enerji kaynaklarının hızla tükenmesine neden olmaktadır. Oransal olarak bakıldığında, Dünyada ki kullanılan toplam enerjinin %60'ını, gelişmiş ülkelerde yaşayan 1 milyar nüfus kullanırken, toplam enerjinin %40'ını diğer 5 milyar nüfus, kullanmaktadır[2].

Enerji kaynaklarını 2 grupta inceleyebiliriz.

A-Yenilenemez Enerji Kaynakları

Bu enerji kaynağını elde etmek için tükenebilir yakıt kullanılması gerekmektedir. Genelde

yenilenemez enerji kaynakları kullanımı zararlıdır. Bu kaynakların kullanımı için kullanılan yakıtlar yakıldığı zaman, doğaya zararlı atıklar ve gazlar salmaktadır.

(Fosil Yakıtlar) Kömür, petrol, doğalgaz

B- Yenilenebilir Enerji Kaynakları

Yenilenebilir enerji kaynakları, fosil yakıtlar ve yenilenemez enerji kaynaklarına göre daha az zararlı veya zararı olmayan enerji kaynaklarına denir. Bu enerji kaynakları, kendisini doğada sürekli yenileyebilen enerji kaynakları'dır. Tükenmeyen enerji kaynakları olarakta adlandırılır.

Rüzgar, jeotermal, güneş, hidrolik, biyokütle, hidrojen enerjisi gibi enerji kaynaklarıdır.

Küresel ısınmanın ve kullanılmakta olan fosil enerji kaynaklarının tükenmek üzere olmasının getirdiği koşullar içinde alternatif enerji arayışları sürmektedir.

Gelişmişlik düzeyi yüksek ülkelerin en önemli ihtiyaçlarının başında gelen enerji tüketimi, sürekli artmakta ve bu artış gelecekte de devam etmektedir. Bugün sahip olduğumuz teknolojik gelişmelerin devam etmesi ve sunduğu imkanların yaşamımızda sürmesi için doğrudan ve dolaylı olarak enerji tüketmek zorundayız. Enerji üretiminde fosil kaynak kullanımının devam edebilme olanağının kalmadığı, kabul edilmesi gereken bir gerçektir. Bu bağlamda çevremizin kendi doğal ürünü olan yenilenebilir enerji kaynaklarının kullanılmasının arttırılması gerçeği her geçen gün daha iyi anlaşılmaktadır [7].

Doğaya saygılı enerji kaynaklarının kullanım ihtiyacı arttıkça, yeni enerji kaynakları konusunda yapılan araştırma faaliyetleri de günden güne artmaktadır. Temiz enerji kaynaklarını kullanmanın avantajları ise herhangi bir enerji hammaddesinden tam bağımsızlıktır. Günümüzde sınırsız ve sorumsuzca enerji tüketiminin yerini, ihtiyacı karşılamaya yönelik, çevreye duyarlı enerji kullanımı almıştır. Enerji tüm Dünyada önemli stratejik konulardan biridir. Enerjinin çok önemli olduğu bir çağda, yapılarımızda konfor seviyesini, en verimli enerji kullanan bina tasarımları belirleyecektir. Tükenebilir enerji kaynakları henüz bitmeden binlarımızda yenilenebilir enerji kaynaklarına yönelmeliyiz. Bunun içinde yenilenebilir enerji kullanan, doğaya saygılı, akıllı bina tasarımları yapmalıyız.

6.5.1 Akıllı Ev Otomasyon Uygulamaları

Akıllı ev veya bina otomasyonlarını dört ana başlıkta toplayabiliriz: Aydınlatma, güvenlik, ısıtma-soğutma-havalandırma otomasyonu, ses ve görüntü sistemleri otomasyonudur [8].

6.5.2 Akıllı Ev Otomasyon Sisteminin Avantajları ve Dezavantajları

Her sistemin avantaj ve dezavantajları olacağı gibi Akıllı Ev Otomasyonunun da hem avantajları hem de dezavantajları vardır. Bu avantajlar ve dezavantajlar bir sonraki sayfada maddeler halinde verilmiştir.

Akıllı ev otomasyonunun avantajları:

- Akıllı Ev Otomasyonunun en büyük avantajı güvenliğimizin sağlanmasıdır.
- Kontrolü sağlanan sistemler sayesinde daha rahat bir yaşam sürdürülmesini sağlar.
- Zaman ve enerji tasarrufu sağlar.
- Bedensel ve fiziksel engelli hastaların daha rahat yaşamlarını sürdürmelerini sağlar.
- İnsanlara konfor sağlar.

Akıllı ev otomasyonunun dezavantajları:

- Kullanımı rahat olduğundan dolayı bir süre sonra insanları tembelleştirir.
- Uzaktan kontrol olduğundan dolayı diğer kişilerin sisteme girmesi olumsuz sonuçlar yaratır.
- İnsanların mekanikleşmesine neden olabilir.
- İşsizliği artırır [8].

6.5.3 Akıllı Ev Otomasyonun Sisteminde Yapılabilecek Kontroller

Akıllı ev otomasyonu insanların günlük yaşantısını etkileyen pek çok parametreyi kontrol altına alır. Kontrol altına alınan bu parametreler sayesine insanlar mutlu, huzurlu ve daha az stresli yaşamaktadırlar. Bu parametreler nem, aydınlatma, yangın, sıcaklık gibi parametrelerdir[8].

Aydınlatma

Akıllı ev otomasyonunda kontrol edilen parametrelerden biri aydınlatmadır. Aydınlatıcının ışık şiddeti insanların göz sağlığı üzerinde önemli bir etkiye sahiptir. İyi bir aydınlatma insanların hem evlerinde hem de ofislerinde verimli olmasını sağlamak amacıyla önemlidir. Aydınlatma kontrolünde çeşitli yöntemler vardır. Uzaktan kontrol sistemleri ve sensörler bu kontrol yöntemlerindendir. Akıllı ev otomasyonunda yapılan aydınlatma kontrolü sayesinde enerji için yapılan harcamalarda tasarruf edinilmiş olunur.

İklimlendirme

Akıllı ev otomasyonunda kontrol edilen parametrelerden bir diğeri iklimlendirmedir. Günümüzde insanlar zamanının büyük bir kısmını kapalı mekânlarda geçirmektedirler. Bu mekânlarda yaşayan insanların fazlalığından dolayı ortamın daha ferah olması gerekmektedir. İklimlendirme sayesinde kapalı alanlardaki insanların daha konforlu, daha verimli zaman geçirilmesi sağlanır. İklimlendirme sıcaklık, havalandırmanın ve nemin birlikte kontrolü demektir. İklimlendirme sayesinde ekonomik açıdan tasarruf edinilmiş olunur.

Yangın

Bir yangın anında en önemli unsur yangın çıktığı an bilginin gerekli yerlere en kısa zamanda ulaştırılmasıdır. Akıllı ev otomasyonunda bulunan yangın detektörleri sayesinde yangına müdahale zamanı en aza indirilmiş olunur. Yazılan program sayesinde de itfaiyeye haber verilerek yangına olabildiğince en az zamanda müdahale edinilmiş olunur.

Perde ve Panjur Kontrolü

Bina otomasyonları sisteminde motorları kontrol etmek artık çok kolaylaşmıştır. Bir yere gidildiğinde evde bir perdeleri kapamayı unuttuğumuzda internet veya telefon aracılığı ile perdeleri kontrol altına almış oluruz. Bu sayede evin güneşlendirilmesi sağlanır [8].

6.6 Sonuç

Günümüz, teknolojisi hızla ilerlerken, yapılarda konfor düzeyi artmaktadır. Konfor düzeyindeki artış çok daha fazla enerji tüketimine neden olmaktadır. Dünyamızda, petrol ve kömüre dayalı enerji kaynaklarının tüketimi, dünyanın doğal kaynaklarının, ormanların, denizlerdeki biyolojik çeşitliliğin yok olmasına sebep olmuştur. Dünyadaki enerji sorununu temiz enerji yoluyla yani yenilenebilir enerji kaynaklarıyla aşmak mümkündür. Enerji kaynaklarının fazla oranda kullanılması Dünya'da enerji krizlerine neden olmuştur. Dünyada enerji kullanımının %45 i binalar tarafından tüketilmektedir. Bir binanın toplam enerji etkinliği birçok faktöre bağlıdır. Bu faktörler arasında;

ısıtma ve havalandırma alanlarında kusursuz işlev gören zamanlama programları, optimize edilmiş açma ve kapama işlevleri, ihtiyaca bağlı olarak hazır bulundurulan enerji, kusursuz ayar parametreleri, optimum tesis boyutlandırması, kusursuz hidrolik ayarlama ve düzenlemesi ve daha birçok faktör bulunmaktadır. Bütün bu faktörler, bir sistemin enerji etkinliğine önemli ölçüde etki eder. Bu sebeple enerji kullanımını en verimli şekilde kullanımı sağlamak amacıyla bina sistemlerinde çalışmalar yapılmaktadır. Binalarda enerji etkin tasarım ön plana çıkmıştır. Doğal enerji kaynaklarının, korunması için pasif ve aktif bina alt sistemlerini, en uygun şekilde kullanılmasını sağlayan akıllı bina sistemleri önem kazanmıştır. Günümüzde doğaya saygılı, sürdürülebilir bir çevre için yapay zeka ile yönlendirilen akıllı binaların gerçekleşmesi zorunluluktur.

6.7 Kaynakça

[1] Olcay, O., 2007, "Akıllı Bina Kavramı ve Akıllı Bina Değerlendirme Metodları", İTÜ, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

[2] Mangan, S.D., 2006, "Akıllı Binalarda Alt Sistem Değerlendirmesi: İstanbul Örneği", İTÜ, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

[3] Yigen, O., 2003, "Ev Otomasyon Kontrolü", Elektrik Elektronik Teknolojisi, Bilgisayar ve Teknoloji Yüksek Okulu, Doğu Akdeniz Üniversitesi

[4] Stefanov, D., H., Bien, Z., Bang, W.,C., (004) "The Samart House for Older Persons and Persons with Physical disabilities: structure, technology arrangements and perspectives", IEEE Transactionson Neural Systemsand Rehabilitation Engineering, 12(2), doi:10.1109/TNSRE2004.828423

[5] www.elektroforum.org/akilli-bina-otomasyonu/556-akilli-ev-kavrami-akilli-ev-nedir.html

[6] www.bestdergisi.com.tr/arsiv/yazi/tridium-bina-otomasyon-sistemi-karylatyrmaly-ustunlukler

[7] Gedik, T., Ö, 2015, "Türkiye'de Yenilenebilir Enerji Kaynakları ve Çevresel Etkileri", İTÜ, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul.

[8] Atasoy, A., 2014, "Akıllı Ev Otomasyonu", Karadeniz Teknik Üniversitesi, Mühendislik Fakültesi, Trabzon.

6.8 Yazar Hakkında

Doç. Dr. Tahsin Turgay,, DPÜ, Mühendislik Fakültesi, İnşaat Mühendisliği lisans eğitimini (1997), Sakarya Üniversitesi Fen Bilimleri Enstitüsü, İnşaat Mühendisliği Yapı Anabilimdalında yükseklisans (2000) ve YTÜ, Fen Bilimleri Enstitüsü, İnşaat Mühendisliği Yapı Anabilimdalında doktora derecesini (2007) tamamlamıştır. ODTÜ, Mühendislik Fakültesi, İnşaat Mühendisliğinde doktora sonrası araştırmacı olarak çalışmıştır(2012).

Doç. Dr. Tahsin Turgay, Abant İzzet Baysal Üniversitesinde, Yıldız Teknik üniversitesinde, Ortadoğu Teknik Üniversitesinde çeşitli akademik görevlerde bulunmuştur. Sakarya Üniversitesi, Sanat Tasarım ve Mimarlık Fakültesi, Mimarlık Bölümü kurucu bölüm başkanıdır. Yazarın, yapı mühendisliği, yapı mekaniği, yapılarda bünyesel modelleme, yapıların performansı, yapıların onarım veya güçlendirilmesi, mimaride taşıyıcı sistemler, tarihi yapıların taşıyıcı sistemleri, sürdürülebilir mimarlık alanlarında çeşitli araştırmaları mevcut olup çalışmaları devam etmektedir.

7. Genetik Algoritmalar

Genetik Algoritmalarla Sezgisel Optimizasyon

Dr. Emre AKADAL

7.1 Giriş

Genetik algoritmalar (GA), doğal yaşam sürecini taklit ederek çalışan ve optimizasyon problemlerinde sıklıkla başvurulan bir arama yöntemidir. John Holland tarafından ortaya atılan fikir [1], Holland'ın, David Goldberg ve diğer öğrencileri tarafından geliştirilmiş [2], diğer bilim insanların geliştirme ve uygulamalarıyla popüler bir hal almıştır.

Sezgisel yöntemler, optimizasyon problemlerinin çözüme ulaştırılması için sıklıkla kullanılmaktadır. Ele alınan bir problem için çözüm kümesinin tamamının taranması pratikte pek mümkün olmayabilir. $(-10,+10) \in R$ aralığında yapılacak bir gerçek sayı aramasında mümkün tüm durumların taranması, çok uzun süre ve bilgisayar gücü gerektirecektir. Ancak sezgisel yöntemler tüm mümkün çözümleri denemeden çözüme ulaşmayı hedeflediği için çoğu problemi hızlı ve yüksek bilgisayar gücü gerektirmeden çözüme ulaştırabilmektedir. Bu durum da sezgisel yöntemlerin önemini ortaya koymaktadır.

Kitabın bu bölümünde Klasik GA'nın çalışma prensibinden, öne çıkan kodlama tiplerinden, genetik operatörlerden ve amaç fonksiyonu hazırlanmasından bahsedilecek; farklı kodlama tipleri için hazırlanmış örnek uygulamalar sunulacaktır. Bölüm içerisinde uygulamalar için R dili [3] ve bu dil için hazırlanmış GA paketi [4] kullanılmaktadır. GA özel bir terminolojiye sahip olduğu için mümkün olduğunca terimlerin ilk kullanıldıkları yerde parantez içinde eş anlamları da sunulmuştur. Bu kitap bölümünde bahsedilen özellikler klasik GA özellikleri olmakla birlikte gerçekleştirilen çok sayıda çalışma bu özellikleri iyileştirmek üzerine öneriler sunmuştur. Bu sebeple GA dünyasının bu bölümde sunulandan çok daha geniş olduğu bilinmelidir.

7.2 Genetik Algoritmalara Genel Bakış

GA'nın doğayı taklit ederek arama yaptığından bahsedilmişti. Daha ayrıntılı ele alındığında GA'nın doğayı taklit ettiği ilk nokta, yinelenen (iterasyonel) bir süreç olmasıdır. Her bir yineleme, GA için bir nesli temsil etmektedir. Bitirme koşulu sağlanana kadar GA, yeni nesiller oluşturarak arama yapmaktadır. Her bir nesilde GA operatörleri tekrar uygulanmakta ve bu sayede yeni nesil oluşturulmaktadır.

Nesiller bir topluluktan (populasyon / population) oluşmaktadır. Bir nesilde birden fazla topluluk öneren çalışmalar olsa da klasik GA için her bir nesil, bir topluluktan meydana gelmektedir. Topluluk ise bireylerden meydana gelmektedir. Topluluk içerisinde bulunan her bir birey (kromozom) gerçekte GA için bir aday çözümü (candidate solution) ifade etmektedir.

GA'nın doğayı taklit ettiği bir diğer nokta "çevreye uyum sağlayanın hayatta kalması" prensibidir. Çevre ile kastedilen koşullar GA içerisinde amaç fonksiyon ile belirlenir. Bir diğer deyişle bireylerin uyum sağlaması gereken çevre, amaç fonksiyonunun kendisidir. Topluluğu oluşturan her bir birey, çevreye yani amaç fonksiyona uyum sağlayabildiği oranda sonraki nesillerde yer alma şansı kazanmakta ve yeni bireyler meydana getirebilmektedir. Amaç fonksiyonları bir bireyi girdi olarak alan, optimize edilmek istenilen probleme göre bu birey ile elde edilen sonucu başarı ya da ceza puanı olarak geri döndüren fonksiyonlardır. Amaç fonksiyonu başarı puanı döndürüyorsa ele alınan problem maksimizasyon, ceza puanı döndürüyorsa minimizasyon problemi olarak görülebilir.

Her bir iterasyonda, eldeki topluluğa seçilim, çaprazlama ve mutasyon operatörleri belirlenen olasılıklarla uygulanmakta ve yeni nesil elde edilmektedir. GA araması, belirlenen iterasyon sayısına ulaşıldığında ya da beklenen optimum sonuca erişildiğinde sona ermekte ve bu sonuca ulaşmayı sağlayan birey arama sonucu olarak elde edilmektedir.

GA, ele alınan problemden bağımsız olarak çalışmaktadır [5]. Her bir bireye karşılık sayısal bir değer üretebilen amaç fonksiyonunun yazılmış olması yeterli görülebilir. Problemin içeriği, GA'nın çalışma mekaniğine etki etmemesi beklenmektedir. Bu durum GA'nın uygulanabilirliğini önemli ölçüde arttırmaktadır.

7.3 Genotip ve Fenotip

GA aramasındaki her bir bireyin bir çözüm önerisi ifade ettiği ve içerdiği belirtilmişti. Klasik GA'larda her bir birey $0, 1$ alfabesini kullanan bir karakter dizilimi olan kromozomdan meydana gelmektedir. Kromozomun her bir karakterine bit, her bir anlamlı parçasına gen adı verilmektedir. Aşağıda bir kromozom örneği sunulmuştur.

$$0001110101011100010101$$

Bir kromozom, örtük olarak amaç fonksiyonuna iletilecek girdi değerini taşımaktadır. Amaç fonksiyonu içerisinde kromozom, probleme özgü olarak değerlendirilebilmek üzere ele alınmaktadır. Kromozomun GA için işlenebilir haline genotip (genotype), amaç fonksiyonu içinde değere dönüştürülmüş haline ise fenotip (fenotype) adı verilmektedir [6].

Genotip, ele alınan problemde değeri belirlenmek istenilen karar değişkeninin alabileceği değerlerin kümesine bağlı olarak fenotipe dönüştürülebilir. Genotip-fenotip dönüşümünün en basit örneği, kromozomun 2 tabanında yazılmış bir sayı olarak kabul edilmesi ve 10 tabanına dönüştürülmesi olarak görülebilir. Bu duruma örnek olan bir dönüşüm aşağıdaki gibidir.

$$(1011010110001001)_2 = (46473)_{10}$$

Böyle bir kromozom yapılanması ve fenotip dönüşümü 2 kısıtı beraberinde getirmektedir. Birinci kısıt, çözüm kümesinin büyüklüğüdür. Kromozomun taşıyabileceği farklı anlam sayısı, kaç bitten oluştuğu ile orantılıdır. n bitten oluşan bir kromozom, en fazla 2^n farklı değere dönüştürülebilir. Dolayısıyla arama yapılacak çözüm kümesinin büyüklüğüne baştan karar verilerek kromozom uzunluğunun buna göre seçilmesi gerekmektedir. İkinci kısıt ise bu dönüşüm ile elde edilebilecek tüm değerlerin doğal sayılar kümesinin elemanı olmasıdır. Bu kısıta karşılık olarak, aşağıdaki uyarlama ifadesi kullanılarak elde edilen değerin istenilen aralığa uyarlanması sağlanabilmektedir.

$$(\frac{max - min}{2^L - 1}) \times z$$

Böylece ele alınan bir 1011001011 kromozomu, 10 sayı tabanına dönüştürüldüğünde 715 değerini döndürürken fenotipin $[0, 20]$ aralığına uyarlanmasıyla $13,9784946237$ değerine eşit olmaktadır. Alınabilecek değer aralığı aynı kalırken kromozom uzunluğunun (bit sayısının) arttırılması, elde edilecek uyarlanmış değerin daha hassas sonuca sahip olmasını sağlayacaktır [7].

Optimize edilmek istenen problem, her zaman matematiksel bir fonksiyon olmayabilir. Bir labirent çözme problemi düşünüldüğünde, her bir kromozom labirentten çıkmayı deneyen hareketler sırasını gösteren bir aday çözümü ifade edecektir. Bu problemde fenotip oluşturulurken kromozom, anlamlı hareket parçalarına dönüştürülmelidir.

Yalnızca 4 farklı hareketin olduğu varsayılır ve her bir hareket bir gen ile ifade edilirse aşağıda verilen anlamlandırma tablosu kullanılabilir.

Tablo 7.1: Labirent Problemi İçin Genlerin Anlamları

Gen	**Anlamı**
00	İleri
01	Sağa
10	Sola
11	Geri

Bu problemin çözümü için aşağıdaki gibi bir rastgele aday çözüm ele alınabilir.

$$10000100000111100101$$

Genotip olan bu kromozomun değerlendirilebilmesi için fenotip dönüşümü gerçekleştirmek gerekmektedir. Bu aşamada eldeki anlam seti 2 bitlik genlerden oluştuğu için kromozomun 2 bitlik genlere bölünmesi uygun olacaktır.

$$10 - 00 - 01 - 00 - 00 - 01 - 11 - 10 - 01 - 01$$

Her bir gen için tablo içerisindeki anlamlardan faydalandığında ele alınan kromozom için elde edilecek fenotip aşağıdaki gibi olacaktır.

$$sol - ileri - sag - ileri - ileri - sag - geri - sol - sag - sag$$

Fenotip dönüşümünden sonra aday çözüm amaç fonksiyonu tarafından değerlendirilebilir biçime ulaşmıştır. Bu başlık altında verilen örneklere dikkat edildiğinde GA için söylenilen "problemden bağımsız çalışma" özelliği daha anlaşılabilir olacaktır. GA, belirlenen sayıda bitten oluşan kromozomlar üreterek genetik operatörleri çalıştırır. Kromozomların alacağı başarı ya da ceza puanı, amaç

fonksiyonu içerisindeki fenotip dönüşümü sayesinde üretilmektedir. Bu sebeple uygulayıcının asıl görevi problemi uygun şekilde amaç fonksiyonuna dönüştürmektir.

7.4 Genetik Operatörler

Genetik operatörler, GA'nın her bir nesilde topluluktaki bireyler üzerinde uyguladığı işlemleri ifade etmektedir. Tüm operatörler başlangıçta belirlenen olasılıklarla (uygulayıcı belirlemezse ön tanımlı değerlerle) uygulanırlar. Klasik GA için 3 temel operatör bulunmaktadır [2]. Bu operatörler aynı zamanda GA'nın doğayı taklit etme biçimidir.

Seçilim operatörü, hayatta kalabilme; çaprazlama operatörü, üreme; mutasyon operatörü ise bir kromozomun mutasyona uğramasını yani rastgele bir özelliğinin değiştirilmesi işlemini uygulamaktadır.

Alt başlıklar ile bu 3 temel operatörün çalışma prensibi anlatılacak, GA ile arama sürecine etkisi tartışılacaktır.

7.4.1 Seçilim Operatörü

Seçilim operatörü, GA içerisinde "ortama uyum sağlayanın hayatta kalması" prensibinin uygulanmasıdır [8]. Toplulukta yer alan tüm bireyler, amaç fonksiyonu ile değerlendirilerek bir uygunluk değerine sahip olurlar. Seçilim operatörü, rastgele seçtiği n sayıda kromozomu uygunluk değerine göre değerlendirerek hangilerinin sonraki nesilde yer alması gerektiğini belirler.

Örnek olarak 4 kromozomdan oluşan bir topluluk olduğunu, bu kromozomların 10 sayı tabanına dönüştürülerek fenotiplerinin elde edildiğini farz edelim.

Tablo 7.2: Örnek Topluluk ve Uygunluk Değerleri

Kromozom	**Uygunluk Değeri**
0100101101	301
1011100010	738
0000110110	54
1101011100	860

Seçilim operatörünün 2'li değerlendirmeler yaptığını varsayarsak;

0100101101 301
1011100010 738

karşılaşmasını 738 uygunluk değerine sahip kromozom kazanacaktır. Eğer seçilen 2'li,

0100101101 301
0000110110 54

olursa sonraki nesilde yer alma ve yavru kromozomlar üretme hakkında sahip olan kromozom 301 uygunluk değerine sahip olan olacaktır.

Bunun yerine tüm topluluk üyelerini uygunluk değerlerine göre sıralanarak belirlenen sayıda kromozom doğrudan sonraki nesle kopyalanabilirdi. Ancak bu yöntem, genel (global) optimum çözüme ulaşmada zorluk yaratabilir. Bu sebeple seçilim operatörü rastgeleliği de içererek, düşük ihtimalle de olsa uygunluk değeri düşük kromozomların yeni nesilde bulunabilmesine olanak sağlamaktadır.

Mümkün olasılıklar incelendiğinde yalnızca en kötü uygunluk değerine sahip kromozomun yeni nesilde yer almayacağı söylenebilir. Topluluktaki diğer tüm kromozomlar farklı olasılıklar dahilinde yeni nesilde yer alma şansına sahiptir.

Seçilim operatörünün uygulanma yöntemi temel olarak yukarıda anlatıldığı gibi olsa da literatürde zaman içerisinde önerilen çeşitli seçilim yöntemleri bulunmaktadır [9]. Bunlar şu şekilde sıralanabilir:

- Rulet Tekeri (Roulette Wheel) ve Olasılıksal Evrensel (Stochastic Universal) Seçme İle Uygunluk-Orantılı Seçilim
- Sigma Derecelendirmesi (Sigma Scaling)
- Boltzmann Seçilimi (Boltzman Selection)
- Sıralandırma Seçilimi (Rank Selection)
- Turnuva Seçilimi (Tournament Selection)
- Sabit Durum Seçilimi (Steady-State Selection)

Seçilim operatörünün uygulanması sonucunda seçilen kromozomlar çaprazlama operatörüne iletilirken seçilmeyen kromozomlar topluluktan elenirler.

7.4.2 Çaprazlama Operatörü

Çaprazlama operatörü, seçilim operatörü tarafından sonraki nesillerde yer almasına karar verilen kromozomların yavru aday çözümler üretmesini sağlayan işlemleri gerçekleştirir. Bu sayede uygunluk değeri daha iyi bir ya da daha fazla kromozom elde edilmesi beklenmektedir [10].

Çaprazlama operatörü, rastgele ele aldığı iki kromozomu rastgele bir ya da birden fazla noktadan böler ve çapraz olarak tekrar birleştirerek yeni kromozom(lar) oluşturur. Eldeki kromozomlar aynı şekilde topluluk içerisinde yer alırken oluşturulmuş yeni kromozomlar da topluluğa eklenir.

0100101101 Uygunluk değeri: 301
0000110110 Uygunluk değeri: 54

Yukarıdaki iki kromozomun çaprazlanmak üzere ele alındığını düşünelim. Gösterim kolaylığı açısından tek noktadan çaprazlama yöntemi kullanılacaktır. Çaprazlama noktası olarak da 2. karakter sonrasını kabul edelim. Bu durumda kromozomlar aşağıdaki şekilde bölünecektir.

01 / 00101101
00 / 00110110

1. kromozomun 1. kısmıyla 2. kromozomun 2. kısmı; 2. kromozomun 1. kısmıyla 1. kromozomun 2. kısmı birleşerek aşağıdaki kromozomları elde edebiliriz.

0100110110 Uygunluk değeri: 310
0000101101 Uygunluk değeri: 45

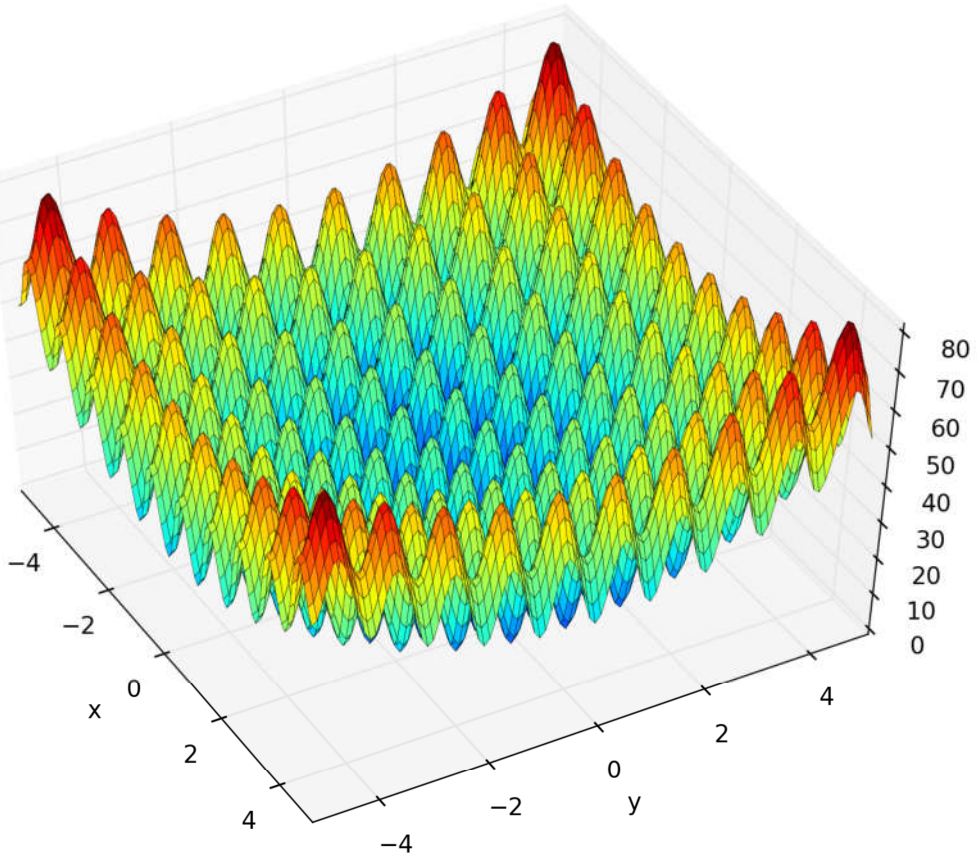
Yeni yavru kromozomların uygunluk değerlerine bakıldığında 1. kromozomun, çaprazlamaya giren iki kromozomdan da daha iyi bir uygunluk değerine sahip olduğu; 2. kromozomun ise iki kromozomdan da daha kötü bir uygunluk değerine sahip olduğu görülmektedir. Bu durumda 2. kromozom yüksek olasılıkla bir sonraki nesilde yer alamayacaktır. 1. kromozomun ise çaprazlamaya giren kromozomlardan daha iyi bir uygunluk değerine sahip olduğu görülmektedir. Bu sayede çaprazlama operatörüyle eldeki en iyi çözümden daha iyi bir çözüme ulaşıldığı görülmek-

tedir. 1. Kromozom yüksek olasılıkla sonraki nesilde yer alacak ve çaprazlamaya girerek daha iyi kromozomlar oluşturma şansı yakalayacaktır.

7.4.3 Mutasyon Operatörü

Düzgün bir arama yüzeyine sahip fonksiyonlarda optimuma ulaşmak nispeten daha kolaydır. Seçilen herhangi bir noktanın çevresi incelendiğinde arama yüzeyinin hangi yönde hangi değerleri ürettiği belirlenebilir ve aranılan noktaya (minimum ya da maksimuma) kolayca ulaşılabilir. Ancak ele alınan tüm problemlerin arama yüzeyi kolayca taranabilecek biçimde olmayabilir. Şekilde, rastrigin fonksiyonunun $x, y \in [-5, 5]$ aralığında aldığı z değerlerini gösteren grafik sunulmuştur.

$$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$$



Şekil 7.1: Rastrigin Fonksiyonuna Ait Grafik

Rastrigin fonksiyonu, optimizasyon yöntemlerini test etmek için kullanılan bir minimizasyon problemidir. Grafik incelendiğinde arama yüzeyinin çok fazla girinti ve çıkıntıya sahip olduğu görülmektedir. Bu sebeple ulaşılan bir optimum noktanın yakın çevresine bakarak global optimumun hangi yönde olduğuna karar vermek oldukça zordur.

GA araması sırasında topluluk bir minimumda yoğunlaşmış olabilir. Buna yerel optimuma takılma adı verilir. Global optimuma ulaşabilmek için bazı aday çözümlerin yerel optimum dışında

daha iyi bir uygunluk değeri araması gerekmektedir. GA'nın yerel optimuma takılmasını engellemek için mutasyon operatörü kullanılmaktadır. Mutasyon operatörü çok düşük bir olasılıkla rastgele seçilen bir kromozomun rastgele seçilen bir özelliğini değiştirir [10]. Klasik GA için özellikten kasıt bitlerdir. Değiştirmekten kasıt ise 1'i 0, 0'ı 1 yapmaktır. Bu sayede topluluk içerisindeki bir kromozom, çözüm uzayının rastgele farklı bir noktasına sıçrar. Eğer şans eseri daha iyi bir çözüme ulaşırsa seçilim ve çaprazlama operatörleri sayesinde daha iyi bölgede yeni kromozomlar oluşturur. Amaç her zaman en uyguna yani global optimuma erişmektir.

1011000010

Yukarıdaki kromozom üzerine mutasyon operatörünün uygulanacağını farz edelim. Ele alınan kromozomun seçilen rastgele bir biti (burada 6. bit olsun) tersine çevrilmelidir. Yukarıda verilen kromozomun 6. biti 0'dır. Bitin 1 olarak değiştirilmesi gerekir. Operatör uygulandıktan sonra yeni kromozom aşağıdaki gibi olacaktır.

1011001010

GA'nın başarılı bir arama süreci gerçekleştirebilmesi için genetik operatörlerin dengeli bir şekilde uygulanması gerekmektedir. Mitchell, bu dengeden keşif/faydalanma (exploration/exploitation) dengesi olarak bahsetmektedir [2]. Keşif, arama yapılan tüm çözüm uzayının taranması, hiç ulaşılmamış alan kalmaması gerekliliğidir. Faydalanma ise bulunan bir optimumun yakın çevresinde daha iyi bir optimum çözümün olup olmadığının aranmasıdır. Tüm uzayın taranmış olması, bulunan en uygun bölgenin ise iyi irdelenerek en uygun noktanın belirlenmesi GA'nın başarısı açısından oldukça önemlidir. Genetik operatörler bu dengeyi sağlayabilecek özelliklere sahiptir. Uygulayıcının operatörleri doğru olasılıklar dahilinde kullanması başarılı bir çözüme ulaşmasında etkili olacaktır.

7.5 Amaç Fonksiyonu

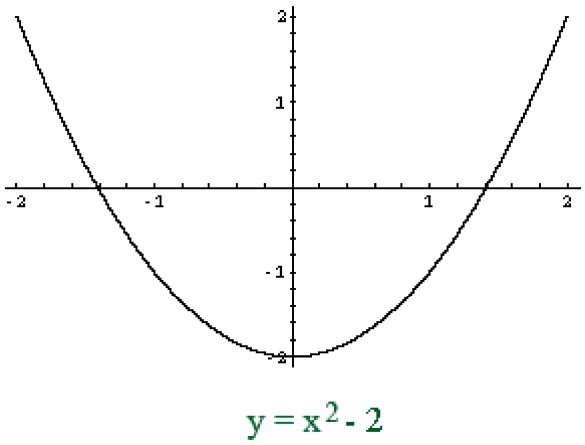
Optimize edilmek istenilen problem, her bir kromozoma karşılık sayısal bir değer üretecek şekilde GA'ya tanıtılmalıdır. Bu tanıtmaya amaç (objective) fonksiyon, uygunluk (fitness/utility/profit) fonksiyonu ya da maliyet (cost) fonksiyonu gibi isimler verilmektedir. Maliyet fonksiyonu bir minimizasyon probleminin ele alındığını gösterirken diğer fonksiyonlar genellikle maksimizasyon problemini ifade ederler.

Basit bir örnek ele alırsak, $y(x) = x^2 - 2$ fonksiyonunun $x \in [-2,2]$ aralığındaki minimum değerini aradığımızı düşünelim. Fonksiyon grafiğinden de kolaylıkla anlaşılabildiği gibi fonksiyon $x = 0$ değerini aldığında minimum değere ulaşmaktadır.

Bu fonksiyonun minimumunu GA ile aramak istediğimizi düşünelim. Bu durumda amaç fonksiyonu R dili ile yazıldığında aşağıdaki gibi gözükecektir.

```
myFitnessFunction <- function (ch) {
myX <- binary2decimal(ch)
y <- myX^2 - 2
return(y)
}
```

Kod bloğu incelendiğinde bir fonksiyon tanımlaması yapıldığı görülmektedir. Fonksiyon ch isimli bir değişken almaktadır. GA, bu değişkenin değerini kromozomun kendisi olarak ayarlayacaktır. R'nin GA paketi içerisinde bulunan binary2decimal fonksiyonu ikili tabandaki bir kromozomu



Şekil 7.2: Rastrigin Fonksiyonuna Ait Grafik

10 tabanında bir değere dönüştürmeyi sağlamaktadır. Kod bloğunda fonksiyon içerisine ulaşan kromozom 10 tabanına dönüştürülerek fenotip elde edilir. Ardından bu değer *myX* değişkenine yüklenir ve çözülmek istenilen *y* fonksiyonu içerisindeki karar değişkeni yerine koyulur. Bu sayede $y(myX)$ elde edilir. Fonksiyon elde edilen değeri geri döndürür. Bu sayede eldeki amaç fonksiyon, iletilen kromozoma karşılık uygunluk değerini döndürmüş olmaktadır.

Ele alınan problem her zaman tek bir durumun optimize edilmesinden oluşmuyor olabilir. Örnekle açıklamak gerekirse bir araba satın alma tercihinde fiyat ve konforu baz aldığımızı düşünelim. Amacımız düşük fiyata yüksek konfor elde etmek olacaktır. Ancak bu iki amacı ayrı ayrı ele almak bizi gerçek bir çözüme ulaştırmayacaktır. İki durumu da dikkate aldığımızda ise en iyi çözüm daima ütopik kalacaktır. Buna amaçların çatışması (objective conflict) adı verilir. Bu tarz problemlere ise çok amaçlı problemler denilmektedir. GA kullanılarak çok amaçlı problemlerin çözülebilmesini sağlayacak algoritmalar önerilmiştir. Önde gelenleri şu şekilde sıralanabilir [11,12]:

- VEGA (Vector Evaluated Genetic Algorithm)
- Ağırlıklandırarak Toplama Yöntemi (Weighted Sum Method)
- MOGA (Multiobjective Genetic Algorithm)
- NPGA (Niched-Pareto Genetic Algorithm)
- NSGA (Non-Dominated Sorting Genetic Algorithm)
- SPEA (Strengh-Pareto Evolutionary Algorithm)
- NPGA-2
- NSGA-II

Bu kitap bölümünde klasik GA'nın çalışma yapısından bahsedildiği için uygulama başlığı altında tek amaç fonksiyonlu problemler ele alınacaktır.

7.6 Uygulamalar

Bu başlık, GA'nın çalışma mantığını ve kullanımını örnek üzerinde göstermek için düzenlenmiştir. Ele alınan optimizasyon problemleri, öncesinde sonuçları bilinen, GA kullanılarak en iyi çözüm arandığında elde edilecek sonucun test edilebilmesi için kullanılan problemlerdir.

Tüm uygulamalarda R dili ve GA paketi kullanılmıştır. R, istatistiksel hesaplama ve görselleştirme için kullanılan bir özgür yazılım geliştirme ortamıdır [3]. GA paketi, GA aramaları gerçekleştirilebil-

mesini sağlayan bir R paketidir [13,14]. R'yi kullanabilmek için r-base ve RStudio yazılımlarını bilgisayarınıza kurmanız önerilir. Ardından GA paketini çalıştırabilmek için R konsolunda aşağıdaki komutu çalıştırmanız gerekmektedir:

```
install.packages("GA")
```

Bu komut GA paketini bilgisayarınıza indirir ve kullanıma hazır hale getirir. Paket için herhangi bir güncelleme yayınlanmadığı sürece bu komutu aynı bilgisayarda tekrar çalıştırmanıza gerek kalmayacaktır.

7.6.1 Basit Bir Uygulama

Bu uygulama R diliyle GA aramasının nasıl yapılabileceğini göstermek için hazırlanmış temel düzeyde basit bir uygulamadır. Optimize edilmek üzere ele alınan fonksiyon;

$$z = f(x,y) = -(x^2 + y^2) + 4$$

olarak seçilmiştir. $f(x,y)$ fonksiyonu 2 karar değişkeni alan bir fonksiyondur. Fonksiyon incelendiğinde $x = 0, y = 0$ değerlerinde maksimum değeri aldığı kolayca görülmektedir. Aynı sonucun GA tarafından da bulunması beklenmektedir.

R kodu içerisinde ilk önce daima kullanılacak paketlerin çağrılması ve ilgili kütüphanelerin aktifleştirilmesi gerekmektedir. Bilgisayara yüklü GA paketinin kullanılabilir hale gelmesi için aşağıdaki komutla çağırma işlemi gerçekleştirilmelidir.

```
require("GA")
```

GA araması başlatılmadan önce arama yapılacak problemi GA'ya tanıtılabilmesi için fonksiyon olarak tanımlanması gerekmektedir. Fonksiyon tanımlaması aşağıdaki şekilde gerçekleştirilebilir.

```
myFitnessFunction <- function(ch) {
x <- binary2decimal(ch[1:10])
y <- binary2decimal(ch[11:20])

return(-x^2 - y^2 + 4)
}
```

Burada dikkat edilmesi gereken nokta, arama yapılan fonksiyon 2 karar değişkenine sahip olmasına rağmen amaç fonksiyonun tek bir girdi almasıdır. Eldeki problemin kaç karar değişkenine sahip olduğundan bağımsız olarak klasik GA aday çözümü her zaman tek bir kromozom olarak ifade edecektir. Genotipi fenotipe dönüştürmek uygulayıcının görevidir. Yukarıdaki kod bloğunda görüldüğü gibi kromozomun ilk 10 bitlik bölümü 10 tabanında bir sayıya dönüştürülerek x değişkenine, ikinci 10 bitlik bölümü ise yine 10 tabanına dönüştürülerek y değişkenine atanmıştır. Sonrasında aday çözümden gelen x ve y değerleri dikkate alınarak uygunluk değeri hesaplanmış ve bu değer döndürülmüştür. Fonksiyon, 20 bitlik bir kromozoma karşılık eldeki fonksiyonun uygunluk değerini döndürmektedir.

Amaç fonksiyonu tanımlandıktan sonra GA paketinde bulunan ga fonksiyonu çağrılarak arama gerçekleştirilebilir. Aramayı gerçekleştirmek üzere örnek kod bloğu aşağıdaki gibidir.

```
myga <- ga(
type = "binary",
fitness = myFitnessFunction,
nBits = 20,
maxiter = 100,
maxFitness = 4,
popSize = 30
)
```

ga fonksiyonu içerisinde, GA aramasında uygulanması istenilen parametreler tanımlanabilmektedir. type niteliği kromozom tipini, fitness niteliği kromozomların değerlendirileceği amaç fonksiyonunu, nBits niteliği kromozom uzunluğunu, maxiter arama yapılacak en fazla iterasyon (nesil) sayısını, maxFitness hedeflenen optimum çözümü, popSize topluluk büyüklüğünü tanımlayabilmemize olanak sağlamaktadır. Arama işlemi maxFitness değerine ulaşana kadar ya da maxiter ile tanımlanan iterasyon sayısı tamamlanana kadar devam eder. Hangi koşul önce sağlanırsa arama sona erer. Kod bloğu çalıştırıldığında karşılaşılan ekran görüntüsü aşağıdaki gibidir.

```
GA | iter = 35
Mean = -133.5333 | Best =    4.0000
```

Şekil 7.3: GA Arama Sonucunun Ekran Çıktısı

Gerçekleştirilen örnekte GA araması 35. nesilde hedeflenen en iyi sonuca ulaştığı için arama otomatik olarak sonlandırılmıştır. GA'nın ulaştığı en iyi sonucu elde etmek için aşağıdaki komut kullanılabilir:

```
bestch <- myga@solution
```

Bu kod, en iyi sonucu üreten kromozomu bestch değişkenine tanımlamaktadır. bestch değişkeni ekrana yazdırıldığında aşağıdaki ekran görüntüsü elde edilecektir:

```
> print(bestch)
     x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20
[1,]  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0
```

Şekil 7.4: En İyi Kromozomun Ekrana Bastırılması

Kromozomun fenotip dönüşümü yapıldıktan sonra ekrana bastırılması için ise aşağıdaki kod bloğu kullanılabilir:

```
print(binary2decimal(myga@solution[1:10]))
print(binary2decimal(myga@solution[11:20]))
```

Kod bloğu çalıştırıldığında karşılaşılacak ekran çıktısı aşağıdaki gibi olacaktır:

```
> print(binary2decimal(myga@solution[1:10]))
[1] 0
> print(binary2decimal(myga@solution[11:20]))
[1] 0
```

Şekil 7.5: Elde Edilen Çözümün Ekrana Bastırılması

Şekilden de görüleceği üzere GA araması sonucunda elde edilen çözüm $x = 0, y = 0$'dır. Çalıştırılabilir kodun tamamı aşağıdaki gibidir:

```
require("GA")

myFitnessFunction <- function (ch) {
x <- binary2decimal(ch[1:10])
y <- binary2decimal(ch[11:20])

return (-x^2 - y^2 + 4)
}

myga <- ga(
type = "binary",
fitness = myFitnessFunction,
nBits = 20,
maxiter = 100,
maxFitness = 4,
popSize = 30
)

bestch <- myga@solution

print(binary2decimal(myga@solution[1:10]))
print(binary2decimal(myga@solution[11:20]))
```

7.6.2 İkili Kodlama ile Optimizasyon

$$\pi = 3,14159265358979323846264338327 9...$$

Dünyaca ünlü irrasyonel sayı pi (π), bir çemberin çevresinin çapına oranına eşittir. Tüm irrasyonel sayılar gibi π de virgülden sonra düzensiz ve sonsuz sayıda rakam barındırmaktadır. Yaklaşık değeri $3,14$ olarak kabul edilen π sayısını tam olarak belirleyebilmek için kullanılan çeşitli yaklaşımlar ve teoriler mevcuttur. Bu çalışmalar π sayısının tam değerini en hassas şekilde belirleyebilmek için gerçekleştirilmektedir. Ancak pratikte π'nin yaklaşık değeri çoğu hesaplama için yeterli gelmektedir.

$3,14$'ten sonra en sık kullanılan π yaklaşımı $22/7$'dir. Bu sayının tam değeri ve gerçek π sayısından yaklaşık uzaklığı aşağıdaki gibidir.

$$\frac{22}{7} = 3,142857143$$

$$|\pi - \frac{22}{7}| \cong 0,001264489$$

Ayrıca en sık kullanılan π yaklaşımı olan $3,14$'ün de gerçek π sayısından uzaklığına bakmak istersek elde edeceğimiz değer aşağıdaki gibi olacaktır.

$$|\pi - 3,14| \cong 0,001592654$$

Bu örnekte, π sayısının verilen 2 yaklaşımından daha yakın bir yaklaşım aranacaktır. Bu aramada, oranlandıklarında yaklaşık π sayısını verecek 2 tane 3 basamaklı tamsayı aranmaktadır.

GA paketini kullanabilmek için hazır hale getiriyoruz:

```
require("GA")
```

Çözmek istediğimiz problemi amaç fonksiyonuna dönüştürmemiz gerekiyor. 2 sayı aradığımız için kromozomu iki parçaya (gene) bölmemiz gerekecek. Her bir gen parçası 10 bitten oluşacak, bu sayede her bir genin alabileceği değer aralığı $[0, 1023]$ olarak belirlenebilecektir.

```
myFitnessFunction <- function (ch) {
x <- binary2decimal(ch[1:10])
y <- binary2decimal(ch[11:20])

return (-abs(3.14159265359 - (x / y)))
}
```

Amaç fonksiyonu içerisinde kromozom 2 gene ayrılarak 10 tabanına dönüştürüp x ve y değişkenlerine yüklenmektedir. Sonrasında elde edilen x/y oranının gerçek π sayısından uzaklığı belirlenmektedir. Amacımız bu uzaklığın minimum olması, bu sayede gerçek π'ye en yakın oranın belirlenmesidir. R'nin GA paketi varsayılan olarak maksimizasyon yaptığı için elde edilen fark mutlak değerle pozitif hale getirildikten sonra -1 ile çarpılarak negatife dönüştürülür. Böylece GA maksimizasyon yapmaya devam ederken aslında bizim aramamız minimizasyon yapmaya devam eder.

Amaç fonksiyonu tanımlandıktan sonra GA aramasını başlatacak kod bloğunu yazabiliriz:

```
myga <- ga(
type = "binary",
fitness = myFitnessFunction,
nBits = 20,
maxiter = 10000,
maxFitness = 0,
popSize = 100
)
```

Kromozom kodlamamız ikili, kromozomların değerlendirilmesinde kullanılacak amaç fonksiyonumuz myFitnessFunction, kromozom uzunluğu 20 bit, en fazla nesil sayısı 10.000, ulaşmak istediğimiz değer (π'ye uzaklık) 0, topluluk büyüklüğü 1000 olarak seçilmiştir. Bu kod bloğu çalıştırıldığında aşağıdaki sonucu almamız mümkündür.

```
GA | iter = 10000
Mean = -1.819426e-01 | Best = -2.667640e-07
```

Şekil 7.6: Elde Edilen Çözümün Ekrana Bastırılması

Arama sonucunda ulaşılan sonucu elde etmek için şu kod bloğunu çalıştırabiliriz:

```
bestch <- myga@solution

x <- binary2decimal(myga@solution[1:10])
y <- binary2decimal(myga@solution[11:20])

print(x)
print(y)

print(x / y)
```

Bu aramada elde edilen çözüm aşağıdaki gibidir.

$$x = 710, y = 226, \frac{x}{y} = 3,14159292$$

Elde edilen yeni yaklaşımın gerçek π sayısından uzaklığı aşağıdaki gibidir.

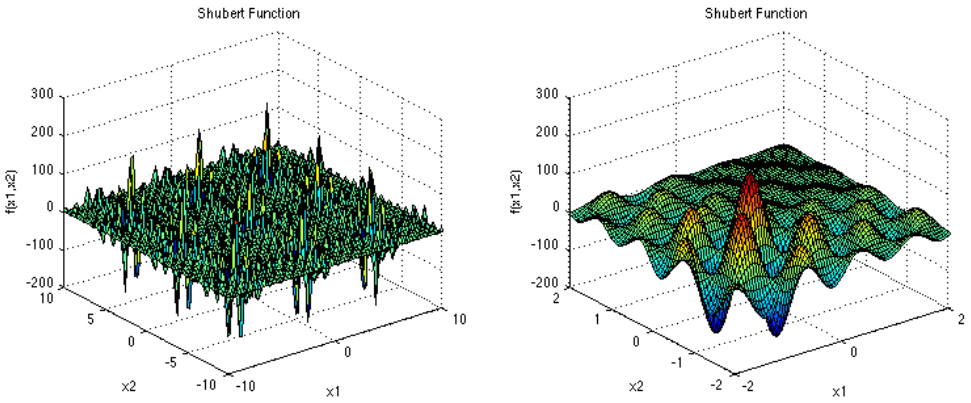
$$|\pi - \frac{710}{226}| \cong 0,0000002668 = 2,668 \times 10^{-7}$$

Böylece en sık kullanılan iki yaklaşımdan çok daha başarılı bir yaklaşım elde etmiş olduk.

7.6.3 Gerçek Sayılı Kodlama ile Optimizasyon

Optimizasyon problemlerinin test edilmesinde kullanılan test fonksiyonları mevcuttur. Bu fonksiyonların hangi değerlerde minimum ya da maksimum oldukları önceden bellidir. Hatta fonksiyonun zorluğunu daha kolay anlamak adına grafikleri de çizilmiştir. Böylece optimizasyon yöntemi ele alınan test fonksiyonunun optimizasyonunda kullanılır ve başarısı test edilmiş olur.

Bu örnekte bir test fonksiyonu olan Shubert kullanılacaktır. Fonksiyonun 2 karar değişkeni aldığı ve optimum değere ulaştığında $f(x_i^*) = -186,7309$ sonucunu ürettiği bilinmektedir. Fonksiyonun tanımı ve grafiği aşağıdaki gibidir.



Şekil 7.7: Shubert Fonksiyon Tanımı

$$f(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$$

Şekil 7.8: Shubert Fonksiyon Grafiği

Fonksiyon için minimum araması gerçekleştirmeden önce GA paketini kullanılabilir hale getirmemiz gerekmektedir.

```
require("GA")
```

Bu problem, karar değişkenlerine gerçek sayılar kümesinden değerler atanmasını gerektirdiği için klasik GA kromozom yapısı yerine Gerçek Sayılı Genetik Algoritmalar tercih edilecektir. Böylece GA, talep edilen aralık ve sayıda gerçek sayıyı amaç fonksiyonuna gönderecektir. Bu duruma uygun amaç fonksiyonu aşağıdaki gibi tanımlanabilir.

```
myFitnessFunction <- function (ch) {
x1 <- ch[1]
x2 <- ch[2]
ii <- c(1:5)

sum1 <- sum(ii * cos((ii+1)*x1+ii))
sum2 <- sum(ii * cos((ii+1)*x2+ii))

y <- sum1 * sum2
return(-y)
}
```

Kod bloğunda görüldüğü gibi ch değişkeni 2 değer almakta, bu iki değer x1 ve x2 değişkenlerine yüklenmekte ve fonksiyon sonucu üretilmektedir. Sonuç, minimizasyon problemine dönüştürülebilmesi için -1 ile çarpılmıştır.

Amaç fonksiyonu tanımlandıktan sonra GA araması aşağıdaki kod bloğu ile başlatılabilir.

```
myga <- ga(
type = "real-valued",
fitness = myFitnessFunction,
min = c(rep(-10, 2)),
max = c(rep(10, 2)),
popSize = 100,
maxiter = 5000
)
```

Kromozom tipi "real-valued", amaç fonksiyonu myFitnessFunction, çözüm yapılacak arama uzayı $x_i = [-10, 10]$, topluluk büyüklüğü 100, iterasyon sayısı 5000 olarak seçilmiştir. Arama sonucunun ekran görüntüsü aşağıdaki gibidir.

Ulaşılan en iyi çözümün, beklenen çözüme eşit olduğu görülmektedir. Bu çözümü üreten x_1 ve x_2 değerlerini aşağıdaki komut ile öğrenebiliriz.

```
print(myga@solution)
```

```
GA | iter = 5000
Mean = 144.8985 | Best = 186.7309
```

Şekil 7.9: GA arama sonucu

Komutu çalıştırdığımızda;

$$x_1 = -1,425128, x_2 = 5,482864$$

değerlerine ulaşıldığını görebiliriz. Böylece ele aldığımız fonksiyonu minimum eden karar değişkenlerine başarıyla erişmiş olmaktayız.

7.6.4 Permütasyon Kodlama ile Optimizasyon

Çok sayıda kromozom kodlama tipi olduğundan bahsedilmişti. İkili kodlama ve gerçek sayılı kodlama ile ilgili örnekler sunuldu. GA ile sık kullanılan kromozom tiplerinden biri de permütasyon kodlamasıdır. Bu kodlama 1'den istenilen sayıya kadar olan bir sıranın nasıl olması gerektiğini belirler. Her bir kromozom ile eldeki sayılar yeni bir sıra olarak amaç fonksiyonuna gönderilir ve değerlendirilir.

Bu örnekte permütasyon kodlamayla uygulanabilecek en ünlü problemlerden biri olan Gezgin Satıcı Problemi (Traveling Salesman Problem) ele alınacaktır. Problem basitçe, önceden tanımlı n adet noktanın tamamının en kısa yol izlenerek ziyaret edilebilmesi problemidir. Amaç, GA'nın böyle bir problem çözümünde nasıl kullanıldığını göstermek olduğu için belirlenen az sayıda nokta ile bir arama gerçekleştirilecektir. Aramada kullanılacak noktalar tablodaki gibidir.

Tablo 7.3: Ele Alınan Noktalar

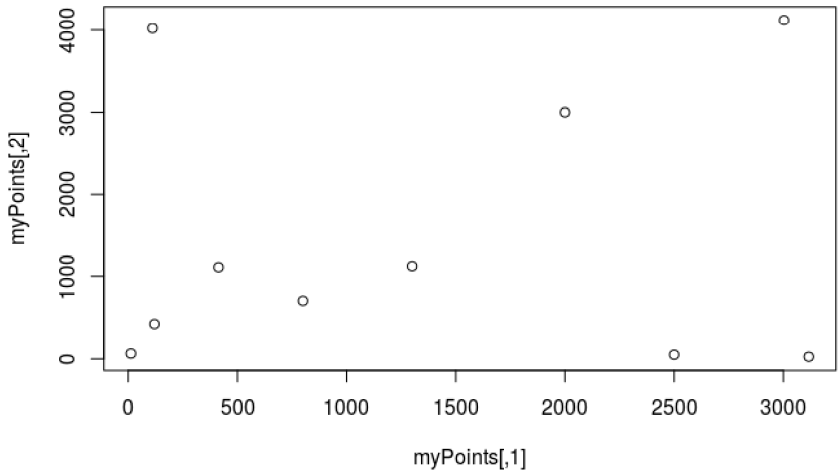
x	**y**
12	65
3003	4123
800	704
3116	26
2500	51
1300	1123
2000	3000
413	1110
120	420
111	4021

Noktaların grafik üzerinde gösterimi aşağıdaki gibidir.

Artık bu noktaları en kısa yolu katederek nasıl dolaşabileceğimiz probleminin çözümünü arayabiliriz. Tüm uygulamalarda olduğu gibi öncelikle GA paketini kullanıma hazır hale getirmeliyiz:

```
require( "GA" )
```

Bu problemde kullanacağımız iki yardımcı fonksiyon olacak. euc.dist fonksiyonu iki koordinat arasındaki uzaklığı döndüren fonksiyondur. showTSPResult fonksiyonu ise elde ettiğimiz çözümü grafik üzerinde bize sunacak olan fonksiyondur. Bu fonksiyonların tanımı aşağıdaki gibidir.



Şekil 7.10: Noktaların Dağılımı

```
euc.dist <- function(x1, x2) {
sqrt(sum((x1 - x2) ^ 2))
}

showTSPResult <- function (ch) {
plot(myPoints)

for (i in 2:length(ch)) {
segments(
myPoints[ch[(i-1)],1],
myPoints[ch[(i-1)],2],
myPoints[ch[i],1],
myPoints[ch[i],2]
)
}

segments(
myPoints[ch[1],1],
myPoints[ch[1],2],
myPoints[ch[length(ch)],1],
myPoints[ch[length(ch)],2]
)
}
```

GA aramasına başlamadan önce belirlediğimiz noktaları matris formatında R dilinde yazmalıyız:

```
myPoints <- matrix(c(12, 65,
3003, 4123,
800, 704,
3116, 26,
2500, 51,
1300, 1123,
2000, 3000,
413, 1110,
120, 420,
111, 4021), ncol = 2, byrow = TRUE)
```

Arama işleminden önce son olarak amaç fonksiyonumuzu da kodlamamız gerekmektedir. Amaç fonksiyonu, GA'nın gönderdiği aday çözümdeki sıralamayı takip etmemiz durumunda ne kadar mesafe katedeceğimizi hesaplayacak ve problemimiz minimizasyon problemi olduğu için -1 ile çarparak sonucu döndürecektir.

```
myFitnessFunction <- function (ch) {
totaldistance <- 0

for (i in 2:length((ch))) {
totaldistance <- totaldistance +
euc.dist(
myPoints[(ch[(i-1)]),],
myPoints[ch[i],]
)
}

totaldistance <- totaldistance +
euc.dist(
myPoints[ch[length(ch)],],
myPoints[1,]
)

return(-totaldistance)
}
```

Bu tanımlamalardan sonra GA aramasını başlatabiliriz:

```
myga <- ga(
type = "permutation",
fitness = myFitnessFunction,
popSize = 100,
maxiter = 1000,
min = 1,
max = 10
)
```

Kromozom kodlama tipi permutation, amaç fonksiyonu myFitnessFunction, topluluk büyüklüğü 100, iterasyon sayısı 1000, sıralama başlangıç 1, bitiş 10 olarak seçilmiştir. Kod bloğunu çalıştırdığımızda alınacak ekran görüntüsü aşağıdaki gibi olacaktır.

```
GA | iter = 1000
Mean = -12561.53 | Best = -12135.54
```

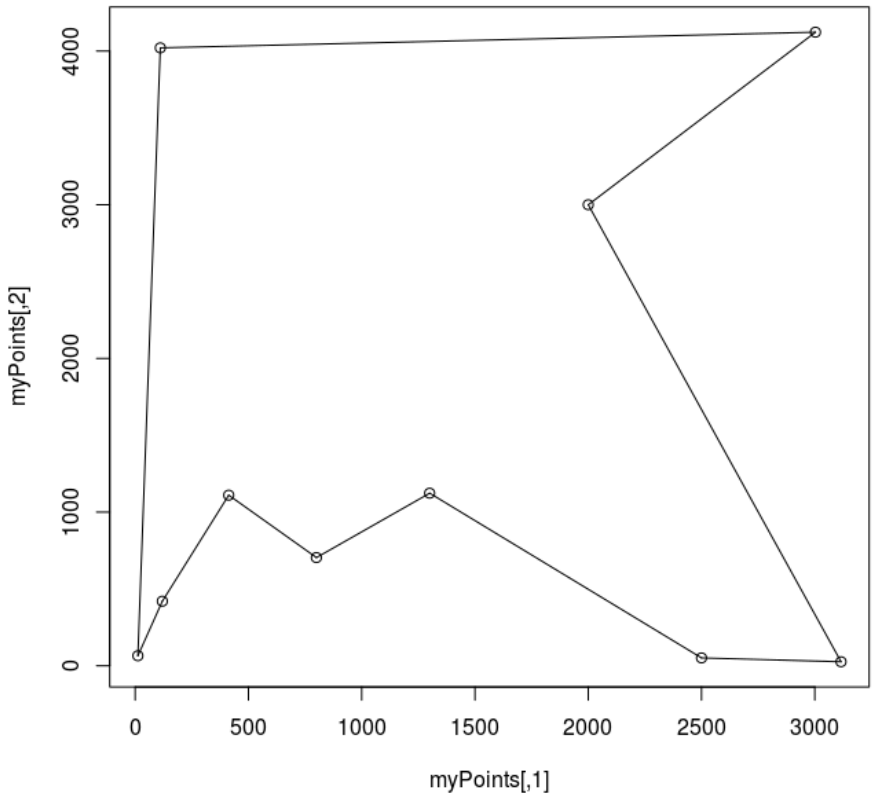
Şekil 7.11: GA Araması Sonucu

Elde edilen en iyi çözüme ulaşmak için kullanılacak komut şudur:

```
print(myga@solution)
```

En iyi çözümün grafiğini elde etmek için daha önce tanımladığımız showTSPResult fonksiyonunu kullanabiliriz. Komut ve ekran çıktısı aşağıdaki gibi olacaktır.

```
showTSPResult(myga@solution)
```



Şekil 7.12: Çözümün Görselleştirilmesi

Grafik incelendiğinde elde ettiğimiz çözümün anlamlı bir çözüm olduğu söylenebilir.

7.7 Kaynakca

[1] J. H. Holland, “Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence,” Ann Arbor, MI Univ. Michigan Press, 1975.

[2] M. Mitchell, “Genetic algorithms: An overview,” Complexity, vol. 1, no. 1, pp. 31–39, 1995.

[3] R Core Team, “R: A Language and Environment for Statistical Computing.” Vienna, Austria, 2017.

[4] L. Scrucca, “On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution,” Submitt. to R J., 2016.

[5] M. H. Satman, “Machine coded genetic algorithms for real parameter optimization problems,” Gazi Univ. J. Sci., vol. 26, no. 1, pp. 85–95, 2013.

[6] S. Bekiroglu, T. Dede, and Y. Ayvaz, “Implementation of different encoding types on structural optimization based on adaptive genetic algorithm,” Finite Elem. Anal. Des., vol. 45, no. 11, pp. 826–835, 2009.

[7] E. Akadal, “Ham verilerin genetik algoritmalarla ilişkisel veritabanlarına dönüştürülmesi ve bir uygulama.”.

[8] M. Srinivas and L. M. Patnaik, “Adaptive probabilities of crossover and mutation in genetic algorithms,” IEEE Trans. Syst. Man. Cybern., vol. 24, no. 4, pp. 656–667, 1994.

[9] M. Mitchell, An introduction to genetic algorithms. MIT press, 1998.

[10] R. L. Haupt and S. E. Haupt, Practical genetic algorithms. John Wiley& Sons, 2004.

[11] R. T. Marler and J. S. Arora, “Survey of multi-objective optimization methods for engineering,” Struct. Multidiscip. Optim., vol. 26, no. 6, pp. 369–395, 2004.

[12] H. A. Taboada and D. W. Coit, “Multi-objective scheduling problems: Determination of pruned Pareto sets,” IIE Trans., vol. 40, no. 5, pp. 552–564, 2008.

[13] L. Scrucca, “GA: A Package for Genetic Algorithms in R,” J. Stat. Softw., vol. 53, no. 4, pp. 1–37, 2013.

[14] L. Scrucca, “On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution,” Submitt. to R J., 2016.

8. Dengesiz Veri Setlerinde Sınıflandırma

Dengesiz Veri Setlerinde Sınıflandırma

Araş. Gör. Dr. Elif KARTAL, Araş. Gör. Dr. Zeki ÖZEN

8.1 Giriş

Makine öğrenmesi, yapay zekânın çalışma alanlarından biri olup, temelinde makineleri kendi kendine karar verebilir hale getirme fikri yatmaktadır. Makinelerin öğrenme sürecinde iki temel yöntemden faydalanılmaktadır: Danışmanlı öğrenme (supervised learning) ve danışmansız öğrenme (unsupervised learning). Danışmanlı öğrenme ile danışmansız öğrenme arasındaki en temel fark; danışmanlı öğrenmede bir örneğin ait olduğu sınıf değerinin algoritmanın eğitimi ve testi aşamalarında verilmesi; ancak danışmansız öğrenmede örneğin ait olduğu sınıf / grup / küme değerinin algoritmaya verilmemesi, algoritmanın örneğe ait nitelikleri kullanarak kendi kendine gruplandırmasıdır. Hastalık teşhisi, risk değerlendirme, bir bankada gerçekleştirilen işlemler arasından dolandırıcılık tespiti (fraud detection) gibi sınıflandırma problemleri danışmanlı öğrenme ile ilişkilendirilirken; danışmansız öğrenmeden kümeleme / gruplandırma problemlerinin çözümünde faydalanılmaktadır.

Danışmanlı öğrenmeye dayalı makine öğrenmesi tekniklerini kullanarak bir tahmin / sınıflandırma / öngörü gerçekleştirilmesi söz konusu olduğunda; kullanılan algoritma, (varsa) algoritmanın parametreleri, performans değerlendirme yöntemi kadar veri setinin yapısı da son derece önemlidir. Özellikle üzerinde çalışılan veri setinde tahmin edilmek istenen hedef niteliğin sınıf değerleri arasındaki dağılımın her zaman %50 - %50 biçiminde olmayabileceği göz önünde bulundurulmalıdır. Durumun daha anlaşılır olması için toplumda nadir görülen bir hastalık üzerine yapılan bir araştırma ele alınsın. Araştırmada veri seti oluşturulurken, hem hastalık tanısı bulunmayan hem de hastalık tanısı konmuş olan örneklerin toplanması amaçlanmaktadır. Ancak hastalığın nadir görülmesi sebebi ile 100 vakadan sadece üçünde hastalığa rastlanmaktadır. Bu nedenle de veri setinde hastalık durum bilgisini içeren hedef niteliğin sınıf değerleri (hastalık var / hastalık yok) sırasıyla %3 ile %97 biçiminde dağılmaktadır. Hedef niteliğin sınıf etiketlerinin frekansı arasında verilen örnekteki gibi dengesiz bir dağılım söz konusu olduğunda literatürde dengesiz veri (imbalanced data) olarak bilinen durum ortaya çıkmaktadır.

Dengesiz veri setleri ile çalışırken analiz sonuçlarında mükemmele yakın elde edilen doğruluk değeri modelin çok başarılı olduğu anlamına gelmemektedir. Örneğin; az önce örneklenen veri setini kullanan bir öğrenme algoritması hasta olmayan 97 kişinin tamamını doğru, hasta olan 3 kişinin tamamını ise yanlış tahmin etmiş olsun. Bu durumda algoritma %97 doğrulukla çalışmaktadır. Sadece doğruluk değerine bakıldığında algoritma performansının çok iyi olduğu düşünülebilir. Fakat bu durumda algoritmanın azınlıkta olan sınıfa ait (%3) örneklerin tamamını yanlış sınıflandırmış olması göz ardı edilmiş olacaktır; çünkü algoritmanın belirleyicilik değeri %100 iken duyarlılık değeri ise %0'dır. Bu nedenle, performans değerlendirmesi yapılırken doğruluk değerinin yanında duyarlılık, belirleyicilik, negatif / pozitif öngörü değeri gibi performans değerlendirme ölçülerinin de dikkate alınması daha sağlıklı performans değerlendirmesi yapılmasını sağlayacaktır. Dengesiz veri setlerinde karşılaşılan performans değerlerindeki bu uçurumun iyileştirilmesi amacıyla undersampling, oversampling ve SMOTE (Synthetic Minority Over-sampling Technique) gibi veri setindeki örneklem yapısını değiştiren yöntemler kullanılmaktadır.

Kitabın bu bölümünde yazarlar; dengesiz veri seti kullanılarak oluşturulan yapay sinir ağı modelinin performans değeri ile undersampling, oversampling ve SMOTE yöntemlerinin kullanıldığı modellerin performans değerlerini karşılaştırmış ve yorumlamıştır. Bu kapsamda UCI Machine Learning Repository'de ücretsiz kullanıma açık olan veri setlerinden Pima Indians Diyabet Veri seti kullanılmıştır.

8.2 Makine Öğrenmesi

Makine öğrenmesi fikrinin temelinde, bilgisayarların sadece kendisine verilen görevleri yapan ve komutları işleyen bir cihaz olmasından ziyade insan gibi öğrenebilen sistemler olması yatmaktadır. İnsanlarda öğrenme süreci hayatları boyunca gerek ebeveynleri, gerek öğretmenleri, gerekse yaşadıkları sosyal çevrede başkalarıyla etkileşime girmesiyle gerçekleşir. Peki, makinelerde öğrenme nasıl olmaktadır?

Makine öğrenmesi için yapılan literatürdeki tanımlar ele alınacak olursa içlerinde konuya Mitchell (1997)'in öğrenme tanımı ile başlamak doğru olacaktır. Mitchell (1997) "Eğer bir bilgisayar programının G görevlerinde P ile ölçülen performansı deneyim D ile artıyorsa, o bilgisayar programının bazı G görevlerinin sınıflarına ve performans ölçüsü P'ye göre deneyim D'den öğrendiği söylenmektedir" demiştir (Balaban ve Kartal, 2015). Tanım, bir tekerleme gibi ilk okunduğunda kavraması zor gibi gözükse de, tanımda yer alan dört ana unsur (Şekil 1) açıklandığında tanımın ne kadar sade olduğu ortaya çıkacaktır (Balaban ve Kartal, 2015):

1. **Görev:** İnsanlar yaptıkları eylemleri birer amaç doğrultusunda yerine getirirler. İnsanlardan bilgisayarlara geçecek olursak, klasik programlamada sistem analizi ve tasarımı evresi, problemin tanımlanması yani bilgisayarın ne iş yapacağının belirlenmesiyle başlamaktadır. Benzer yaklaşımla bilgisayarları öğrenir hale getirmek için onlara neyi öğreneceğinin ya da görevinin ne olduğunun söylenmesi gerekir. Bilgisayara araştırma problemi olarak maç sonuçlarının tahmini, hava durumu tahmini, kaza öngörüsü gibi pek çok görev verilebilir.
2. **Deneyim:** İnsanın öğrenme kaynakları çok çeşitli olmakla birlikte ağırlıklı olarak okuduklarından, gördüklerinden ve tecrübelerden öğrenmektedir. Bilgisayarlar için deneyim, bilgisayarlara tanımlanan görevi çözmesi için sunulan veridir. Eğer hastalık teşhisi gerçekleştirilecekse; hasta olan ve hasta olmayan kişilere ait hastalığı tanımaya yardımcı olacak kan basıncı, yaş, cinsiyet vb. veri algoritmaya verilir. Algoritmalar da bu veriyi tecrübe olarak kabul eder ve öğrenme / eğitim için kullanırlar.
3. **Algoritma:** Algoritma; kısaca bir problemin çözümü için adım adım izlenen yol anlamına

Şekil 8.1: Makine öğrenmesindeki temel unsurlar.

gelmektedir. Makine öğrenmesi alanında sınıflandırma ve kümeleme ana başlıkları altında klasik makine öğrenmesi algoritmaları, yapay sinir ağları öğrenme algoritmaları, genetik algoritmalar, karar ağaçları, destek vektör makineleri gibi pek çok algoritma geliştirilmiştir.

4. **Performans:** Problemin çözümünü ararken hedefe uygun olarak hangi algoritmanın kullanılacağı, hangisi ya da hangilerinin en iyi performansı vereceği büyük önem teşkil etmektedir; çünkü çok sayıda unsur performansa etki etmektedir. Kullanılan modelin ve algoritmanın parametrelerinin optimize edilmiş olması, veri setinin eğitim ve test veri seti amacıyla bölünme yöntemi ve veri setinin tabiatı bu unsurlar arasında sayılabilir. Algoritmalar genellikle parametreli kullanım yapısındadır. Örneğin k-en yakın komşu algoritmasında k değerine atanan sayı ile algoritmanın performansı değişmektedir. Benzer şekilde kitabın bu bölümündeki uygulamada kullanılan yapay sinir ağındaki katman ve nöron sayısı kurulan modelle ilgili parametreler iken, yapay sinir ağının öğrenmesi için kullanılan Geri Yayılım algoritmasında da öğrenme oranı, momentum oranı, hata oranı gibi parametreler bulunmaktadır. Algoritma parametrelerinin optimum değerlerde olması algoritmanın daha iyi öğrenmesini ve dolayısıyla daha iyi performans göstermesini sağlayacaktır.

Kurulan modelin ve kullanılan algoritmanın bir görevi ne kadar iyi öğrenip öğrenmediğini görebilmek için algoritmanın performansının ölçülmesi gereklidir. Farklı makine öğrenmesi tekniklerinin ya da farklı parametreler ve aynı makine öğrenmesi tekniği kullanılarak oluşturulan modellerin performansını kıyaslamak için çeşitli performans değerlendirme yöntemleri ve ölçüleri geliştirilmiştir. Şekil 2'de en sık kullanılan performans değerlendirme ölçüleri verilmiştir (wikipedia, 2017; Balaban ve Kartal, 2015).

Kurulan modellerin ve kullanılan algoritmaların performans kıyaslaması için genellikle doğruluk (accuracy) değeri kullanılmaktadır. Doğruluk değeri, algoritmanın doğru tahmin ettiği örnek sayısının veri setindeki toplam örnek sayısına bölünmesiyle hesaplanmaktadır.

$$Doğruluk = \frac{DP + DN}{DP + DY + YN + DN}$$

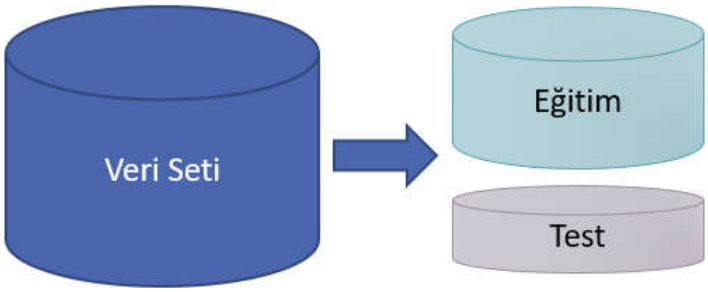
Öte yandan doğruluk, performans değerlendirilmesinde kullanılacak tek ölçü olmamalıdır. Özellikle kitabın bu bölümünde ele alınan model performans değerlendirmesine benzer biçimde; du-

		Gerçek Durum				
	Tüm popülasyon (N)	Pozitif olma durumu (POD)	Negatif olma durumu (NOD)	***Görülme Oranı*** $= \frac{\sum POD}{\sum N}$	***Doğruluk*** $= \frac{\sum DP + DN}{\sum N}$	
Tahmini Durum	Pozitif Tahminler (PT)	Doğru Pozitif (DP)	Yanlış Pozitif (YP) *(Tip I Hata)*	***Pozitif Öngörü Değeri*** $= \textbf{Kesinlik} = \frac{\sum TP}{\sum PT}$	***Yanlış Bulgu Oranı*** $= \frac{\sum YP}{\sum PT}$	
	Negatif Tahminler (NT)	Yanlış Negatif (YN) *(Tip II Hata)*	Doğru Negatif (DN)	***Yanlış İhmal Oranı*** $= \frac{\sum YN}{\sum NT}$	***Negatif Öngörü Değeri*** $= \frac{\sum DN}{\sum NT}$	
		Doğru Pozitif Oranı $= \textbf{Duyarlılık} = \frac{\sum DP}{\sum POD}$	***Yanlış Pozitif Oranı*** $= \frac{\sum YP}{\sum NOD}$	***Pozitif Olabilirlik Oranı*** $= \frac{Duyarlılık}{Yanlış\ Pozitif\ Oranı}$	***Tanısal Üstünlük Oranı*** $= \frac{Pozitif\ Olabilirlik\ Oranı}{Negatif\ Olabilirlik\ Oranı}$	***F − Ölçüsü*** $= \frac{2}{\frac{1}{Duyarlılık} + \frac{1}{Kesinlik}}$
		Yanlış Negatif Oranı $= \frac{\sum YN}{\sum POD}$	***Doğru Negatif Oranı*** $= \textbf{Belirleyicilik} = \frac{\sum DN}{\sum NOD}$	***Negatif Olabilirlik Oranı*** $= \frac{Yanlış\ Negatif\ Oranı}{Belirleyicilik}$		

Şekil 8.2: Performans değerlendirmesinde kullanılan çeşitli ölçüler.

yarlılık, belirleyicilik, pozitif ve negatif öngörü değeri, F-ölçüsü gibi ölçülere de bakılmalıdır (Kartal, 2015; Özen, 2016).

Kurulan modelin ve kullanılan algoritmanın performansına etki eden bir diğer önemli nokta da veri setinin eğitim ve test veri seti olarak bölünmesi işlemidir. Literatürde hold-out ve k-kat çapraz geçerleme (k-fold cross validation) gibi çeşitli performans değerlendirme yöntemleri bulunmaktadır. Bu bölüm kapsamında yapılan uygulamada veri setini eğitim ve test veri seti olarak ayırmak için hold-out yöntemi tercih edilmiştir (Şekil 3). Hold-out yönteminde veri seti %50-%50, %70-%30, %80-%20 ve benzer oranlarda ikiye ayrılabilir.



Şekil 8.3: Hold-out performans değerlendirme yöntemi.

Veri setini eğitim ve test olarak ayırırken dikkate alınması gereken bazı önemli noktalar şöyle sıralanabilir:

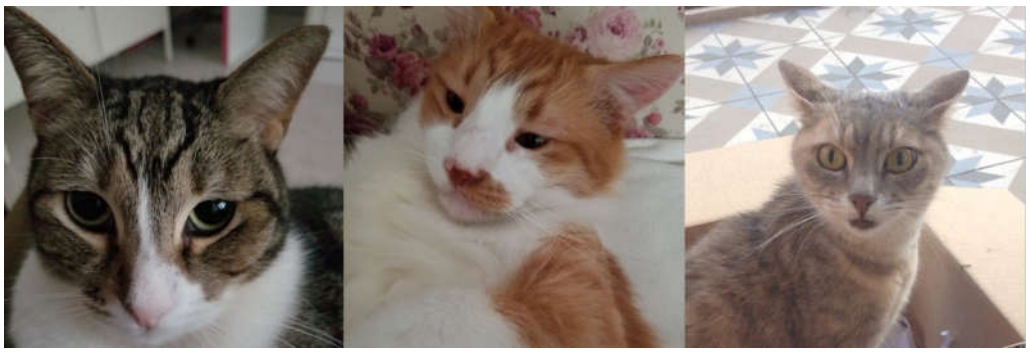
- Eğitim veri setindeki örneklerin sayıca test veri setindeki örneklerden fazla olması beklenir. Makine öğrenmesinde, algoritma ne kadar çok veri ile eğitilirse algoritmanın öğrenme performansı da o kadar artacaktır. Bu nedenle de genellikle algoritmanın eğitimi için kullanılan eğitim veri setindeki örnek sayısı test veri setindeki örnek sayısından fazla tutulur.
- Eğitim ve test veri seti ayrımında örneklerin rastgele dağılması önemlidir. Eğer örnekler elle müdahale edilerek eğitim ve test veri setlerine dağıtılıyorsa, bu durum yapılacak analizin güvenilirliğine gölge düşürecektir.
- Veri setinin eğitim ve test veri seti olarak bölünmesi sırasında hedef niteliğe ait sınıfın eğitim ve test veri setlerindeki dağılımına özen gösterilmelidir. Hem eğitim hem de test veri setinde,

ana veri setindeki her bir sınıfa ait örnekler bulunmalıdır. Örneğin; bir kliniğe ait toplamda 300 örnekten oluşan bir veri setinde hedef nitelik hastalık durumu olsun. Hedef niteliğin de "Var" ve "Yok" şeklinde iki sınıf değeri olsun. Hastalığın görüldüğü örnek sayısı 120, görülmediği örneklerin sayısı 180 olsun. Veri setinin %90'lık kısmının eğitim, %10'luk kısmının test veri seti olacak biçimde ikiye ayrıldığını düşünülsün. Eğer sınıf dağılımları göz ardı edilirse, hastalığın görüldüğü 120 örneğin 120'si de eğitim veri setine gidebilir. Bu durumda test veri setinde hastalığın görüldüğü hiçbir örnek yer almayacaktır. Bu şekilde yapılan performans değerlendirmesi ise sağlıklı olmayacaktır; çünkü algoritmanın genel doğruluk değerinin yanında veri setindeki sınıfları ne oranda doğru tahmin ettiği de performans değerlendirmesi için önemli bir ölçüdür.

Bazen de tam aksi durum söz konusu olabilir. Klinik veri seti örneğinden devam edilecek olursa, eğitim veri setine hastalığın görüldüğü örneklerden hiçbiri denk gelmeyerek hastalık görülen örneklerin tümü test veri setine düşebilir. Algoritmanın eğitim veri setinde hiç görmediği bir sınıfa ait örneklerin test veri setinde bulunması durumunda, algoritma o sınıfa ait örnekleri eğitilirken gördüğü, tanıdığı ve öğrendiği diğer sınıfa atamaya eğilimli olacaktır. Bu nedenle; hedef niteliğin sınıf değerlerine ait örneklerin eğitim ve test veri setlerinin her ikisinde de bulunması beklenir.

8.3 Sınıflandırma

Sınıflandırma problemleri danışmanlı öğrenme biçimi ile ilişkilendirilmiştir. Danışmanlı öğrenmede kullanılan algoritmaya veri setindeki her bir örneğe ait özelliklerle beraber o örneğin ait olduğu sınıf bilgisi de verilmektedir. Örneğin; Şekil 4'teki gibi çok sayıda kedi fotoğraflarının toplandığı ve bu fotoğraflar üzerinden kedilerin özelliklerinin (niteliklerinin) çıkarılarak kedilerin ırklarına göre sınıflandırıldığı amaçlansın (G görevi).



Şekil 8.4: Örnek kedi fotoğrafları.

Kedi ırkı sınıflandırma problemi için 1000 kayıttan oluşan Tablo 1'deki gibi bir kedi veri seti kullanılsın (Deneyim D).

Makine öğrenme algoritmasına kedi özellikleri ile kedinin hangi ırktan olduğu bilgisi bir arada verilerek eğitilmesi sağlanır. Daha sonra algoritmaya kulak tipi, kuyruk tipi, renk gibi özellikleri verilen Şekil 5'teki kedinin Tekir, Norveç Orman, Van kedisi gibi ırk tahmin işlemi bir sınıflandırma örneğidir.

Renk	**Kulak tipi**	**Kuyruk tipi**	**Tüy tipi**	**...**	**Irk**
Siyah-Beyaz	Küçük dar	Normal	Yoğun, kısa	...	???

Tablo 8.1: Kedi veri seti.

Renk	Kulak tipi	Kuyruk tipi	Tüy tipi	...	Irk
Siyah-Gri-Truncu	Sivri	Kıvrık	Kısa	...	Tekir
Sarı-Beyaz	Uzun yassı	İnce	Seyrek	...	Norveç Orman
...	...	...	...	...	...
Gri-Beyaz	Kısa geniş	Kısa dolgun	Uzun	...	Tekir

Şekil 8.5: Irkı belirsiz, sınıflandırılmak istenen yeni kedi.

8.4 Dengesiz Veri Setleri

Çoğu standart algoritma, veri setinde dengeli sınıf dağılımının olduğunu / eşit hatalı sınıflandırma maliyeti olduğunu varsayar / bekler (He ve Garcia, 2009). Tersi bir durumda yani algoritmaya dengesiz veri setleri gösterildiğinde, algoritmalar veri dağılım özelliklerini düzgün bir şekilde temsil edemez. Bu ise veri setindeki hedef niteliğin sınıf değerleri arasında sakıncalı doğruluk değerlerinin elde edilmesine sebep olur (He ve Garcia, 2009). Dengesiz sınıf dağılımının gerçek hayatta karşılaşılan problemlerde sıklıkla karşılaşıldığını söylemiştir (Liu, Wu ve Zhou, 2009). Dolandırıcılık tespiti, istenmeyen e-posta filtreleme, hastalık taraması gibi alanlarda görülebilmektedir (elitedatascience.com, 2017).

Algoritmaların performansına etki eden ve genellikle göz ardı edilen nokta veri setindeki sınıflar arası dengedir. "Var" ve "Yok" gibi veya "Hasta" ve "Sağlıklı" gibi iki sınıf değerine sahip bir veri setinin kullanıldığı modelin performansı incelendiğinde sınıfların her birinin doğru tahmin edilme oranının birbirine yakın olması beklenir. Örnek vermek gerekirse %50-%50 sınıf frekansına sahip bir veri setinin kullanıldığı bir modelde sınıflara ait doğruluk değerinin %95-%5 veya %98-%2 sınıf frekansının kullanıldığı modelden iyi olması beklenir.

Bir firma müşteri kayıp analizi (CHURN) gerçekleştirmek istiyor. Müşteri veri tabanından müşterilerine ilişkin tüm veriyi çekerek veri setini oluşturmuştur. Bir müşterinin kayıp müşteri mi yoksa devam eden müşteri mi olduğunu gösteren niteliğin sınıf değerlerinin frekansı arasında Tablo 2'de verilen durum söz konusudur.

Tıpkı bir insan nasıl üzerinde daha fazla durduğu ya da daha fazla araştırma yaptığı bir konuyu daha iyi kavrar ve öğrenirse, makine öğrenmesi algoritmaları da de benzer biçimde çalışmaktadır. Verilen tabloya göre algoritma organizasyonun devam eden müşterilerine ait özellikleri kayıp müşteri özelliklerine göre daha iyi öğrenecektir. Dahası kayıp müşteri özelliklerini belki de hiç öğrenemeyecektir. Azınlıkta olan sınıfa ait örneklerden çok az veya hiç öğrenememiş algoritmanın verdiği

Tablo 8.2: Kayıp müşteri analizi için oluşturulan veri seti.

Toplam Gözlem Sayısı:	1000
Devam Eden Müşteri Sayısı:	950
Kayıp Müşteri Sayısı:	50

yüksek doğruluk değeri yanıltıcı olabilmektedir. Bu nedenle dengesiz veri setlerinde sınıfların doğruluk oranlarındaki bu uçurumu giderebilmek için farklı yöntemlere başvurulmalıdır.

- Örneğin; daha fazla veri toplanarak, sayıca eksik olan sınıf değerinin frekansı arttırılabilir; ancak söz konusu araştırma problemi bir hastalığın teşhisi olduğunda veri toplama diğer problem türlerine göre biraz daha zor hatta imkânsız olabilmektedir.
- Doğruluk değeri dışında farklı performans değerlendirme ölçüleri kullanıldığında analiz sonuçları daha sağlıklı yorumlanabilir. Belirleyicilik, kesinlik, tanısal üstünlük oranı, F-Ölçüsü gibi performans değerlendirme ölçüleri ile daha güvenilir bir değerlendirme yapılabilir.
- Klasik makine öğrenmesi algoritmaları eğitim veri setinde tüm sınıflara ait örneklerin yer almasını ister. Dengesiz veri setleriyle çalışırken sadece tek bir sınıftaki örneklerden öğrenen algoritmalar da kullanılabilir. Örneğin Var / Yok, Az / Çok, Riskli / Risksiz gibi iki sınıf verisini kullanan değil de sadece tek bir sınıf verisiyle öğrenen algoritmalar tercih edilebilir. Literatürde sadece tek bir sınıf ile eğitilen algoritmaların kullanılması tek sınıf-sınıflandırma (one class-classification) olarak geçmektedir. Aykırı durum tespitinde (anomaly detection) sıklıkla tercih edilen bu yöntemde, algoritmaya eğitim aşamasında sadece öğrenmesi istenen sınıfa ait örnekler sunulur (Kartal, Özen ve Gülseçen, 2016). Bu sınıf da genellikle azınlıkta olan örneklere ait sınıf olmaktadır. Böylece algoritma sadece azınlıkta olan örneklerden öğrenmektedir. Test aşamasında ise algoritmaya her iki sınıfa ait örnekler verilerek performans değerlendirmesi gerçekleştirilir. Algoritma test aşamasında öğrendiği sınıfa benzetemediği bir örneği aykırı durum / örnek olarak algılar.

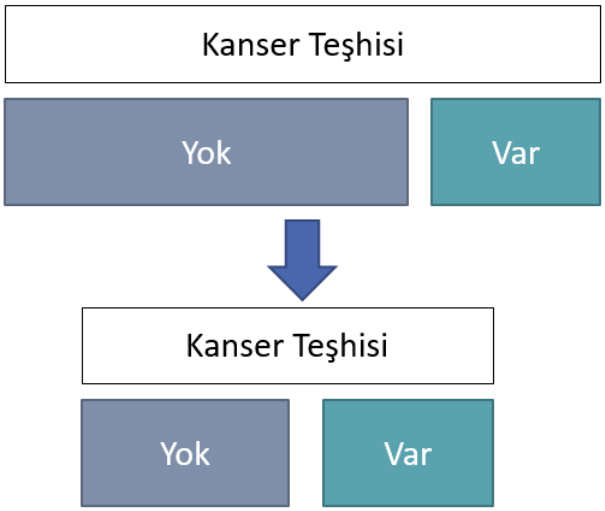
Bu çalışmada dengesiz veri setiyle çalışırken yukarıda bahsedilen yöntemlerden undersampling, oversampling ve sentetik veri üretme yöntemlerinden faydalanılmıştır.

8.4.1 Undersampling

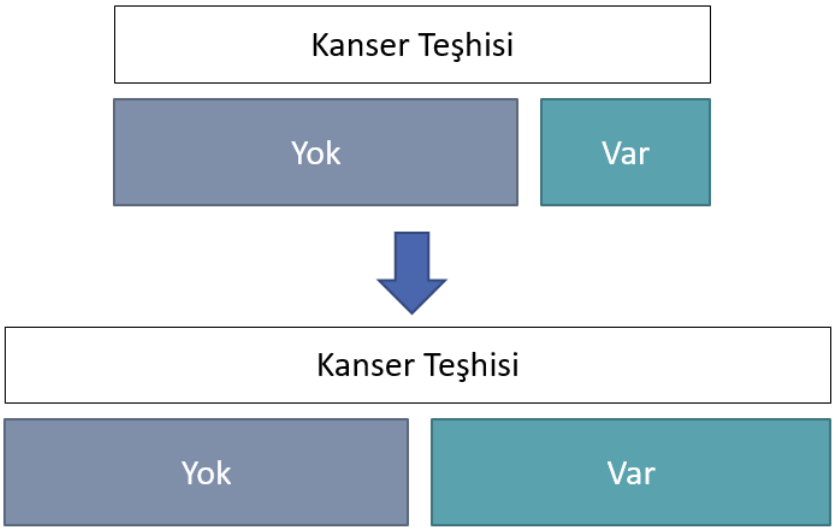
Undersampling yöntemi; hedef niteliğin sayıca çok olan sınıf etiketi sayısının (çoğunluk sınıfı / majority class) sayıca az olan sınıf etiketine (azınlık sınıfı / minority class) yaklaştırılmasıdır (Analytics Vidhya Content Team, 2016; Baesens, Vlasselaer ve Verbeke, 2015) (Şekil 6). Eğer çoğunluk sınıfında yapılan azaltma rastgele yapılırsa Random Undersampling, istatistiksel bilgi kullanılarak yapılırsa buna da İnformed Undersampling adı verilir (Shelke, Deshmukh ve Shandilya, 2017).

8.4.2 Oversampling

Oversampling yöntemi; hedef niteliğin sayıca az olan sınıf etiketi sayısının sayıca çok olan sınıf etiketine yaklaştırılmasıdır (Analytics Vidhya Content Team, 2016; Baesens ve diğerleri, 2015) (Şekil 7). Random Oversampling ve Synthetic Oversampling olarak kategorize edilebilir (Shelke ve diğerleri, 2017): İlk yöntemde mevcut azınlık örnekleri, bir azınlık sınıfının boyutunu arttırmak için çoğaltılırken, diğer yöntemde azınlık sınıfı örnekleri için suni örnekler üretilmektedir.



Şekil 8.6: Undersampling yönteminin şematize edilmesi.



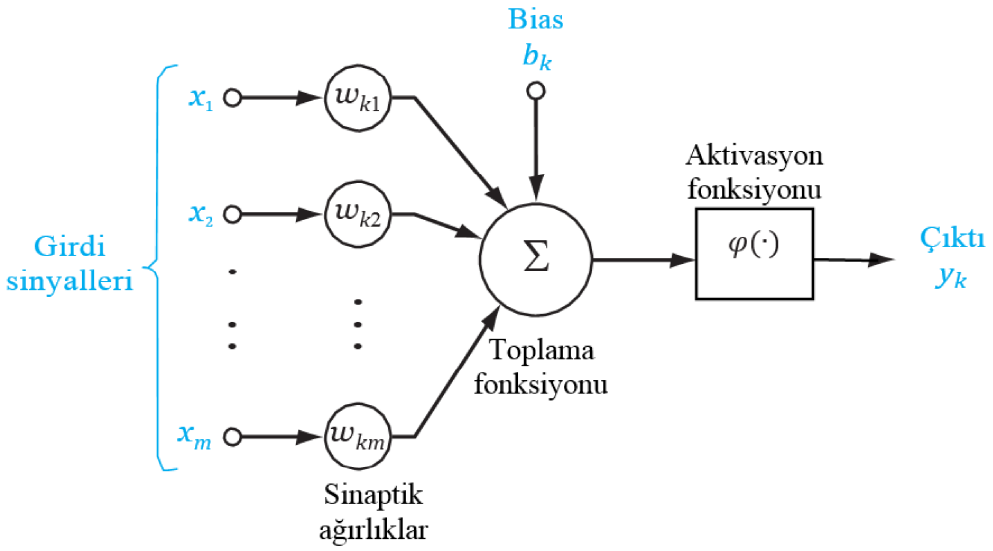
Şekil 8.7: Oversampling yönteminin şematize edilmesi.

8.4.3 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE'de her azınlık sınıfı örneği alınır ve bu örneğe ait örneğin k komşusunun herhangi birine ya da tümüne bakılarak sentetik örnekler oluşturulur. Böylece azınlık sınıfı aşırı örneklenmiş olur (Chawla, Bowyer, Hall ve Kegelmeyer, 2002). Diğer sampling yöntemlerinden en temel farkı, azınlık sınıfındaki örneklerin kopyalanarak çoğaltılması yerine yakın komşularına bakılarak sentetik örneklerin üretilmesidir.

8.5 Yapay Sinir Ağları

Yapay sinir ağları, insanın biyolojik sinir sisteminin çalışma mekanizmasını bilgisayar sistemlerinde taklit ederek bilgisayarların da insanlar gibi öğrenme, tahminde bulunma, sınıflandırma gibi işlevleri yerine getirmesi için kullanılan yapay zekânın bir alt disiplinidir. Yapay sinir ağları, özellikle çözümü doğrusal olmayan problemlere verdiği başarılı çıktılar sayesinde kullanım alanı ve yaygınlığını artırmıştır. Yapay sinir ağları, insan sinir sisteminde olduğu gibi birbirine bağlı basit işlem elemanı olan yapay sinir hücrelerinden (nöron) oluşmaktadır. Şekil 8'de basit bir nöronun bileşenleri gösterilmiştir (Haykin, 2008).



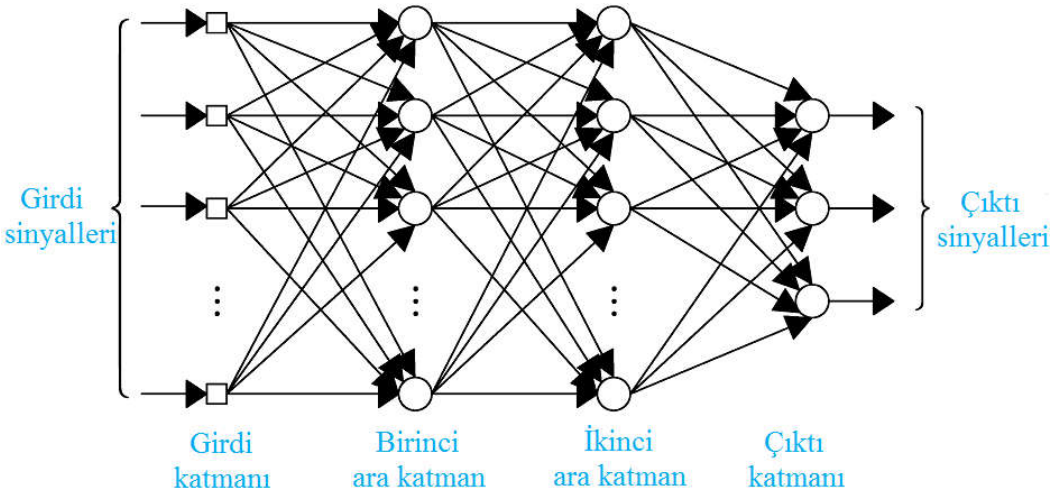
Şekil 8.8: Basit bir nöronun bileşenleri

Basit bir nöronun çalışması şu şekilde gerçekleşmektedir: Ağın eğitimi için nörona verilen girdiler $(x_1, x_2, .., x_m)$, her bir nöronda bulunan ağırlıklar (w) ile çarpılır. Her bir girdiye uygulanan bu işlem sonrası elde edilen çarpımlar ve kullanılıyorsa bias nöronunun ağırlık değeri (b_k) bir toplama fonksiyonuna sokularak bir toplam değer elde edilir. Hesaplanan bu toplam değer, bir sonraki sinir hücresine veya çıktı katmanına genelde olduğu gibi değil bir aktivasyon fonksiyonuna tabi tutularak iletilir. Örneğin aktivasyon fonksiyonu olarak eşik değer (threshold) fonksiyonu seçildiğinde, nöronun hesapladığı toplam değer pozitif sayı ise bir sonraki sinir hücresine 1, değilse 0 iletilir (Fyfe, 2000). Çıktı katmanında ise ağın ulaşması gereken hedef değer ile ağın ürettiği değer arasındaki fark yani ağın hatası hesaplanır.

Tarihi süreç içerisinde birçok yapay sinir ağı mimarisi ve öğrenme algoritması geliştirilmiştir.

Bu çalışmada en çok kullanılan yapay sinir ağı mimarilerinden çok katmanlı ileri beslemeli (multi layer feed forward) yapay sinir ağı mimarisi, ağın öğrenme algoritması olarak da Geri Yayılım (Backpropagation) algoritması kullanılmıştır.

Çok katmanlı ağ mimarisinde girdi katmanı, bir veya daha fazla ara katman (gizli katman) ve çıktı katmanları bulunmaktadır. Şekil 9'da iki ara katmanı olan çok katmanlı yapay sinir ağı mimarisinin yapısına ait grafik verilmiştir (Keller, Liu ve Fogel, 2016).



Şekil 8.9: Çok katmanlı yapay sinir ağı mimarisi

Çok katmanlı ileri beslemeli yapay sinir ağlarında, girdi katmanından alınan sinyaller ara katmanlardaki nöronlar tarafından işlenerek çıktı katmanına doğru iletilirler. Bu mimaride öğrenme algoritması olarak yaygınlıkla Geri yayılım (Backpropagation) algoritması kullanılmaktadır. Geri yayılım algoritmasının çalışma prensibini kısaca şu şekilde özetlemek mümkündür (Özen, Kartal ve Gülseçen, 2017):

- Ağ başlangıç durumuna hazırlanır:

Çok katmanlı mimaride bulunacak ara katman ve her bir ara katmanda kullanılacak nöron sayısı belirlenir. Nöronların ağırlıklarına rastgele küçük değerler atanır. Ağın öğrenme oranı ve hata oranı belirlenir. Ara katman ve çıktı katmanındaki toplama ve aktivasyon fonksiyonları belirlenir.

- Girdi sinyalleri ağda ileri doğru iletilir.

Her bir nöron, gelen sinyalleri ağırlık değeriyle çarpar. Toplama fonksiyonu ile hesaplanan değer bir sonraki katmana aktivasyon fonksiyonu ile iletilir.

- Ağın hatası hesaplanır.

Çıktı katmanında ağın hesaplamış olduğu değer ile gerçek değer arasındaki fark (ağın hatası) ölçülür. Ağın hatası belirlenen hata oranının üzerinde ise ağın henüz öğrenmediği kabul edilir.

- Ağırlıklar güncellenerek çıktı geri yayılır.

Ara katmanlardaki nöronların ağırlıkları güncellenerek hesaplama işlemi bu sefer çıktı katmanından girdi katmanına doğru (geri yayılarak) ters yönlü olarak çalıştırılır. Girdi katmanına gelen sinyallerle güncellenen ağırlıklar kullanılarak tekrar ileri yönlü hesaplamaya devam edilir. Sinyallerin ileri ve geri yönlü hesaplanarak devam etmesi süreci, çıktı katmanında ağın üretmiş olduğu değer ile gerçek değer arasındaki fark, önceden belirlenen hata oranının altında kalana kadar devam eder. Bu durumda ağın öğrenme işlemini tamamladığı kabul edilerek ağın çalışması durdurulur.

Öğrenmesi tamamlanan ağa daha sonra eğitim veri setinde görmediği örnekler (test veri seti) verilerek, örneklerin sınıfları tahmin edilmeye çalışılır. Bu işleme ağın test edilmesi denilmektedir. Test aşaması sonrasında ağın tahmin ettiği sınıf değerleri ile gerçek sınıf değerleri karşılaştırılarak ağın doğruluğu ve diğer performans ölçüleri hesaplanır.

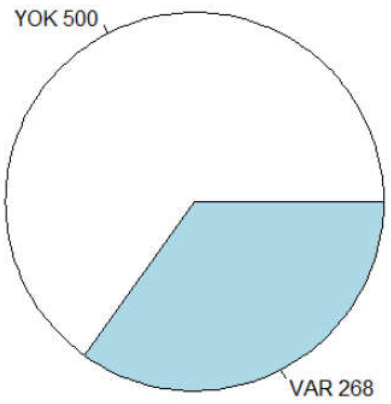
8.6 Yapay Sinir Ağları İle Diyabet Hastalığı Teşhisi Uygulaması

Bu çalışmada veri seti olarak UCI Machine Learning Repository'de ücretsiz kullanıma açık olan veri setlerinden Pima Indians Diyabet veri seti (https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes) kullanmıştır (Lichman, 2013). diyabetDB olarak adlandırılan veri setinde; hamilelik sayısı, glukoz konsantrasyonu, diastolik kan basıncı, triseps kalınlığı, iki saatlik serum insulin değeri, vücut kitle indeksi, diyabet soyağacı fonksiyonu, yaş ve kadının şeker hastası olup olmadığını belirten hedef nitelik olmak üzere dokuz nitelik bulunmaktadır. Yapay sinir ağının öğrenmesi için ilk sekiz nitelik, diyabet hastalığının "Var" ya da "Yok" biçiminde sınıflandırılması için kullanılacaktır. Tüm analizler R programlama dili ile RStudio editörü kullanılarak gerçekleştirilmiştir (cran.r-project.org, 2017; RStudio, 2017). Şekil 10'da veri setinin ilk altı satırını içeren ekran görüntüsü yer almaktadır.

	dogumSayisi	glukozKons	kanBasinci	trisepKalinligi	serumInsulin	VKI	pedigreeFonks	yas	diabetKarar
1	6	148	72	35	0	33.6	0.627	50	VAR
2	1	85	66	29	0	26.6	0.351	31	YOK
3	8	183	64	0	0	23.3	0.672	32	VAR
4	1	89	66	23	94	28.1	0.167	21	YOK
5	0	137	40	35	168	43.1	2.288	33	VAR
6	5	116	74	0	0	25.6	0.201	30	YOK

Şekil 8.10: diyabetDB veri setindeki örneklere genel bakış.

Veri setinde toplamda 768 örnek yer almaktadır. Bunlardan 268 tanesinde diyabet hastalığı görülürken (diyabetKarar = "VAR") 500'ünde görülmemektedir (diyabetKarar = "YOK") (Şekil 11).



Şekil 8.11: Veri setinde diyabetKarar hedef niteliğinin sınıf dağılımı.

Veri setindeki niteliklerin özet bilgisi Şekil 12'deki gibidir.

```
> summary(diabetDB)
  dogumSayisi       glukozKons      kanBasinci     trisepKalinligi
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
  serumInsulin        VKI        pedigreeFonks        yas        diabetKarar
 Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00   YOK:500
 1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00   VAR:268
 Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
 Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
 3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
 Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Şekil 8.12: diyabetDB veri setinin özet görünümü.

Analizlere geçmeden önce veri setindeki hedef nitelik diyabetKarar haricindeki tüm nitelikler, min-max normalizasyon yöntemi ile normalize edilmiştir (Walesiak ve Dudek, 2016). Normalizasyon işlemi gerçekleştirildikten sonra veri seti Şekil 13'deki hale gelmiştir.

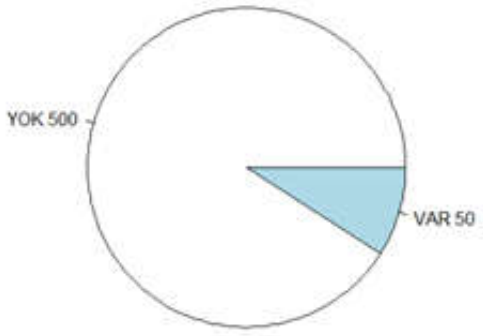
```
> summary(diabetDB)
  dogumSayisi       glukozKons      kanBasinci     trisepKalinligi
 Min.   :0.00000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.05882   1st Qu.:0.4975   1st Qu.:0.5082   1st Qu.:0.0000
 Median :0.17647   Median :0.5879   Median :0.5902   Median :0.2323
 Mean   :0.22618   Mean   :0.6075   Mean   :0.5664   Mean   :0.2074
 3rd Qu.:0.35294   3rd Qu.:0.7048   3rd Qu.:0.6557   3rd Qu.:0.3232
 Max.   :1.00000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
  serumInsulin        VKI        pedigreeFonks        yas        diabetKarar
 Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   YOK:500
 1st Qu.:0.00000   1st Qu.:0.4069   1st Qu.:0.07077   1st Qu.:0.0500   VAR:268
 Median :0.03605   Median :0.4769   Median :0.12575   Median :0.1333
 Mean   :0.09433   Mean   :0.4768   Mean   :0.16818   Mean   :0.2040
 3rd Qu.:0.15041   3rd Qu.:0.5455   3rd Qu.:0.23409   3rd Qu.:0.3333
 Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
```

Şekil 8.13: diyabetDB veri setinin normalizasyon sonrası özet görünümü.

diyabetDB veri setinin sınıf dağılımı dengesiz veri seti tanımına uymamaktadır. Bu nedenle veri setini dengesiz hale getirmek amacıyla toplamda 268 tane olan diyabet hastası örneğinin içinden rastgele 50 adet örnek seçilmiştir. Böylelikle veri setinde 50 diyabet hastasına karşın, 500 hastalığın görülmediği örnek yer almış, hedef niteliğin sınıf dağılımında yaklaşık %90'a %10'luk bir oran elde edilmiştir. Yeni durumda veri setindeki hedef niteliğin sınıf dağılımı Tablo 3'te verilmiştir.

Tablo 8.3: Diyabet hastalığı analizi için oluşturulan diyabet veri seti.

Toplam gözlem sayısı:	550
Diyabet hastalığı görülmeyen örnek sayısı:	500
Diyabet hastalığı görülen örnek sayısı:	50



8.6.1 Dengesiz Veri Seti ile Problemin Çözümü

Çalışmanın bu bölümünde yapay sinir ağları öncelikle dengesiz veri seti ile analize sokulmuştur. Bu durumda diyabetKarar hedef niteliğinin yapay sinir ağlarına gösterilebilmesi için "Var" ve "Yok" sınıf değerleri sırasıyla 1 ve 0 olasılık değerlerine çevrilmiştir. Bunun sebebi yapay sinir ağlarının kategorik niteliklerle çalışmak yerine nümerik değerlerle çalışmasıdır. 1 → Hastalığın görülme durumunu, 0 → hastalığın görülmeme durumunu ifade etmektedir.

Veri seti eğitim ve test olmak üzere Hold-out yöntemi kullanılarak ikiye ayrılmıştır (Kuhn, 2016). Şekil 14'te eğitim ve test veri setlerinde hedef niteliğin sınıflarına ait örnek sayıları verilmiştir.

```
> table(egitim$diabetKarar)

  0   1
350  35
> table(test$diabetKarar)

  0   1
150  15
```

Şekil 8.14: Eğitim ve test veri setindeki sınıf değeri frekansları.

neuralnet kütüphanesi (Fritsch ve Guenther, 2016) kullanılarak gerçekleştirilen yapay sinir ağı analizlerinde tek ara katman kullanılmıştır ve ara katmanda nöron sayısı 10 olarak sabit tutulmuştur. Girdi katmanında 8, çıktı katmanında ise 1 nöron kullanılmıştır. Hata fonksiyonu olarak Hata Kareleri Toplamı (Sum of Squared Errors), öğrenme oranı olarak 0.01, aktivasyon fonksiyonu olarak da Sigmoid (Logistic) fonksiyonu kullanılmıştır. Çıktı nöronunun 0-1 arasında bir değer üretmesi sağlanmıştır. Oluşturulan yapay sinir ağı modeli Şekil 15'te verilmiştir.

Eğitim aşaması sonrasında yapay sinir ağlarına test veri setindeki örnekler gösterilerek modelin performans değerleri hesaplanmıştır. Şekil 16'da ön-işleme yapılmadan yani dengesiz veri seti kullanılarak yapılan analize ait performans değerleri yer almaktadır.

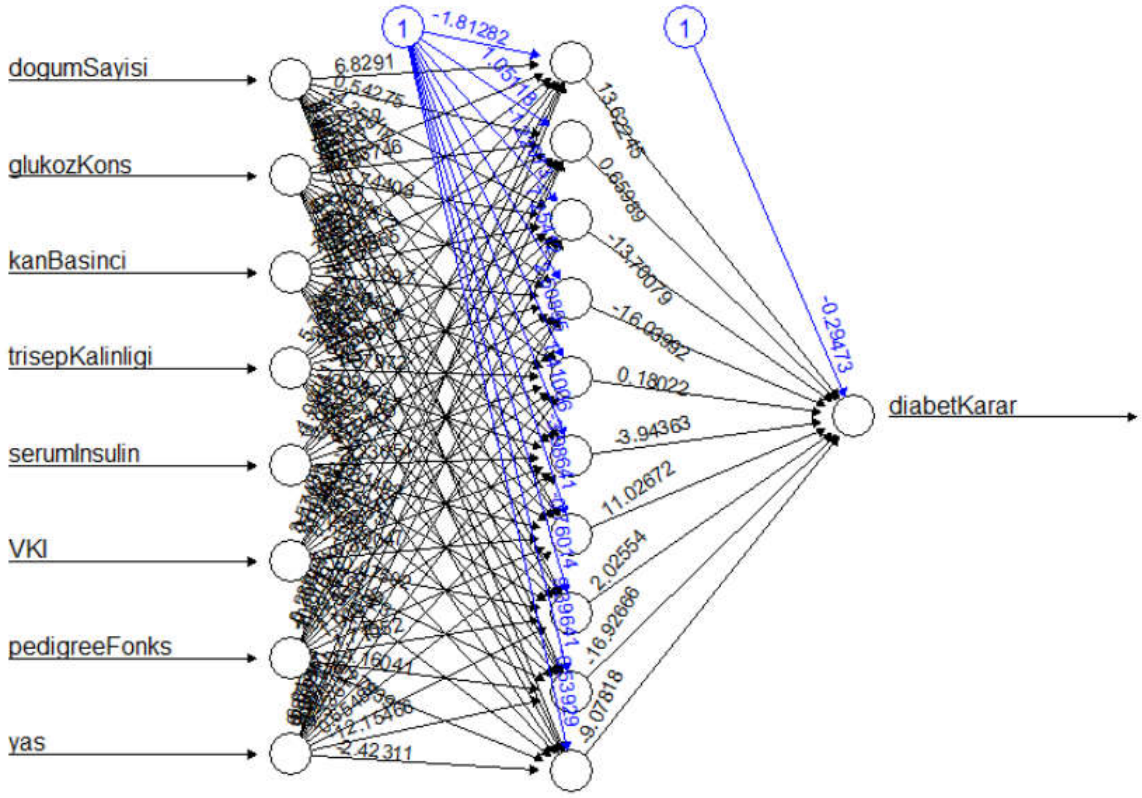
Analiz sonuçlarına bakıldığında modelin doğruluk değerinin 0.903 olduğu görülebilir. Doğruluk değerine bakarak modelin performansının iyi olduğu söylenebilir; ancak hangi sınıfı ne kadar doğru tahmin ettiği de yorumlanmalıdır.

$$Doğruluk = \frac{6+143}{6+7+9+143} = 0.903$$

Yapay sinir ağının verdiği yüksek doğruluk değerinin büyük oranda çoğunluk sınıfından kaynaklandığı açıkça görülmektedir. Oysaki model azınlık sınıfında olan 15 örneğin yalnızca 6'sını doğru tahmin ederek bu sınıfta çok düşük performans göstermiştir. Duyarlılık 0.4, belirleyicilik ise 0.953 olarak bulunmuştur.

$$Duyarlilik = \frac{6}{6+9} = 0.4$$

$$Belirleyicilik = \frac{143}{7+143} = 0.953$$



Şekil 8.15: Çalışmada kullanılan yapay sinir ağı modeli.

```
Confusion Matrix and Statistics

          Reference
Prediction VAR YOK
       VAR   6   7
       YOK   9 143

               Accuracy : 0.9030303
                 95% CI : (0.8473054, 0.9435516)
    No Information Rate : 0.9090909
    P-Value [Acc > NIR] : 0.6688830

                  Kappa : 0.3758865
 Mcnemar's Test P-Value : 0.8025873

            Sensitivity : 0.40000000
            Specificity : 0.95333333
         Pos Pred Value : 0.46153846
         Neg Pred Value : 0.94078947
             Prevalence : 0.09090909
         Detection Rate : 0.03636364
   Detection Prevalence : 0.07878788
      Balanced Accuracy : 0.67666667

       'Positive' Class : VAR
```

Şekil 8.16: Dengesiz veri setinden elde edilen performans değerlendirme sonuçları.

Benzer şekilde negatif öngörü değeri 0.941'lerde iken pozitif öngörü değeri 0.462 bulunmuştur. F-Ölçüsü ise 0.427 olarak hesaplanmıştır.

$$F-Ölcüsü = \frac{2}{\left(\frac{1}{Duyarlılık}+\frac{1}{Kesinlik}\right)} = \frac{2}{\left(\frac{1}{0.400}+\frac{1}{0.462}\right)} = 0.429$$

8.6.2 Undersampling Yöntemi İle Problemin Çözümü

diyabetDB veri setinde azınlıkta olan sınıfa ait (hastalık görülen) örnek sayısı 50 olduğu için, toplamda 500 adet olan hastalık görülmeyen örnekten rastgele 50 tanesi seçilmiş ve bu örnekler, hastalık görülen örnekler ile birleştirilerek toplamda 100 örnekten oluşan yeni bir veri seti oluşturulmuştur. Yeni veri setinde hedef niteliğin sınıf değerlerinin frekans dağılımı Şekil 17'de verilmiştir.

```
> table(diabetDB$diabetKarar)

YOK VAR
 50  50
```

Şekil 8.17: Undersampling yönteminin uygulandığı veri setinde hedef niteliğin sınıf değerlerinin frekans dağılımı.

Veri seti, %70 eğitim ve %30 test olacak şekilde ikiye ayrılmıştır. Hedef niteliğin eğitim ve test veri setlerindeki sınıf değeri dağılımları Şekil 18'deki gibidir.

```
> table(egitim$diabetKarar)

 0  1
35 35
> table(test$diabetKarar)

 0  1
15 15
```

Şekil 8.18: Undersampling uygulandığında eğitim ve test veri setindeki sınıf değeri frekansları.

Undersampling yöntemiyle oluşturulan veri setinin performansı Şekil 19'da yer almaktadır. Buna göre; doğruluk değeri 0.903'ten 0.833'e düşmüş olsa da, duyarlılık 0.4'ten 0.733'e, pozitif öngörü değeri 0.462'den 0.917 değerine, F-ölçüsü ise 0.429'dan 0.815'e çıkmıştır.

8.6.3 Oversampling Yöntemi ile Problemin Çözümü

diyabetDB veri setinde çoğunlukta olan sınıfa ait (hastalık görülmeyen) örnek sayısı 500 olduğu için, azınlıkta olan (hastalık görülen) 50 adet örnek çoğaltılarak 500 örneğe çıkarılmış ve örnekler birleştirilerek toplamda 1000 örnekten oluşan yeni bir veri seti oluşturulmuştur. Hedef niteliğin sınıf değerlerinin frekans dağılımı Şekil 20'de verilmiştir.

Benzer şekilde veri seti, %70 eğitim ve %30 test olacak şekilde ikiye ayrılmıştır. Hedef niteliğin eğitim ve test veri setlerindeki sınıf değeri dağılımları Şekil 21'deki gibidir.

Modelin test veri seti üzerindeki performansı Şekil 22'den incelenebilir. Buna göre doğruluk değeri değişmemiş, duyarlılık 0.973 ve belirleyicilik 0.833, pozitif öngörü değeri 0.854, negatif öngörü değeri 0,969 ve F-ölçüsü 0.91 olacak şekilde hesaplanmıştır. Oversampling yöntemi uygulanarak elde edilen performans değerlendirme sonuçları; duyarlılık, pozitif ve negatif öngörü değerleri ile F-ölçüsünün dengesiz veri seti ile yapılan analiz sonuçlarına göre arttığını göstermektedir.

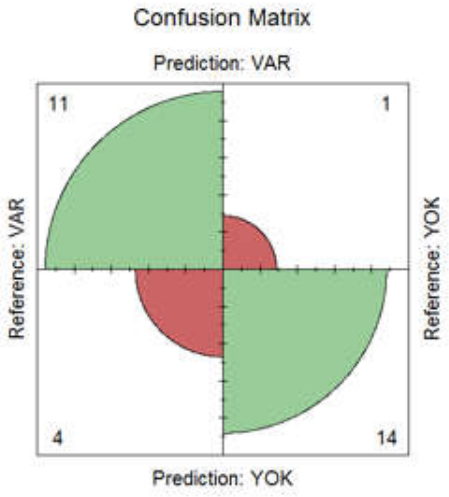
8.7 SMOTE Yöntemi ile Problemin Çözümü

Problemi SMOTE ile çözerken (Torgo, 2010) veri setinde hedef nitelikteki azınlık sınıfının frekans değeri ve çoğunluk sınıfı frekans değerleri birbirine yaklaştırılmıştır. Yani toplamda 50 adet olan hastalık görülen örnekler en yakın beş komşusuna bakılarak çoğaltılmış, hastalık görülmeyen örnekler ise azaltılmıştır. Toplamda 500 örnekten oluşan bir veri seti oluşturulmuştur. Hedef niteliğin sınıf değerlerinin frekans dağılımı Şekil 23'te verilmiştir.

Benzer şekilde yine veri seti, %70 eğitim ve %30 test olacak şekilde ikiye ayrılmıştır. Yeni durumda hedef niteliğin eğitim ve test veri setlerindeki sınıf değeri dağılımları Şekil 24'teki gibidir.

SMOTE yöntemiyle oluşturulan modelinin performansı Şekil 25'te verilmiştir. Buna göre doğruluk değeri 0.903'ten 0.847'ye düşse de, undersampling ve oversampling yöntemlerinde olduğu gibi duyarlılık (0.912) ve belirleyicilik (0.793) değerleri ile pozitif öngörü değeri (0.785) ve negatif öngörü değeri (0.915) birbirine daha yakın elde edilmiştir. Bu bilgiler ışığında F-ölçüsü ise SMOTE yöntemi uygulanarak elde edilen performans değerlendirme sonuçlarının ön-işleme gerçekleştirilmeden (dengesiz veri seti ile) yapılan analiz sonuçlarına göre kıyaslaması yapıldığında duyarlılık ve

```
Confusion Matrix and Statistics

          Reference
Prediction VAR YOK
       VAR  11   1
       YOK   4  14

               Accuracy : 0.8333333
                 95% CI : (0.6527883, 0.9435783)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.0001624571

                  Kappa : 0.6666667
 Mcnemar's Test P-Value : 0.3710933695

            Sensitivity : 0.7333333
            Specificity : 0.9333333
         Pos Pred Value : 0.9166667
         Neg Pred Value : 0.7777778
             Prevalence : 0.5000000
         Detection Rate : 0.3666667
   Detection Prevalence : 0.4000000
      Balanced Accuracy : 0.8333333

       'Positive' Class : VAR
```



Şekil 8.19: Yapay sinir ağı modelinin veri setine undersampling uygulandığında elde edilen performans değerlendirme sonuçları.

```
> table(diabetDB$diabetKarar)

YOK VAR
500 500
```

Şekil 8.20: Oversampling yönteminin uygulandığı veri setinde hedef niteliğin sınıf değerlerinin frekans dağılımı.

```
> table(egitim$diabetKarar)

  0   1
350 350
> table(test$diabetKarar)

  0   1
150 150
```

Şekil 8.21: Oversampling uygulandığında eğitim ve test veri setindeki sınıf değeri frekansları.

pozitif öngörü değerlerinin arttığı görülmektedir.

8.8 Tartışma ve Sonuçlar

Kitabın bu bölümünde yazarlar; dengesiz veri seti kullanılarak oluşturulan yapay sinir ağı modelinin performans değeri ile undersampling, oversampling ve SMOTE yöntemlerinin kullanıldığı modellerin performans değerlerini karşılaştırmış ve yorumlamıştır. Bu kapsamda ele alınan diyabet veri seti, dengesiz veri seti haline getirilmiş, yani sınıf dağılımları dengesiz veri seti karakterini yansıtacak şekilde düzenlenmiştir. Sonraki aşamada, performans değerlendirme yöntemi olarak Hold-out seçilmiştir. Veri seti %70'lik kısmı eğitim ve %30'luk kısmı test veri seti olacak şekilde ikiye ayrılmıştır. Yapay sinir ağı modeli eğitim veri seti ile eğitilmiş olup, modelin test veri seti ile performansı ölçülmüştür.

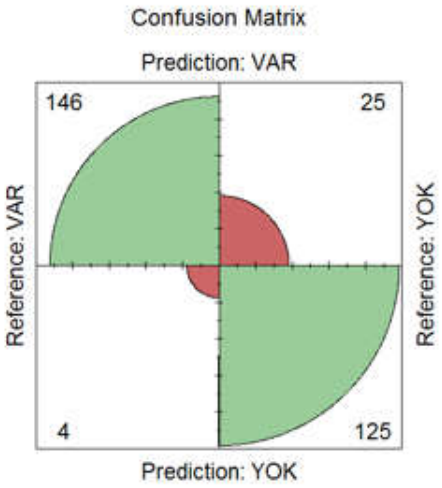
Dengesiz veri seti ile gerçekleştirilen analizin performans değerlerinin literatürdeki örneklere benzer biçimde çok yüksek bir doğruluk değerinde; ancak düşük duyarlılık ve pozitif öngörü değerinde olduğu görülmüştür.

Dengesiz veri setlerinde görülen bu durumun üstesinden gelebilmek için önerilen yöntemlerden undersampling, oversampling ve SMOTE yöntemleri veri setine uygulanmış ve oluşan yeni veri setlerinin her biri üzerinde yapay sinir ağı analizleri tekrarlanmıştır. Tablo 4'te bu dört veri setiyle gerçekleştirilen analiz sonuçlarına yer verilmiştir.

Tablo 4'te verilen F-ölçüsü değerlerine göre;

- En kötü performans ön-işleme olmadan dengesiz veri seti üzerinde gerçekleştirilen analizde elde edilmiştir (0.429).

```
Confusion Matrix and Statistics

          Reference
Prediction VAR YOK
       VAR 146  25
       YOK   4 125

               Accuracy : 0.9033333
                 95% CI : (0.8641251, 0.9343014)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 0.00000000000000022204

                  Kappa : 0.8066667
 Mcnemar's Test P-Value : 0.000204084

            Sensitivity : 0.9733333
            Specificity : 0.8333333
         Pos Pred Value : 0.8538012
         Neg Pred Value : 0.9689922
             Prevalence : 0.5000000
         Detection Rate : 0.4866667
   Detection Prevalence : 0.5700000
      Balanced Accuracy : 0.9033333

       'Positive' Class : VAR
```



Şekil 8.22: Yapay sinir ağı modelinin veri setine oversampling uygulandığında elde edilen performans değerlendirme sonuçları.

```
> table(diabetDB$diabetKarar)

YOK VAR
300 200
```

Şekil 8.23: SMOTE yönteminin uygulandığı veri setinde hedef niteliğin sınıf değerlerinin frekans dağılımı.

```
> table(egitim$diabetKarar)

  0   1
218 132
> table(test$diabetKarar)

 0  1
82 68
```

Şekil 8.24: SMOTE uygulandığında eğitim ve test veri setindeki sınıf değeri frekansları.

Tablo 8.4: Genel performans değerlendirme tablosu.

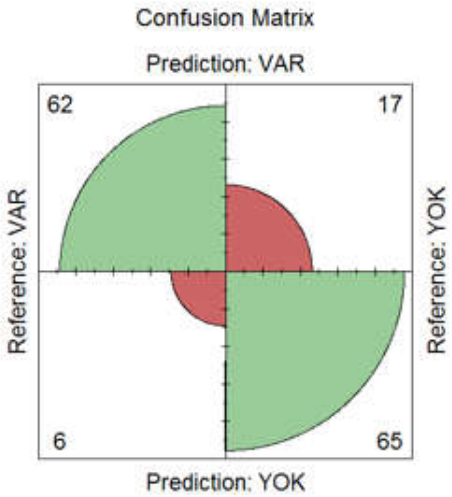
	Doğruluk	Duyarlılık	Belirleyicilik	Poz. Öng. Değ.	Neg. Öng. Değ.	F-Ölçüsü
Ön-işleme olmadan	0.903	0.4	0.953	0.427	0.941	0.429
Under Sampling	0.833	0.733	0.93	0.917	0.778	0.815
Over Sampling	0.903	0.973	0.833	0.854	0.969	0.91
SMOTE	0.847	0.912	0.793	0.785	0.915	0.844

- En iyi performans ise, oversampling ile elde edilmiştir (0.91). Oversampling yöntemini sırasıyla SMOTE (0.844) ve undersampling (0.815) yöntemleri takip etmektedir.

Dengesiz veri seti ve çeşitli yöntemlerle dengeli hale getirilen veri setleri ile yapılan analiz sonuçları, veri setindeki hedef niteliğin sınıf dağılımlarının, en az kullanılan model ve algoritma kadar önemli olduğunu göstermektedir. Bu sebeple kurulan model ne olursa olsun, veri setindeki hedef niteliğin sınıf değeri dağılımları da kontrol edilerek analizlere devam edilmelidir. Veri setindeki hedef niteliğin sınıf değerlerine ait frekans değerleri arasında bir dengesizlik tespit edilirse, bir kısmı bu bölümde anlatılan teknikler ile veri seti dengeli hale getirilerek analizlerin tekrarlanmasının faydalı olacağı düşünülmektedir.

Bu çalışmada yapılan analiz sonuçlarının göstermiş olduğu bir diğer önemli husus ise, çeşitli algoritmalarla (yapay sinir ağları, genetik algoritmalar, klasik makine öğrenmesi algoritmaları vb) kurulan modelin performansı ölçülürken dikkat edilmesi gereken tek kıstas doğruluk değeri olmamalıdır. Özellikle dengesiz veri setlerinde çoğunlukta olan sınıfa ait örnekler yüksek başarıyla doğru sınıflandırılırken, sayıca azınlıkta olan sınıfa ait örnekler ise çok düşük doğrulukla sınıflandırılmaktadır. Bu nedenle sadece doğruluk değerine bakarak modelleri kıyaslamak yanıltıcı olabilmektedir.

```
Confusion Matrix and Statistics

          Reference
Prediction VAR YOK
       VAR  62  17
       YOK   6  65

               Accuracy : 0.8466667
                 95% CI : (0.7788603, 0.900248)
    No Information Rate : 0.5466667
    P-Value [Acc > NIR] : 0.00000000000005731405

                  Kappa : 0.6948523
 Mcnemar's Test P-Value : 0.03705622

            Sensitivity : 0.9117647
            Specificity : 0.7926829
         Pos Pred Value : 0.7848101
         Neg Pred Value : 0.9154930
             Prevalence : 0.4533333
         Detection Rate : 0.4133333
   Detection Prevalence : 0.5266667
      Balanced Accuracy : 0.8522238

       'Positive' Class : VAR
```



Şekil 8.25: Yapay sinir ağı modelinin veri setine SMOTE uygulandığında elde edilen performans değerlendirme sonuçları.

Modellerin performanslarını karşılaştırmak için doğruluk değerinin yanında F-ölçüsü gibi birkaç performans ölçüsüyle hesaplanan ölçülerin dikkate alınması tavsiye edilmektedir.

8.9 Kaynakça

Analytics Vidhya Content Team. (2016, 28 Mart). Practical Guide to deal with Imbalanced Classification Problems in R. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/ adresinden erişildi.

Baesens, B., Vlasselaer, V. V. ve Verbeke, W. (2015). Fraud Analytics Using Descriptive, Predictive, and Social Network Techniques: A Guide to Data Science for Fraud Detection. John Wiley & Sons.

Balaban, M. E. ve Kartal, E. (2015). Veri Madenciliği ve Makine Öğrenmesi Temel Algoritmaları ve R Dili ile Uygulamaları (Birinci Baskı.). İstanbul: Çağlayan Kitabevi.

Chawla, N. V., Bowyer, K. W., Hall, L. O. ve Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321–357.

cran.r-project.org. (2017). The Comprehensive R Archive Network. 14 Haziran 2017 tarihinde https://cran.r-project.org/ adresinden erişildi.

elitedatascience.com. (2017, 5 Temmuz). How to Handle Imbalanced Classes in Machine Learning. EliteDataScience. 26 Eylül 2017 tarihinde https://elitedatascience.com/imbalanced-classes adresinden erişildi.

Fritsch, S. ve Guenther, F. (2016). neuralnet: Training of Neural Networks. https://CRAN.R-project.org/package=neuralnet adresinden erişildi.

Fyfe, C. (2000). Artificial neural networks and information theory. University of Paisley. http://bit.ly/2yTVxIY adresinden erişildi.

Haykin, S. O. (2008). Neural Networks and Learning Machines (3 edition.). New York: Pearson.

He, H. ve Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. doi:10.1109/TKDE.2008.239

Kartal, E. (2015). Sınıflandırmaya Dayalı Makine Öğrenmesi Teknikleri ve Kardiyolojik Risk Değerlendirmesine İlişkin Bir Uygulama. İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Kartal, E., Özen, Z. ve Gülseçen, S. (2016). An Artificial Neural Network Approach To Predict Software Project Effort: One-Class vs. Binary Classification. V. Tecim, Ç. Tarhan ve C. Aydın (Ed.), Smart Technology & Smart Management Akıllı Teknoloji & Akıllı Yönetim içinde (ss. 206–216). İzmir: Gülermat Matbaacılık. http://bit.ly/2hONnGP adresinden erişildi.

Keller, J. M., Liu, D. ve Fogel, D. B. (2016). Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation (1 edition.). Hoboken, New Jersey: John Wiley & Sons.

Kuhn, M. (2016). caret: Classification and Regression Training. http://bit.ly/2gOjIy4 adresinden erişildi.

Lichman, M. (2013). UCI Machine Learning Repository. 2 Haziran 2015 tarihinde erişildi. Adres: http://archive.ics.uci.edu/ml

Liu, X.-Y., Wu, J. ve Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), 539–550.

Mitchell, T. M. (1997). Machine Learning (1st Edition.). McGraw-Hill Science/Engineering/-Math.

Özen, Z. (2016, 25 Mayıs). Kimlik Doğrulaması için Tuş Vuruş Dinamiklerine Dayalı Bir Güvenlik Sisteminin Yapay Sinir Ağları ile Geliştirilmesi. (Yayımlanmamış ph.d. thesis). İstanbul Üniversitesi, İstanbul.

Özen, Z., Kartal, E. ve Gülseçen, S. (2017). Yapay Zekâ ve Yapay Sinir Ağları. Ç. Toros Rifat ve O. Aliefendioğlu (Ed.), Bilgisayar Bilimine Giriş içinde (1., ss. 523–558). İstanbul: Papatya Bilim Yayınevi. http://bit.ly/2zjPFFj adresinden erişildi.

RStudio. (2017). RStudio: Integrated development environment for R. Boston, MA.: RStudio. http://www.rstudio.com/ adresinden erişildi.

Shelke, M. S., Deshmukh, P. R. ve Shandilya, V. K. (2017). A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique. International Journal of Recent Trends in Engineering and Research, 3(4), 444–449. doi:10.23883/IJRTER.2017.3168.0UWXM

Torgo, L. (2010). Data Mining with R, learning with case studies. Chapman and Hall/CRC. http://www.dcc.fc.up.pt/ ltorgo/DataMiningWithR adresinden erişildi.

Walesiak, M. ve Dudek, A. (2016). clusterSim: Searching for Optimal Clustering Procedure for a Data Set. https://CRAN.R-project.org/package=clusterSim adresinden erişildi.

wikipedia. (2017). Sensitivity and specificity. http://bit.ly/2yRzuTk adresinden erişildi.

8.10 Yazarlar Hakkında

Dr. Elif KARTAL, 1 Aralık 1986 yılında Sakarya'da doğmuştur. 2004 yılında Sakarya Anadolu Lisesi'nden mezun olmuş, aynı yıl İstanbul Üniversitesi Fen Fakültesi Matematik Bölümü'nde lisans öğrenimine başlamıştır. 2008 yılında, İ.Ü. Enformatik Bölümü'nde yüksek lisansa başlamış ve yine aynı bölüme Araştırma Görevlisi olarak kabul edilmiştir. İ.Ü. Enformatik Bölümü'nde, 2011 yılında Yapay Sinir Ağları ile Yazılım Projesi Maliyet Tahmini adlı teziyle yüksek lisansını, 2015 yılında, Sınıflandırmaya Dayalı Makine Öğrenmesi Teknikleri ve Kardiyolojik Risk Değerlendirmesine İlişkin Bir Uygulama adlı teziyle doktorasını tamamlamıştır.

2013 yılında 2 ay süreliğine ABD, Ball State University'de ziyaretçi araştırmacı olarak bulunmuştur. Yapay Zekâ, Makine Öğrenmesi ve Veri Madenciliği konularında çalışan Elif Kartal, diğer araştırma alanları olarak Web Tasarımı, Programlama Dilleri, e-Öğrenme konularında çalışmalarını sürdürmektedir.

Dr. Zeki ÖZEN, lisans eğitimini İstanbul Üniversitesi Fen Fakültesi Fizik Bölümü'nde 2008 yılında tamamlamıştır. Aynı yıl İ.Ü. Enformatik Bölümü'nde yüksek lisansa başlamış 2010 yılında bölüme Araştırma Görevlisi olarak kabul edilmiştir. "Bilişim Hukukunda Kaynak Kod İntihali" isimli yüksek lisans tezini 2012 yılında, "Kimlik Doğrulaması için Tuş Vuruş Dinamiklerine Dayalı Bir Güvenlik Sisteminin Yapay Sinir Ağları ile Geliştirilmesi" isimli doktora tezini 2016 yılında tamamlamıştır. 2013 yılında 2 ay süreliğine ABD, Ball State University'de ziyaretçi araştırmacı olarak bulunmuştur. Yapay Sinir Ağları ve Makine Öğrenmesi, Kaynak kod benzerliği İlişkisel veri tabanı konularında çalışmaktadır.

9. Görüntü İşleme Ağırlık Tespiti

Tedarik Zinciri Güvenliği İçin Palet Ağırlıklarının Görüntü İşleme Teknikleri İle Belirlenmesi

Arş. Gör. Tuğçen HATİPOĞLU, Y. Müh. Tevfik ALTINALEV, Prof. Dr. Alpaslan FIĞLALI

Özet

Lojistik sektörünün hızlı gelişimi ve teknolojik ilerlemelerle paralel olarak tedarik zinciri güvenliği konusu da daha fazla önem kazanmaktadır. Bazı tedarik zincirleri, yapıları nedeniyle, tedarik zinciri tehlikelerine daha çok maruz kalmaktadır. Çalışmanın gerçekleştirdiği otomotiv sektörü de tehlike seviyesinin yüksek olduğu sektörler sınıfında yer almaktadır. Güvenli bir tedarik zinciri oluşturmak, zincir içerisinde yer alan bütün bileşenlerin güvenliğinin sağlanması ve her bir element için kapsamlı bir sorumluluk ağı yaratılması ile mümkün olabilir. Birçok ülke ve uluslararası firma, tedarik zinciri güvenliğini sağlamak için kullanılacak stratejiler ve programlar geliştirmektedir. Bu kapsamda, çalışmada otomotiv sektörünün öncü firmalarından birinde tedarikçilerden gelen ve müşterilere gönderilecek olan sevkiyatın güvenilirliğini sağlamak için, toplam palet ağırlığının güvenilir olarak belirlenmesi amaçlanmıştır. Toplam palet ağırlığı içerisinde yer alan parça net ağırlığı, palet ağırlığı ve ambalaj malzemesi ağırlığı gibi bilgilerin bilinmeleri durumunda palet içerisinde eksik/fazla parça veya yabancı malzeme bulunup bulunmadığı belirlenebilecektir. Çalışma kapsamında, güvenli sevkiyat ağırlığı içerisinde en yüksek değişkenliğe sahip palet ağırlıkların doğru tahmin edilebilmesi için görüntü işleme teknikleri kullanılarak değişik boyuttaki paletlerin çıta ve takoz kalınlıkları belirlenmiştir.

Anahtar kelimeler: Tedarik Zinciri Güvenliği, Görüntü İşleme, Tahminleme.

9.1 Giriş

İnternet kullanımı ve e-ticaretin yaygınlaşması lojistik sektörünü dünyada en hızlı gelişme gösteren sektörlerin başına taşımaktadır (Atmaca ve Turgut, 2015). Bununla birlikte teknolojik gelişmeler ve

globalleşme sonucunda tedarik zincirleri de karmaşık yapılı ve pek çok sayıda partnerin bir arada çalıştığı sistemler olarak karşımıza çıkmaktadır. Lojistik hizmet sunucularının son dönemde üzerinde sıkça durdukları alanlardan birisi de Güvenli Tedarik Zinciri konusudur. Tedarik zincirinin güvenliği, kargonun güvenli ve sağlam bir şekilde paketlenmesiyle başlar ve kargonun paketlenmeden son noktadaki nihai kullanıcıya ulaşıncaya kadar meydana gelecek her turlu hasara karşı oluşturulan güvenlik kontrolleriyle devam eder.

Tedarik Zincirinin Maruz Kalabileceği Riskler aşağıdaki gibi sınıflandırılmaktadır (Gupta, 2015):

1) Fiziksel tehlikeler ve riskler

-Fonksiyonel bozulma, -Kaza ve kasıtlı zarar. -Ulaşım (soygun, hırsızlık) -Terörizm

2) Operasyonel tehlikeler ve riskler

-Güvenlik kontrolü -İnsan (hatalı iletişim) -Şirketin performansını, durumunu ve güvenliğini etkileyen aktiviteler

3) Doğal felaketler

-Güvenlik önlemlerinin ve malzemenin kullanılamaz hale gelmesine sebep olabilir. -Şirketin kontrolü dışında etmenler -Dışarıdan alınan malzeme ve servislerdeki arızalar.

Özellikle otomotiv sektöründe Güvenli Tedarik Zinciri Uygulamaları, sektörün aşağıda listelenen özellikleri bakımından çok önemlidir (Manners-Bell, 2014) :

-Bir araba 15,000 kadar parçaya sahip olabilir ve bunlardan sadece bir tanesinin eksikliği üretimin yavaşlamasına hatta durmasına sebep olabilir.

-Yüksek teknolojili parçalar bir aracın maliyetinin yaklaşık % 40'ına denk gelmektedir.

-Bu parçaların büyük çoğunluğu gelişmekte olan ülkelerin üretim bölgelerinde yoğunlaşmış durumdadır ve bu da kırılganlığı arttırmaktadır.

-Tedarikçilere olan aşırı güven, kalite problemlerine yol açabilir.

-Müşteri profili incelendiğinde siparişler genel anlamda periyodik olarak alınmaktadır, bu durum da sektörü ani talep dalgalanmalarına karşı savunmasız bırakmaktadır (2008-2009 krizi sırasında görüldüğü üzere)

-Değişken yeni marketlerde, öngörülen büyüme seviyelerine bağımlılık çok üst seviyededir.

Bu çalışma, yukarıda önemi ve büyüklüğü anlatılan tedarik zincirindeki kırılganlık, riskler ve belirsizlikler altında ulaşım, lojistik ve nakliye ağlarının güvenliğinin sağlanması için bir otomotiv firması deposunda gerçekleştirilmiştir. Çalışmanın amacı, güvenilir tedarik zinciri yönetim sistemi kapsamında belirli sayıda malzeme içeren paletlerin ağırlıklarının belirlenmesinde en yüksek değişkenliğe sahip palet ağırlıklarının tahmin edilmesidir.

9.2 Amaç ve Yöntem

Farklı boyutlara sahip paletlerin çıta ve takoz kalınlıklarını kullanarak istatistiksel bir yöntemle paletlerin ağırlıkları belirlenmek istenmektedir. Bu maksatla, çalışmanın bu bölümünde görüntü işleme teknikleri kullanılarak değişik boyuttaki paletlerin çıta ve takoz kalınlıkları belirlenmeye çalışılmıştır. Paletlerdeki çıta ve takoz kalınlıkları resimlerde kullanılan kalınlığı daha önceden bilinen referans bir nesneyle karşılaştırılarak bulunmuştur.

9.3 Paletler ve Resimler Hakkında Genel Bilgi

Çıta ve takoz kalınlıkları belirlenmek üzere üç farklı boyutta palet kullanılmıştır. Bu paletlerin boyutları 90×114, 170×132 ve 170×114 cm.dir.

Her bir palet tipi için aynı ortamda RGB (Red, Gren, Blue) formatında çekilmiş 30 adet fotoğraf kullanılmıştır. Görüntü işleme teknikleri uygulanmadan önce çıta ve takoz kalınlıklarının daha iyi belirlenebilmesi için fotoğraflar 90 derece sağa döndürülmüştür. Döndürme işleminden sonra fotoğrafların boyutları 4608×3456 pikseldir.

9.4 Kullanılan Yöntem ve Algoritma

Literatüre bakıldığında birçok alanda uygulama olanağı bulan görüntü işleme tekniklerinin kumaş hatalarının tespiti (Çelik vd., 2012), ergonomide çalışma duruşlarının analizi (Figlali vd., 2015) ve mimari yapıların yeniden modellenmesi (Kelong vd., 2008) gibi spesifik konularda da başarılı olarak uygulandığı görülmektedir.

Paletlerin çıta ve takoz kalınlıklarını bulmak için kullanılacak resimler hazır hale getirildikten sonra görüntü işleme teknikleriyle dönüştürme ve kenar algılama işlemlerine tabi tutulur. Daha sonra işlenmiş resimler üzerinden kalınlıklar belirlenir.

Görüntü işleme teknikleri Java programlama dilinin görüntü işleme için hazırlanmış hazır kütüphaneleri kullanılarak yapılmıştır.

Görüntü işleme için kullanılan algoritma üç adımdan oluşmaktadır.

9.4.1 Birinci Adım

Öncelikle görüntü işleme için hazırlanmış RGB formatındaki fotoğraflar literatürde mevcut yöntemlerden birisi kullanılarak Grayscale (gri tonlamalı) formatına dönüştürülür. Literatürde yaygın olarak kullanılan dönüşüm formülü insanların renkleri algılamasına uygun olarak ağırlıklandırılan dönüşüm formüldür. Bu yöntemde fotoğrafın her pikselinin Grayscale değeri aşağıdaki formül yardımıyla hesaplanır.

$$Gray = 0.30 * Red + 0.59 * Green + 0.11 * Blue$$

Bu formülde fotoğrafın her pikselinin Red, Green ve Blue bileşenleri uygun ağırlık katsayılarıyla çarpılarak toplanmaktadır. Katsayılar insanların renkleri algılamasına göre belirlenmektedir.

9.4.2 İkinci Adım

İkinci adımda, kenar algılama teknikleri uygulanarak gri tonlamalı imge haline dönüştürülen resmin kenarları belirlenir. Kenar algılama işlemi temelde resmin çeşitli filtrelerden geçirilerek yüksek parlaklığa sahip piksellerinin vurgulanma işlemidir. Çünkü fotoğraflarda genellikle yüksek parlaklığa sahip alanlar kenarları ifade ederler.

Literatürde kenar algılama amacıyla kullanılan çok sayıda kenar algılama operatörü ve filtresi vardır. Bizim projemizde 1968 yılında Sobel ve Feldman tarafından bulunan 3×3 boyutunda Sobel Kenar Algılama operatörü kullanılmıştır. Resim Sobel Kenar Algılama operatörü ile filtrelendikten sonra her pikselin mutlak gradyen değeri hesaplanır ve daha sonra daha önceden belirlenmiş bir eşik değeri kullanılarak kenar olabilecek yüksek parlaklıktaki pikseller süzülerek diğer pikseller siyah renk olarak belirlenir. Böylece resmin kenar algılaması tamamlanmış olur.

9.4.3 Üçüncü Adım

Üçüncü adımda, kenarları algılanmış resmin y-eksenine izdüşümü hesaplanarak grafik üzerinde gösterilir. Resmin izdüşümü aşağıdaki şekilde bulunur.

(i) Resmin her bir y değerindeki izdüşümü, bütün x değerlerindeki parlaklık değerleri toplanarak hesaplanır.

(ii) Hesaplanan değerler iki boyutlu çizgi grafikle gösterilir.

(ii) Daha sonra grafiğe bakılarak çıtalara, takoz bölümüne ve referans nesneye karşılık gelen yerler belirlenerek kalınlıklar piksel cinsinden hesaplanır.

Son olarak piksel cinsinden hesaplanan değerler, referans nesnenin kalınlığıyla oranlamaya tabi tutularak istenen uzunluk birimi cinsinden hesaplanır.

9.5 Örnek Üzerinde Algoritmanın Gösterimi

90×114 cm. boyutunda palet için çekilmiş RGB formatında ve 90 derece sağ döndürülmüş 30 adet fotoğraftan bir tanesi Şekil 1.'de gösterilmiştir.

Şekil 9.1: RGB Formatında Orijinal Fotoğraf

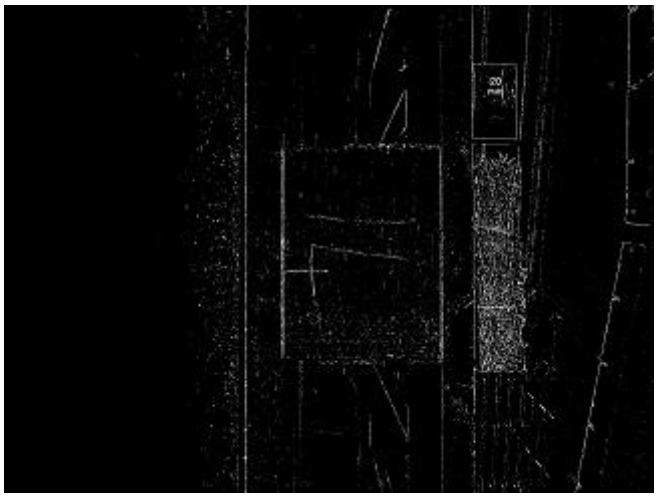
Birinci adımda üçüncü bölümde anlatılan dönüştürme tekniği kullanılarak RGB formatındaki imge Grayscale formatına dönüştürülür. Elde edilen gri tonlamalı resim Şekil 2'de gösterilmiştir.

İkinci adımda, üçüncü bölümde anlatılan kenar algılama tekniği kullanılarak gri tonlamalı resmin kenarları belirlenir ve 40 eşik değeri için süzülerek kenarlar daha vurgulu hale getirilir. Kenarları algılanmış resim Şekil 3'de gösterilmiştir.

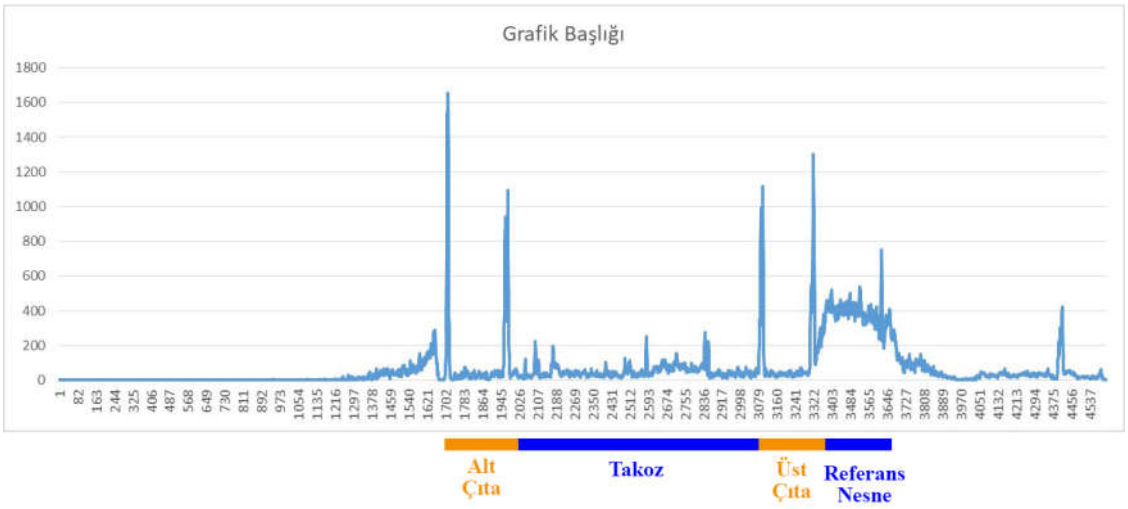
Üçüncü adımda, kenarları algılanmış resmin y eksenine izdüşümü bulunarak iki boyutlu çizgi grafikle gösterilir. Daha sonra grafik üzerinde çıta, takoz ve referans nesneye karşılık gelen yerler belirlenir. İzdüşüm grafiği ve çıta, takoz, referans nesne olarak belirlenen bölgeler Şekil 4'de gösterilmiştir.

Son olarak izdüşüm grafiğine bakılarak alt çıta kalınlığı 263, takoz kalınlığı 1122, üst çıta kalınlığı 223 ve referans nesne kalınlığı 300 piksel olarak belirlenir. Referans nesnenin kalınlığı 20 mm. olarak bilindiği için çıta ve takoz kalınlıkları oranlanarak mm. cinsinden bulunur.

Şekil 9.2: Grayscale Formatındaki Fotoğraf



Şekil 9.3: Kenarları Algılanmış Fotoğraf



Şekil 9.4: Y-Eksenine İzdüşüm Grafiği

9.6 Sonuç

Şirketler, ürünlerinin uluslararası sınırları güvenli bir şekilde geçmesini sağlamak için, kendi istekleri doğrultusunda veya zorunlu kanuni düzenlemelere uymaları gerektiğinden pek çok önlem almaktadırlar. Bununla birlikte, doğal afetler, ürünlerin kırılganlığı, bürokratik süreçler ve diğer kontrol dışı pek çok öngörülemeyen olay, firmaların tedarik zincirlerinin kırılganlığını farketmelerini sağlayarak, bu tip olayların risklerini azaltma ve tedarik zincirlerinin sürekliliğini sağlama konusunda çalışma yapmalarına yönlendirmektedir.

Çalışma Brezilya, Rusya, Kuzey Amerika ve Çin lokasyonlarına V36X Transit ve Cargo araçlarına ait parçaların sevk edildiği bir otomotiv konsolidasyon merkezinde gerçekleştirilmiştir. Firmanın tedarik zinciri içerisinde toplamda 132 imalatçı, 2700 kalem parça (4 müşteri, 9 sevkiyat lokasyonu), haftalık ortalama 150 konteynerlik hacim ile yurt dışı sevkiyatı ve günlük ortalama 30 konteynerlik milkrun seferi bulunmaktadır. Güvenli tedarik zinciri kapsamında çalışma:

- Müşteri üretimini zamanında ve eksiksiz destekleyerek, müşteri memnuniyetini arttırmak
- Eksik sevkiyat kaynaklı oluşabilecek uçak / özel araç maliyetlerinin önüne geçmek
- Eksik/Fazla sevkiyat adetlerinden dolayı, olası gümrük problemlerinin bertaraf etmek
- Müşteri tarafından sevkiyat performans değerlendirmelerimizin bu tarz uygunsuzluklardan dolayı düşürmemek
- Güvenli Tedarik Zinciri kapsamında hatalı parça sevkiyatında oluşabilecek risklerin elemine edilmesi amaçlarını sağlamak için yapılmıştır.

Gönderilen kargo güvenliğini sağlamak amacıyla müşteriye gönderilen sevkiyatın tartımı ile siparişte herhangi bir uygunsuzluk bulunup bulunmadığını tahmin etmek öncelikli hedeftir. Parça ağırlığı, palet ağırlığı, sarf malzeme ağırlığı, köşebent, vcı, strap, baloncuklu naylon, karton kutu ağırlığı sevkiyat ağırlığını oluşturan faktörler olarak belirlenmiştir. Bahsedilen faktörler arasındaki en yüksek değişkenlik palet ağırlığına aittir. Bu nedenle çalışmada görüntü işleme teknikleri kullanılarak

değişik boyuttaki paletlerin çıta ve takoz kalınlıkları belirlenerek, sevkiyat birimindeki en yüksek değişkenliğe sahip palet ağırlıklarının doğru tahmin edilmesi sağlanmıştır. Bununla beraber firmadaki güvenli tedarik zinciri kapsamında olası kayıpların önceden öngörülerek, firmanın tedarik zinciri yapısı daha esnek hale getirilmiştir.

9.7 Kaynakça

Atmaca, H. E., & Turgut, D. (2015). Kargo Şirketi Seçimine Yönelik Kriterlerin Belirlenmesinde Türkiye Genelinde Bir Saha Araştırması, Çukurova Üniversitesi İİBF Dergisi Cilt:19. Sayı:2 ss.65-79.

Manners-Bell, J. (2014). Supply Chain Risk: Understanding Emerging Threats To Global Supply Chains. Yayınevi: Koganpage. United Kingdom.

Gupta,S. (2015). Study on the Concept of Supply Chain Security, Global Journal For Research Analysis, Cilt:4, Sayı:7 ss. 92-94.

Sobel, I., & Feldman, G. (1968). A 3 Isotropic Gradient Operator For Image Processing. Presented at a talk at the Stanford Art ificial Project.

Çelik , H.İ., Dülger , L.C. and Topalbekiroğlu, M. (2012). Görüntü İşleme Teknikleri Kullanarak Kumaş Hatalarının Belirlenmesi, Electronic Journal of Textile Technologies Cilt: 6, Sayı: 1, 22-39.

Figali, N., Cihan, A. Esen, H., Figlali, A., Çeşmeci, D., Güllü, M.K. and Yılmaz, M.K. (2015). Image processing-aided working posture analysis: I-OWAS, Computer & Industrial Engineering, Cilt:85, 384-394.

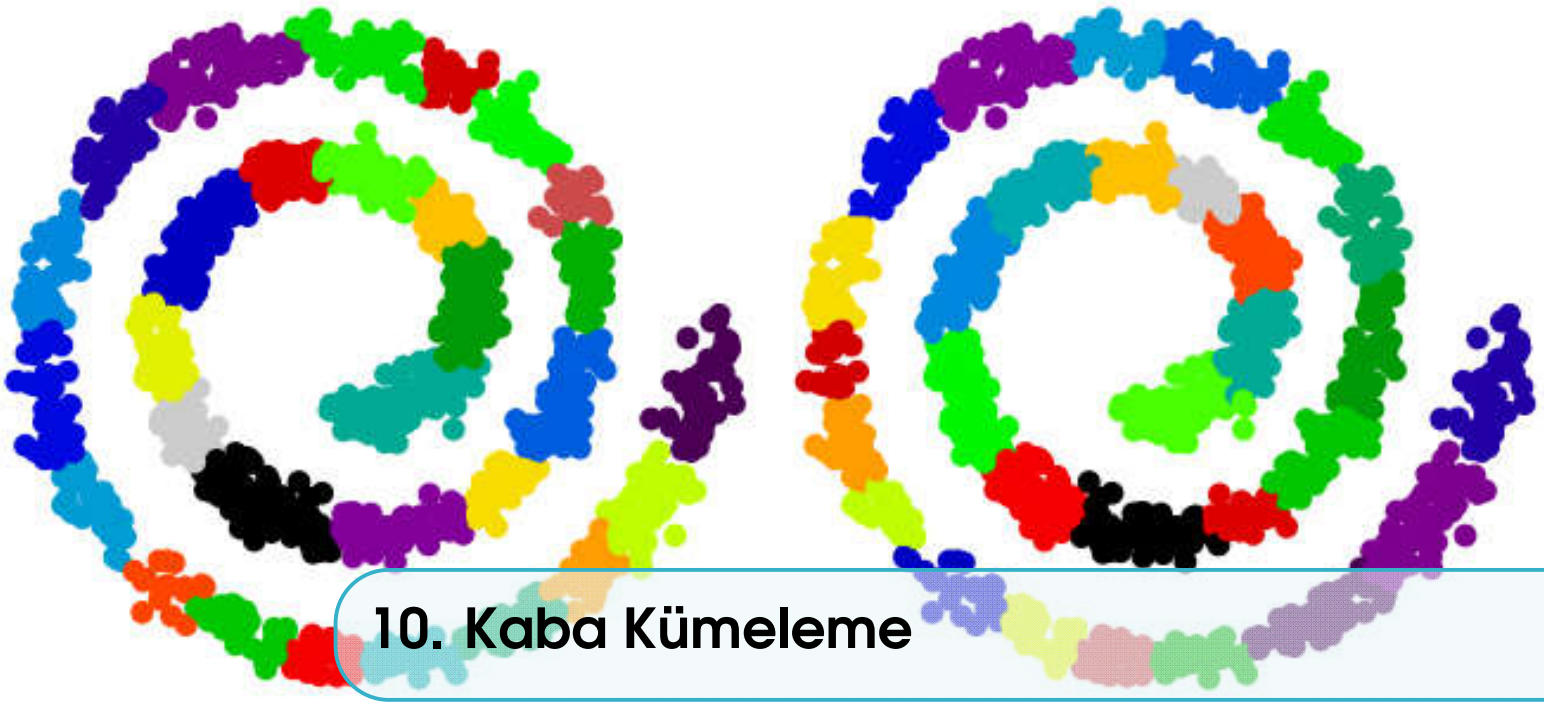
Kelong, T., Yuqing, W., Lin, Y., Riping, Z., Wei, C. and Yahobao, M. (2008). A New Archaeological Remote Sensing Technology, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Cilt: 37(B7), 221-224.

9.8 Yazarlar Hakkında

Tuğçen HATİPOĞLU, lisans eğitimini 2007 yılında Kocaeli Üniversitesi Endüstri Mühendisliği bölümünde tamamladıktan sonra, 2010 yılında Sakarya Üniversitesi Endüstri Mühendisliği'nde Yüksek Lisans eğitimini tamamlamıştır. 2009-2011 yılları arasında Sakarya Üniversitesi Uzaktan Eğitim bölümünde Araştırma Görevlisi olarak çalışmıştır. 2011 yılından itibaren ise Kocaeli Üniversitesi Endüstri Mühendisliği bölümünde Araştırma Görevlisi olarak çalışmakta, yine aynı kurumda doktora eğitimine devam etmektedir. Çalışma alanları Yapay Zeka, İstatistiksel Karar Destek Sistemleri ve Yöneylem Araştırması'dır.

Tevfik ALTINALEV, 1973 yılında Bakırköy/İSTANBUL'da doğdu. İlk ve orta öğrenimini Bakırköy'de tamamladıktan sonra, 1987 yılında Deniz Lisesi'ne başladı. 1991 yılında Deniz Lisesi'ndeki öğrenimini müteakip, 1995 yılında deniz subayı olarak Deniz Harp Okulu Elektronik Mühendisliği bölümünden mezun oldu. 1999-2001 yılları arasında Orta Doğu Teknik Üniversitesi'nde Modelleme ve Simülasyon yüksek lisans eğitimini tamamladı. 2001-2013 yılları arasında Deniz Harp Okulu Endüstri Mühendisliği Bölümü'nde öğretim elemanı olarak çalıştı. 2015 yılından günümüze kadar Kocaeli Üniversitesi Endüstri Mühendisliği Doktora programında öğrenim görmektedir.

Alpaslan FIĞLALI, Lisans, Yüksek Lisans ve Doktora derecelerini sırasıyla 1983, 1985 ve 1992 yıllarında İTÜ'den aldı. 1983-1987 yılları arasında TÜBİTAK MAE-MESAB'da Kısmi Zamanlı Araştırıcı ve Araştırıcı olarak, 1987-2003 yılları arasında İTÜ Endüstri Mühendisliği Bölümü'nde Araştırma Görevlisi, Yardımcı Doçent ve Doçent olarak çalıştı. 2003'ten bu yana Kocaeli Üniversitesi Endüstri Mühendisliği Bölümü'nde Profesör olarak görev yapmaktadır. Yöneylem Araştırması, Metasezgisel Optimizasyon ve Lojistik Yönetimi alanlarında çalışmaktadır.



10. Kaba Kümeleme

Kaba Kümeleme

Yrd. Doç. Dr. Safiye TURGAY Prof.Dr. Orhan TORKUL

10.1 Giriş

Hızla artan veri trafiğinde, mevcut verilerin verimli bir şekilde kullanılması da beraberinde önemli bir durumu ortaya çıkartmaktadır. Veritabanlarındaki bu verileri kullanmak ve yönetsel kararlar aşamasında değerlendirmek bilgi keşfi ve veri madenciliği adını verdiğimiz alanların ortaya çıkmasını zorunlu hale getirmiştir. 1982 yılında ilk olarak Pawlak tarafından önerilen bu kavram, büyük veri tabanlarını kullanarak gerekli olan bilginin keşfini sağlayan önemli bir araç olarak kullanılmaktadır. Kaba kümeler kavramı, veri madenciliği teknikleri ile birlikte kesin olmayan yapıların analizini gerçekleştirmek üzere bulanık mantık yaklaşımından türetilmiştir. Kaba kümeleme teorisi aynı zamanda kural indirgeme ve sınıflandırma konuları ile birlikte özellikle büyük veri tabanlarının analizi işlemini başarılı bir şekilde günlük hayatta uygulanabilmesine olanak sağlamaktadır.

Kaba küme teorisi, bulanık kümeler gibi kesin sınırlamaları içermeyen bir yapıya sahiptir. Dolayısıyla, eksik, yetersiz ve belirsiz bilgileri düzenleyerek veri analizine uygun hale getiren kaba küme yaklaşımı, bu işlem esnasında, eksik bilgi muhakemesi, bilgi tabanı indirgemesi, veri madenciliği ve kural keşfi yöntemlerini kullanmaktadır. Kaba küme teorisi, eğitim verileri üzerindeki yararlı bilginin çıkarımında, veri madenciliği ve veri tabanındaki çalışmalarda oldukça etkin kullanıldığı görülmektedir. Daha çok veri ve veritabanı yönetiminde kullanılabilmesi, bulanık kaba kümeleme kavramının önemini ortaya çıkartmaktadır. Kaba küme teorisi kural çıkarımı ve sınıflandırma konusunda temel problem olan tutarsız ve eksik bilgi ile uğraşan doğal bir yöntem olarak, veri kümesi boyutu ile ilgilenmeden direk olarak kural çıkarsama durumlarını gerçekleştirir. Bu yöntemde kural çıkarımı, yaklaşımlar yoluyla gerçekleşir. Gerçek hayat uygulamalarında nicel-sayısal verilerden oluşan eğitim verileri bu algoritmada da öğrenme süreci için kullanılmaktadır ve öğrenme işleminden sonra ise kural çıkarma süreci gerçekleştirilmektedir.

Bu kitap bölümünde kaba kümeleme teorisine ait temel kavramlar; kaba küme tabanlı bilgi keşfi ve veri madenciliği, kaba kümelemede endüktif öğrenme yapısı uygulamalı olarak ele alınmıştır.

10.2 Literatür Taraması

1982 yılında ilk olarak Pawlak tarafından önerilen kaba kümeler, kesin olmayan ve tamamlanmamış verilerden bilgi elde etmek için kullanılan bir matematiksel yöntem olmakla birlikte büyük veri tabanlarından bilgi keşfini sağlayan önemli bir araç olarak kullanılmaktadır(Pawlak, 1991, 1995). Kaba küme teorisi genellikle; verilerin indirgenmesi, bağımlılıklarının keşfi, verilerin öneminin tahmini, verilerden karar (kontrol) algoritmalarının oluşturulması, verilerin yaklaşık sınıflaması, verilerdeki benzerlik ve farklılıkların keşfi, verilerdeki örüntülerin keşfi, neden-sonuç ilişkilerinin bulunmasını kapsayan süreçleri içermektedir (Pawlak ve Slowinski, 1994; Aydoğan ve Gencer, 2007).

Geniş bir uygulama alanı olan kaba kümeler, tıptan finansa kadar geniş bir uygulama alanı bulmakla birlikte yapay zeka tekniklerinden çelişki çözümlemesi, örüntü algılaması, resim çözümlemesi, özellik çıkarımı, sınıflandırma ve kural indirgeme, makine öğrenmesi tekniklerine uygulanabilmektedir. Bu tekniklerin literatürdeki uygulama biçimlerine örnekler ise, tıp alanında Wang ve arkadaşları (2011), uyku hastalığının teşhisinde kullanılan faktörleri, kaba kümeler ile değerlendirerek bir teşhis sistemi geliştirmişlerdir. Son ve arkadaşları (2012), erken kalp krizi teşhisi için kaba kümeler yardımı ile karar verme modeli geliştirmişlerdir. Hassanian (2007), göğüs kanserinin teşhisinde bulanık kaba küme yaklaşımını kullanarak bütünleşik şema geliştirmişlerdir. Finansal yönden ise, Chen ve Cheng (2012), değişen Pazar koşulları ve çevre analizinde bulanık küme teorisini kullanmışlardır. Beynon ve Deel (2001) şirketlerin iflas tahminini kaba küme teorisi yardımıyla geliştirmişlerdir. Yao ve Herbert, finansal zaman serilerinin analizinde kaba kümeleri kullanmışlardır. Chen ve Cheng (2013) dünya çapında banka endüstrisi için kredi derecelendirmesinde kaba kümeler yardımı ile hibrid modeller ile karar verme mekanizması geliştirmişlerdir.

Yapay zekânın bir alt dalı olan bulanık mantık içerisinde, kaba kümelerin verilerin değerlendirilmesinde ayrı bir yeri vardır. Çelişki çözümlemede, alternatif kural azaltma metotları dikkate alınarak hedef durumun analiz edilmesinde Huang ve arkadaşları (2013) kaba kümeleri kullanmışlardır. Xu ve Kilgour (2013) renkli grafiklerin hibrid atamalarında, çelişki çözümlemesi ile bulanık kaba küme yaklaşımını tercih etmişlerdir. Kaba kümeler ile resim çözümleme uygulamalarına örnek ise, Ramana ve arkadaşları (2011) de yaklaşık küme teorisi yaklaşımı ile görsel resim benzerliğini analiz eden bir yapı önermişlerdir. Özellik çıkarımına örnek olarak, Salamo ve Lopez-Sanchez (2011) durum tabanlı çıkarsama sınıflandırıcısı için kaba küme teorisini kullanarak özellik çıkarsama işlemini gerçekleştirmiştir. Turgay ve arkadaşları (2017) kaba küme kavramını guguk kuşu algoritması ile birlikte modellemişlerdir. Derrac ve arkadaşları (2012) bulanık kaba küme teorisini kullanarak özellik seçiminde anlık seçme sürecini kuvvetlendirici bir algoritma ile önermişlerdir.

Kaba küme teorisi aynı zamanda bazı sezgisel algoritmalar ile birlikte kullanıldığı literatürde görülmektedir. Bunlardan bazıları yasak arama ile birlikte özellik seçimi ve kredi seçimi, sürü optimizasyonu ile birlikte özellik seçimi Changseok B, Wei-Chang Y, Yuk Ying C, Sin-Long L. 2010, karınca kolonisi ve kaba kümeler ile birlikte optimizasyonu (Yumin C, Duoqian M, Ruizhi W. 2010) olarak ele alınmıştır.

10.3 Kaba Kümeler

Bulanık kaba küme kavramında, nicel-sayısal değere sahip veri kümelerinin, üyelik fonksiyonu ve sözel (dilsel) terimlerle gösterimindeki yalınlığı ve anlaşılırlığı ile insanların muhakeme yeteneğine olan benzerliğinden dolayı tercih edilmektedir. Bunun yanısıra analiz sürecindeki verilerin esnek yapıda analiz edilmesi ve eksik bilginin insan muhakeme yapısını örnek alınarak değerlendirmesinden

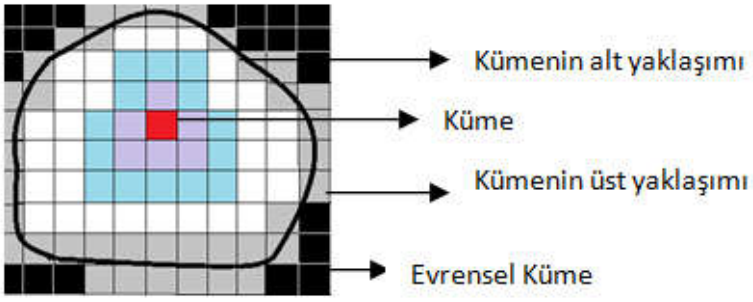
dolayı büyük veri analizinde kullanılabilmektedir. Kaba küme teorisi, hatasız bilginin, yüksek doğruluk derecesi ile tercih edilmesi amacıyla bulanık kümelerin tamamlayıcı bir özelliği olarak ortaya çıkmıştır. Uygulamalarda da görülebileceği gibi veri ve veritabanı uygulamaları sonucunda oldukça gerçekçi sonuçların elde edilmesine olanak sağlamaktadır. Kaba küme analizinin asıl amacı, kavramların yaklaşık değerlerinin analiz edilmesi ve indirgenmesi sürecinde sisteme ait karar kurallarının elde edilmesidir.

Kaba küme teorisinde veriler, nitelik ve öznitelik değerlerinin gösterildiği bir tablo halinde gösterilmesi, eğitim verilerinin belirli kriterler doğrultusunda bölümlere ayrılması ve eşitlik sınıfının kullanılması sürecidir. Öğrenme işleminde, düşük (alt) ve yüksek (üst) yaklaşım olmak üzere iki ayrı bölgede analiz işlemi gerçekleştirilir. Kaba küme teorisinin temelini oluşturan bu kavramlardan düşük yaklaşım ile kesin kurallar elde edilirken, yüksek (üst) yaklaşım yardımıyla mümkün olabilecek olası kurallar elde edilmektedir. Kaba küme teorisi ile geniş veri kümelerinin bozunması, dağıtılmış ve çok etmenli sistemlerde veri madenciliği ve parçasal veri hesaplama için yeni yaklaşımlar geliştirilmiştir. Kaba kümelemeye ait temel kavramlar ise

- Bilgi: Karar Sistemleri (Tablolar)
- Farkındalık
- Yaklaşık Kümeleme
- İndirgemeler ve Çekirdek Kavramı
- Kaba Üyelik
- Öznitelik Bağımlılğı

Kaba küme teorisi aynı zamanda kural indirgeme ve sınıflandırma kavramlarını, pratikte başarılı bir şekilde uygulanabilmesine olanak sağlamaktadır.

Kural indirgeme yaklaşımının en önemli eksikliğinden birisi ise, algoritma değerlendirme esnasında tüm kuralların hepsini birden kural indirgeme olarak kabul eder. 2002'de Jia-Yuarn Guo ve Vira Chankong tarafından geliştirilen Kural İndirgeme Algoritması daha gelişmiş ve etkin sürümü, kaba küme yaklaşımı ile bir önceki algoritmadaki eksikliği kapatmıştır.



Şekil 10.1: Kaba kümenin şematik gösterimi

Kaba kümeler, veri indirgeme, nitelik indirgeme, verideki gizli ilişkilerin belirlenmesi ve karar kurallarına dönüştürülmesi süreci ile ilgilenmektedir. Özellikle makine öğrenmesi, karar analizi, veri tabanlarından bilginin elde edilerek kullanılması, uzman sistemler, karar destek sistemleri

ve örüntü tanıma gibi alanlarda geniş bir kullanım alanına sahiptir. Yaklaşım, özellikle verilerin değerlendirilmesi ve analizi sürecinde, bulanık mantığın temel uygulamasında olduğu gibi, üyelik fonksiyonu değerlerine ihtiyaç duymaz ve istatistiksel açıdan da olasılık dağılıma uygunluk koşulunu aramaz. Kaba küme teorisi alt ve üst yaklaşımları kullanarak, aşağıda yer alan işlem basamakları şle karar kurallarının elde edilmesi sürecini gerçekleştirmektedir: (Pawlak vd., 1995; Jaaman vd., 2009;Yao, 1998; Zhai vd., 2009b)

1. Bilgi sisteminin oluşturulması
2. Ayırt edilmezlik ilişkilerinin belirlenmesi
3. İndirgemelerin yapılması
4. Karar kurallarının oluşturulması
5. Yeni nesnelerin sınıflandırılması.

10.3.1 Bilgi Sistemi ve Karar Tablosu

Bilgi sisteminde yer alan verilerin temsil edilmesi ve bu verilerin gösterim biçimindeki özellik, kaba küme teorisinin küme özellikleri ile alt değer ve üst değer yaklaşımları büyük önem arz etmektedir. Bilgi sisteminin model yapısı ile gösterimi, karar özelliğinin bilgi sistemi içerisinde temsil edilmesi detaylı olarak verilmiştir. Bilgi sistemi, satırlarda birimlerin (örneklerin) ve sütunlarda birimlere ilişkin niteliklerin yer aldığı bir tablodur.

Bilgi Sistemleri/Tablolar

Bilgi sistemi S ile ifade edilir ve $S =< U, A, V >$ formunda gösterilir. Bu formdaki U ve açılımı $U = x_1, x_2, ..., x_n$, nesnelerin boş olmayan sonlu birimler kümesini, A sonlu, boş olmayan değerler kümesini yani koşullu nitelikler kümesini ve V ise her bir niteliğin alabildiği değerler kümesini ifade eder.

$a : U \rightarrow V_a$ $a \in A$ 'dır. V_a ise a'nın değer kümesi olarak adlandırılmaktadır.

Karar Sistemleri/Tablolar

$DS : T = (U, A \cup \{d\})$ $d \notin A$ ile daha çok karar özelliği yerine karar değeri dikkate alınır. A'nın elemanları ise koşul özellikleri olarak adlandırılmaktadır. Tablo 1 'de görüldüğü gibi bazı veriler aynı olabilmekte veya birkaç uyumsuz ya da anlamsız nesneler bir arada temsil edebilir. Bununla birlikte bazı özellikler de gereksiz olabilir.

Tablo 10.1: Karar sistemine ait Tablo gösterimi

	Z1	**Z2**
x1	2	20
x2	2	0
x3	4	5
x4	4	5
x5	6	10
x6	2	10
x7	6	10

10.3.2 Eşdeğerlik (Denklik) Farkındalık İlişkisinin Belirlenmesi

Bu kısımda ayırd edici matrisin oluşturulmasına yönelik olarak gerekli olan işlem basamakları ve özelliklere yer verilmiştir. Bu özellikler içerisinde eşdeğerlik-denklik ilişkisi kavramı, küme yaklaşımı, alt değer ve üst değer yaklaşımları ve kaba küme teorisinin özellikleri ele alınmıştır. Bilgi sistemi matris formatında gösterilmiştir. S bilgi sisteminde A durum özellikleri için ayırd edilebilir matris formunda gösterimi ve analiz işlemlerinin gerçekleştirilmesi detaylı olarak ele alınmıştır. Matris formunda gösterim biçimi ise

$M(A) = (m_{ij})_{nxn}$ ve $M(A) = \begin{cases} \Phi \\ \{a \in A : a(x_i) \neq a(x_j)\} \end{cases}$ 'dir.

$M(A)$ ayırt edilebilirlik matris simetrik özelliğine sahiptir. M(A)'nın her elemanı xi ve xj değerlerini farklı niteleyen özellikler kümesinden oluşmaktadır. Bununla birlikte matris formundaki yapı aynı zamanda eşdeğerlik(denklik) ilişkisi, küme yaklaşımı, alt değer ve üst değer yaklaşımları ile kaba küme teorisinin özelliklerinin detaylı bir biçimde tanımlanması ve daha sonra matris formunda düzenleme biçimleri ele alınarak incelenmiştir. Ayırt edici matris, birimlerin ikili karşılaştırmaları sonucunda hangi nitelikler bakımından farklı değerler aldığını belirler. Belirlenen nitelikler, matriste ait olduğu hücreye yerleştirilir. Böylece ayırt edici matris oluşturulur.

Eşdeğerlik (Denklik) İlişkisi:

Matris formunda (xR_x, herhangi bir nesne için) ikili bir ilişki, $x \in X$, $y \in X$ simetrik (xR_y ise yR_x'dir) ve geçiş durumunu kapsamaktadır (eğer xR_y ve yR_z, ise xR_z'dir). Bir elemanın denklik denklik sınıfı xR_y gibi tüm nesneleri kapsaması durumudur. $[x]_R$ durumu için $R \subseteq X \times X$'dir. IS = (U, A) bir bilgi sistemini temsil etsin. Bu $B \subseteq A$ herhangi bir eşdeğerlik ilişki durumu için: $(x,x') \in IND_{IS}(B)$ için $IND_{IS}(B) = \{(x,x') \in U^2 | \forall a \in B, a(x) = a(x')\}$'dir ve $IND_{IS}(B)$ ye B-anlaşılamazlık ilişkisi denir.

Eğer x ve x', B'de yer alan niteliklerle ayırt edilemezlerse, B-denklik ilişkisinin eşdeğerlik sınıfları $[x]_R$ ile ifade edilir.

Eşdeğerlik yani denklik ilişkisi, evrensel kümenin bölünmesi yani benzer değerler ile benzer olmayan değerlerin farklı iki sınıfta temsil edilmesi ile iki farklı gruba bölünmesi durumunu ifade eder. Bu kapsam içerisinde, bölümler aynı zamanda evrenin yeni alt bölümlerini oluşturmak için kullanılmaktadır. En fazla öneme sahip altkümeler ise, karar özniteliğinde olan değer ile aynı değere sahiptir. Bununla birlikte, Örnek 1'de de görüldüğü gibi "Z3" özelliğinin net bir şekilde tanımlanamaması durumundan dolayı kaba küme yaklaşımından faydalanılmıştır.

Örnek 1: Kaba kümelemede eşdeğerlik denklik durumuna ait örnek ise;

Tablo 10.2: Eş değerlik denklik sınıfı örneğine ait Tablo gösterimi

	Z1	Z2	Z3
x1	2	20	1
x2	2	0	2
x3	4	5	2
x4	4	5	1
x5	6	10	2
x6	2	10	1
x7	6	10	2

– Koşul özniteliklerinin boş olmayan alt kümeleri Z1, Z2 ve Z1, Z2 'dir.

– IND(Z1) = x1,x2,x6, x3,x4, x5,x7

– IND(Z2) = x1, x2, x3,x4, x5,x6,x7

– IND(Z1, Z2) = x1, x2, x3,x4, x5,x7, x6.

10.3.3 Yaklaşık Kümeleme Yaklaşımı

Yaklaşık kümeleme yaklaşımı ile kaba kümeleme sürecinde gerçekleştirilen alt değer ve üst değer yaklaşımlarının özellikleri ile kaba üyelik kavramlarına detaylı olarak yer verilmiştir.

Kümeleme Yaklaşımı

$T = (U, A)$ ile $B \subseteq A$ ve $X \subseteq U$ durumlarını ele aldığımızda, sadece x'i kullanarak $\underline{B}X$ ve $\overline{B}X$ ile B'nin alt ve üst değerlerini kullanarak bilgi türetebiliriz. Türetilen bu bilginin gösterim biçimi ise,

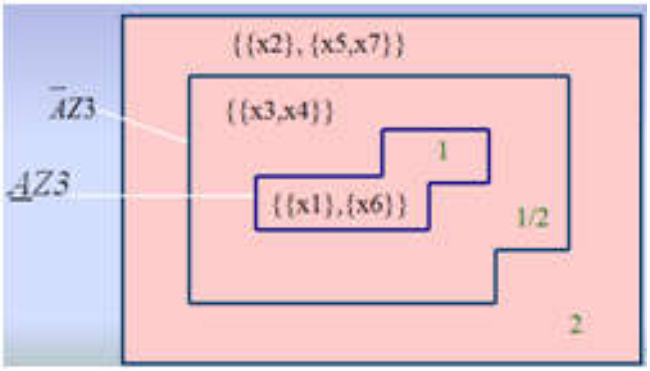
$\underline{B}X = \{x | [x]_B \subseteq X\}$, $\overline{B}X = \{x | [x]_B \cap X \neq \varphi\}$

B-sınır bölgesi, X elemanlarından oluşmaktadır ve $BN_B(X) = \overline{B}X - \underline{B}X$ ile ifade edilir $U - \overline{B}X$ X'in B 'de yer alan durumu tamamen sınıflandırılamamıştır yani tanımlanamamıştır. B-bölgesi dışındaki alan ise bu bölgede yer alan değerler, belirli olarak sınıflandırılabilir fakat X değerine ait değildirler. Kaba küme değeri herhangi bir değere sahip ise sınırlı bölgede yer alabilir yani kesin net (crsip) değer ile ifade edilebilir. Küme yaklaşımı Örnek 2'de uygulamalı olarak gösterilmiştir.

Örnek 2: Tablo 3.2'de yer alan veri kümesine kaba küme yaklaşımı uygulanmış ve aşağıda yer alan sonuçlar elde edilmiştir. Let Z3 = x | Z3(x) = 1.

$\underline{A}Z3 = \{x1, x6\}, \overline{A}Z3 = \{x1, x3, x4, x6\}, BN_A(Z3) = \{x3, x4\}, U - \overline{A}Z3 = \{x2, x5, x7\}$

Sınır bölgesi boş olmadığı müddetçe karar sınıfı Z3, kaba'dır. Kümesel olarak gösterimi biçimine Şekil 3.2'de yer verilmiştir.



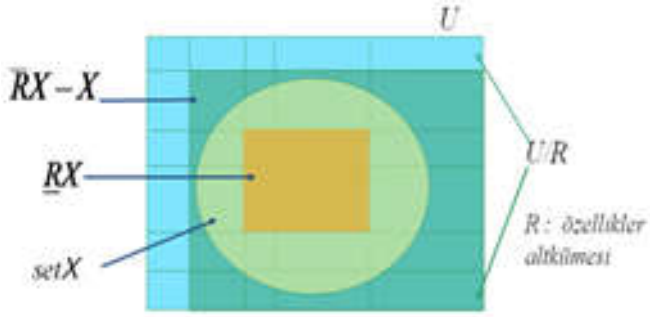
Şekil 10.2: Örnek 2'nin R özellikler alt kümesi ile gösterimi

Alt Değer ve Üst Değer Yaklaşımları

Alt değer ve üst değer yaklaşımları kaba kümelemenin esnek bir yapıda verilerin analizi esnasında sınıflandırılması sürecine olanak sağlayan bir yöntemdir. Şekil 2'de örnek 2'ye ait çözüm tablosunun R özellikler kümesi ile birlikte gösterimi yer almaktadır. Şekil 3 ise U evrensel kümesi ile birlikte alt değer ve üst değer yaklaşımlarının R özellikler alt kümesini göstermektedir.

Üst Değer Yaklaşımı:

$\overline{R}X = \bigcup \{Y \in U/R : Y \cap X \neq \varphi\}$



Şekil 10.3: Alt Değer ve Üst Değer yaklaşımlarının R özellikler alt kümesi ile gösterimi

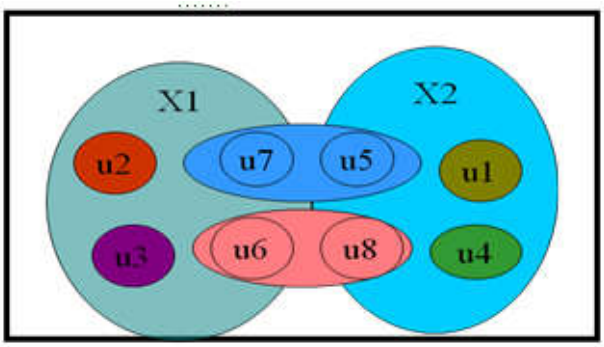
Alt Değer Yaklaşımı:
$\underline{R}X = \bigcup\{Y \in U/R : Y \subseteq X\}$ biçimindedir.
Alt değer ve üst değer yaklaşımlarına ait örnek ise

Tablo 10.3: Alt değer ve üst değer yaklaşımı için Tablo gösterimi

U	Headache	Temp.	Flu
U1	Yes	Normal	No
U2	Yes	High	Yes
U3	Yes	Very-High	Yes
U4	No	Normal	No
U5	No	High	No
U6	No	Very-High	Yes
U7	No	High	Yes
U8	No	Very-High	No

Eşdeğerlik, denklik sınıfı tanımlandığında ise R = Headache, Temp. ye ait veri dağılımları {u1}, {u2}, {u3}, {u4}, {u5, u7}, {u6, u8}. Alt değer ve üst değer yaklaşımlarının grafiksel gösterimine Şekil 3.4'de yer verilmiştir.

$X1 = \{u|Flu(u) = yes\}$
$= \{u2, u3, u6, u7\}$
$RX1 = \{u2, u3\}$
$= \{u2, u3, u6, u7, u8, u5\}$
$X2 = \{u|Flu(u) = no\}$
$= \{u1, u4, u5, u8\}$
$RX2 = \{u1, u4\}$
$= \{u1, u4, u5, u8, u7, u6\}$
$R = Headache, Temp.$
$U/R = \{\{u1\}, \{u2\}, \{u3\}, \{u4\}, \{u5, u7\}, \{u6, u8\}\}$
$X1 = \{u|Flu(u) = yes\} = \{u2, u3, u6, u7\}$
$X2 = \{u|Flu(u) = no\} = \{u1, u4, u5, u8\}$



Şekil 10.4: Alt değer ve üst değer yaklaşımlarının grafiksel gösterimi (Son et al. 2012; https://www.webpages.uncc.edu/ras/KDD-02/rough-sets.PPT)

Kaba Küme Teorisinin Özellikleri

X değeri U-X evrensel kümesi içerisinde yer alması durumunda;

1. $\underline{B}(X) \subseteq X \subseteq \overline{B}X$ Kapsama özelliği(Sınır özelliği)
2. $\underline{B}(\varphi) = \overline{B}(\varphi) = \varphi$, Boş küme özelliği
3. $\underline{B}(U) = \overline{B}(U) = U$ Evrensel küme özelliği
4. $\overline{B}(X \cup Y) = \overline{B}(X) \cup \overline{B}(Y)$ Üst sınır için birleşim özelliği
5. $\underline{B}(X \cap Y) = \underline{B}(X) \cap \underline{B}(Y)$ Alt sınır için kesişim özelliği
6. Alt ve üst sınır için $X \subseteq Y$ özelliği
7. $\overline{B}(X) \subseteq \overline{B}(Y)$ ve $\underline{B}(X) \subseteq \underline{B}(Y)$ için $X \subseteq Y$
8. $\underline{B}(X \cup Y) \supseteq \underline{B}(X) \cup \underline{B}(Y)$ Alt sınır için birleşim özelliği
9. $\overline{B}(X \cap Y) \subseteq \overline{B}(X) \cap \overline{B}(Y)$ Üst sınır için kesişim özelliği
10. $\left.\begin{array}{l} \underline{B}(-X) = -\overline{B}(X) \\ \overline{B}(-X) = -\underline{B}(X) \end{array}\right\}$ Tümleyen özelliği
11. $\left.\begin{array}{l} \underline{B}(\underline{B}(X)) = \overline{B}(\underline{B}(X)) = \underline{B}(X) \\ \overline{B}(\overline{B}(X)) = \underline{B}(\overline{B}(X)) = \overline{B}(X) \end{array}\right\}$ Bütünleyen özelliği

10.3.4 İndirgemeler ve Çekirdek Kavramı

Bu bölümde vazgeçilebilir ve vazgeçilemez durumlarına ait özellikler, bağımsızlık özelliği, tanınabilirlik matrisi(pozitif bölgeye göre), nesnelere göre belirlenebilirlik özelliklerine yer verilmiştir.

İndirgenmiş nitelikler Kümesi: bilgi sistemindeki veri yapısını temsil edecek minimum nitelik kümeleridir.

- Çekirdek nitelikler kümesi: indirgenmiş nitelikler kümelerinin ortak elemanlarından oluşur.
- Benzerlik matrisi: İndirgenmiş nitelik kümeleri "benzerlik matrisi" kullanılarak bulunmaktadır. Benzerlik matrisi nxn boyutuna bir matris olup, n ise temel küme sayısıdır.

Vazgeçilebilirlik ve Vazgeçilemezlik Durumuna Ait Özellikler

$B \subseteq A$ olmak üzere, B koşullu nitelikler alt kümesi, $IND_A(B) =$ ayırt edilemezlik bağıntısını koruyorsa B kümesine ait niteliklerden vazgeçilebilir. Vazgeçilebilir nitelik içermeyen alt kümelere indirgenmiş nitelik kümeleri denir. S=(U,A) bilgi sistemi olmak üzere indirgenmiş nitelik kümesi, $IND_A(B) = IND_A(A)$ ve $IND_A(B - \{a\}^1 IND_A(A)$ olacak şekilde bir $B \subseteq A$, en küçük nitelikler

kümesidir. İndirgeme yapabilmek için ilk olarak ayırt edici matris elde edilmelidir ve ayırt edici fonksiyon bulunduktan sonra indirgeme yapılabilir. İkinci bir yol olarak ayırt edici matris ve fonksiyonu bulmadan da indirgemeler elde edilebilmektedir. Burada niteliklerin mümkün tüm kombinasyonlarına göre birimlerin her biri ayırt edilebilir olmalıdır. Bütün indirgenmiş nitelik kümelerinin kesişimi çekirdek (core) nitelik kümesini verir. Bir S bilgi sisteminin bütün indirgenmiş nitelik kümelerinin bileşim kümesi RED(A) ile gösterilir (Tseng vd., 2016).

$c \in C$ olması durumunda özellik cT'de eğer $POS_C(D) = POS_{(C-\{c\})}(D)$ durumunda ise vazgeçilebilir özelliğini taşıyor diyebiliriz. Aksi durum için vazgeçilemez özelliğini taşıyor diyebiliriz.

D bölgesindeki C-pozitif değeri ise $POS_C(D) = \bigcup_{X \in U/D} \underline{C}X$dir.

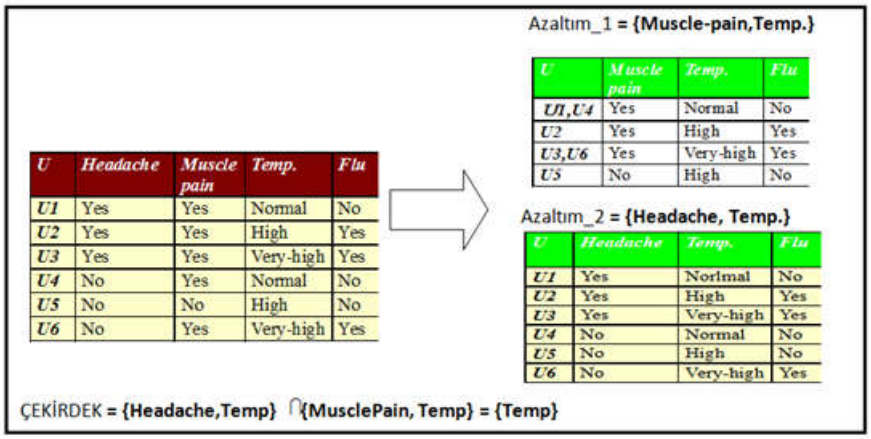
Bağımsızlık Özelliği

Eğer tüm $c \in C$'ler T'de vazgeçilmezlik özelliğini taşıyor ise T = (U, C, D) bağımsızdır.

Azaltım ve Çekirdek Yaklaşımı $POS_R(D) = POS_C(D)$

T '= (U, R, D) bağımsızdır ve özelliklerin kümesine $R \subseteq C$ koşulu ile C indirgemesi durumu koşulu ile sağlar. T'de vazgeçilemez durumuna ait tüm koşul öznitelikleri (değişkenler) ÇEKİRDEK(C) ile gösterildiğinde ise $CORE(C) = \cap RED(C)$'dir.

Azaltım ve çekirdek kullanım durumuna ait örnek şekil 3.5'de gösterilmiştir.



U	Headache	Muscle pain	Temp.	Flu
U1	Yes	Yes	Normal	No
U2	Yes	Yes	High	Yes
U3	Yes	Yes	Very-high	Yes
U4	No	Yes	Normal	No
U5	No	No	High	No
U6	No	Yes	Very-high	Yes

U	Muscle pain	Temp.	Flu
U1,U4	Yes	Normal	No
U2	Yes	High	Yes
U3,U6	Yes	Very-high	Yes
U5	No	High	No

U	Headache	Temp.	Flu
U1	Yes	Normal	No
U2	Yes	High	Yes
U3	Yes	Very-high	Yes
U4	No	Normal	No
U5	No	High	No
U6	No	Very-high	Yes

Şekil 10.5: Azaltım ve çekirdek kullanımına ait örnek (Son et al. 2012; https://www.webpages.uncc.edu/ras/KDD-02/rough-sets.PPT)

Tanınabilirlik Matrisi (pozitif bölgeye göre)

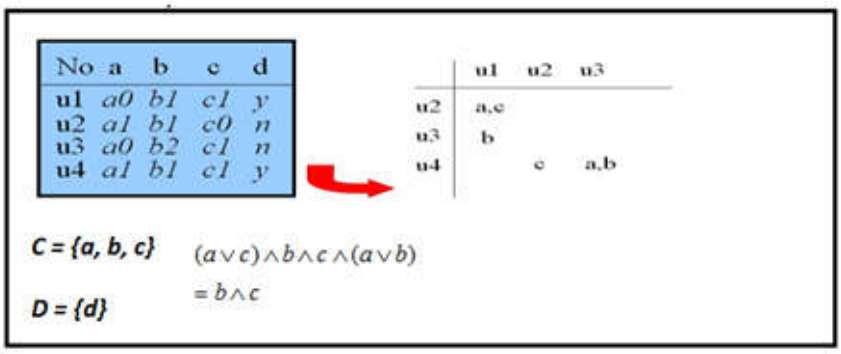
T = (U, C, D) bir karar tablosu olsun ve $U = \{u_1, u_2, ..., u_n\}$ ile birlikte T'nin ayırt edici matrisi olan M(T) ile gösterildiğinde matris şu şekilde tanımlanır:

$m_{ij} = \begin{cases} \{c \in C: c(u_i) \neq c(u_j)\} \; if \; \exists \, d \in D \; [d(u_i) \neq d(u_j)] \\ \lambda \; if \; \forall \, d \in D \; [d(u_i) = d(u_j)] \end{cases}$ i, j = 1,2, ..., n için, D'nin C-pozitif bölgesinde olması durumu ile tanımlanabilir.

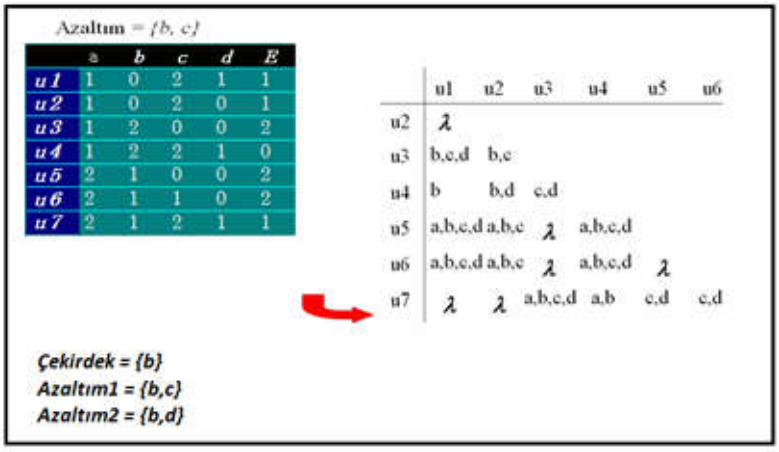
m_{ij}, u_i ve u_j'yi farklı sınıfları sınıflandıran tüm koşul öznitelikleri kapsayan nesnelerin kümesi olarak göstermektedir. T'nin ayırt edici matrisi olan, $M(T)$ ile gösterildiğinde, m_{ij} denklem yapısı ile i, j matrisindeki indislere denk gelen $M(T)$'nin boş olmayan tüm değerler kümesi yani girdileri dikkate alınır. u_i yada u_j bölgesindeki C-pozitif değerlerine aittir. $m_{ij} = \lambda d$ ise boş olan değerler kümesinde dikkate alınabileceğini varsayarak, sistemin mantıksal gerçeklik durumunu göstermiş

olmaktadır. Minimal bölünebilirlik durumu ile bu fonksiyonun T indirgeme durumları pozitif bölgeye göre tanımlanmış olmaktadır.

Örnek 3: Tanınabilirlik matrisini d karar özelliğine ait eşdeğerlik sınıflarını ayırt edebilmek için, Şekil 6'daki tabloda hazırlanan belirlenebilirlik matrisi tarafından tanımlanan koşulları korumak için kullanmaktayız.



Şekil 10.6: Tanımlanabilirlik matrisinin kullanımı (Son et al. 2012; https://www.webpages.uncc.edu/ras/KDD-02/rough-sets.PPT)



Şekil 10.7: Azaltım ve Çekirdek fonksiyonlarının kullanımı (Son et al. 2012; https://www.webpages.uncc.edu/ras/KDD-02/rough-sets.PPT)

Nesnelere göre Belirlenebilirlik İşlevi

Herhangi bir $u_i \in U$ için, $f_T(u_i) = \bigwedge_j \{\vee m_{ij} : j \neq i, j \in \{1,2,...,n\}\}$ minimum ayırıcı normal formdaki her bir mantıksal yapı, indirgeme örneğinde olduğu gibi açıklanabilmektedir.

Nesnelere göre belirlenebilirlik işlevine ait özellikler ise;

1. $m_{ij} \neq \varphi$ koşulu ile $\vee m_{ij}$ tüm değişkenlerin birleşimi olup, $a \in m_{ij}$ dir.
2. $m_{ij} = \varphi$ koşulu ile $\vee m_{ij} = \perp(false)$ dir.
3. $m_{ij} = \lambda$ koşulu ile $\vee m_{ij} = \perp(true)$ dir.

10.3.5 Kaba Üyelik

Kaba üyelik fonksiyonu, X kümesine ait x'in eşdeğerlik sınıfı arasındaki çakışma derecesi $[x]_B$ fonksiyonu ile nitelendirilir.

$\mu_X^B : U \to [0,1]$, $\mu_X^B(x) = \frac{|[x]_B \cap X|}{|[x]_B|}$

Kaba üyelik fonksiyonu, frekans(tekrar) temelli bir tahminci $P(x \in X|u)$ olarak yorumlanabilir. Burada yer alan u, IND (B) eşdeğer sınıfında kullanılmaktadır. Alt ve üst yaklaşımlar için ele alınan formüller, rastgele hassaslık seviyesi için genelleştirilerek elde edilen kaba üyelik fonksiyonu ile gösterilebilir.

$$\begin{matrix} B_{\pi} X = \{x|\mu_X^B(x) \geq \pi\} \\ \bar{B}_{\pi}X = \{x|\mu_X^B(x) > 1-\pi\} \end{matrix}, \pi \in (0.5,\ 1]$$

10.3.6 Öznitelik Bağımlılığı

Öznitelikler arasındaki bağımlılıkları bulmak bilgi keşfi ve veri madenciliği çalışmaları için önemli bir konudur. D öznitelik kümesi tamamen C özniteliklerinin tüm değerlerine bağlı olarak C'deki tüm değerlerin $C \Rightarrow D$, biçiminde ele alınması durumudur.

Öznitelik bağımlılığı, D ve C, A' nın alt kümesi olarak tanımlanması durumunda;

D, C'ye bağlı olarak k derecesinde $(0 \leq k \leq 1)$,

$C \Rightarrow_k D$, eğer $k = \gamma(C,D) = \frac{|POS_C(D)|}{|U|}$

$POS_C(D) = \bigcup_{X \in U/D} \underline{C}(X)$ D bölgesindeki C-pozitif değeri olarak adlandırılmaktadır. Daha ayrıntılı olarak el alınırsa;

$$k = \gamma(C,D) = \sum_{X \in U/D} \frac{|\underline{C}(X)|}{|U|}$$

Eğer $k = 1$ ise D tamamen C'ye bağlıdır.

Eğer $k < 1$ ise D parçasal (k derecesinde) C'ye bağlıdır.

Kaba küme çalışmalarında aynı zamanda sezgisel kaba küme tabanlı özellik seçimi, kaba küme temelli ikili sayı tabanlı kural çıkarsama, genelleştirilmiş dağıtım tablosu (Generalized Distribution Table) ile kural bulma yaklaşımları literatürde yeni yer alan çalışma alanları olarak ifade edilebilmektedir.

Kaba küme çalışmalarında aynı zamanda sezgisel kaba küme tabanlı özellik seçimi, kaba küme temelli ikili sayı tabanlı kural çıkarsama, genelleştirilmiş dağıtım tablosu (Generalized Distribution Table) ile kural bulma yaklaşımları literatürde yeni yer alan çalışma alanları olarak ifade edilebilmektedir.

10.3.7 Kaba Kümelerde Teorik olarak Sınıflandırma

Kaba küme teorisi tanımsal olarak dört temel sınıf içerisinde yer alabilir, bunlar kaba küme tanımlaması, içsel olarak kaba küme tanımlaması, dışsal olarak kaba küme tanımlaması ve toplamda kaba küme tanımlaması olarak gruplandırılabilir.

– x değeri B-'de tanımlandığında $\underline{B}(X) \neq \varphi$ ve $\overline{B}(X) \neq U$ koşullarını sağladığında kaba küme olarak tanımlanabilir.

– x içsel olarak B'de tanımlanamaması durumunda, $\underline{B}(X) = \varphi$ ve $\overline{B}(X) \neq U$ içsel olarak kaba küme tanımlaması yapılabilir.

– x dışsal olarak B'de tanımlanamadığında $\underline{B}(X) \neq \varphi$ ve $\overline{B}(X) = U$ koşullarını sağladığında dışsal olarak kaba küme tanımlaması yapılabilir.

– x toplamda B'de tanımlanamadığında $\underline{B}(X) = \varphi$ ve $\overline{B}(X) = U$ koşullarında toplamda kaba küme olarak tanımlanamaz.

Alt sınır ve üst sınır yaklaşım değerlerinin doğrulaması işleminde ise;

$$\alpha_B(X) = \frac{|\underline{B}(X)|}{|\overline{B}(X)|}$$

$|X|$ değeri $X \neq \varphi$ müddetçe denenir ve $0 \leq \alpha_B \leq 1$ aralık değerlerinde,
eğer $\alpha_B(X) = 1$ ise x B yaklaşımı ile kesin net değere sahiptir.
eğer $\alpha_B(X) = 1$ ise x B yaklaşımı ile x kabadır denir.

Kaba küme analiz işlemi esnasında kullanılan tablolarda bazen bazı sorunlar ile karşılaşılabilir ki, bu sorunlar; aynı veya anlaşılmaz olan verilerin birkaç kez tekrar edilebilmesi yada bazı değişken yada özelliklerin gereksiz olabilmeleridir. Bu tür özelliğe sahip verilerin kaldırılması gerçekleştirilecek olan sınıflandırmanın kalitesini olumsuz yönde etkilemez.

Kaba kümelemede ayırt edilemezlik(indiscernibility) özelliğini taşıyan veri kümeleri kullanılarak alt sınır ve üst sınır biçiminde tanımlanır. Genellikle bu tarz veri kümelerine ait alt kümeler bulunarak saklama depolama işlemi gerçekleştirilir ve bu işlemede ayırt edilemezlik adı verilir.

10.4 Sonuç

Kaba kümeler, bilgi keşfi ve veri madenciliği için sağlam bir temel teşkil eder. Verilerde gizli olan kalıpları keşfetmek için matematiksel yöntemlerden; özellik seçimi; özellik çıkarma; veri azaltma; karar verme sürecinde kullanılmak üzere kural oluşturma ve ilişkisel kurallar oluşturmak üzere kullanılabilir. Kaba kümelemeye ait temel kavramlar ise; bilgi: karar sistemleri (tablolar); farkındalık durumu; yaklaşık kümeleme; indirgemeler ve çekirdek kavramı; kaba üyelik; öznitelik bağımlılığıdır. Bu kavramlar ve uygulama biçimlerine detaylı olarak bu bölümde yer verilmiştir. Kaba küme yaklaşımı ile verilerdeki kısmi veya tam bağlılıklar tanımlanabilir, gereksiz veriler elimine edilerek boş değerler belirlenebilir. Eksik veri, dinamik veri yapıları belirlenebilir. Kaba küme teorisi ile aynı zamanda tutarsızlık gösteren veriler ve kesin olmayan gizli çıkarımların bulunması ile aynı bulanık mantıkta olduğu gibi eksik, yetersiz ve belirsiz bilgilerin düzenlenerek analiz edilmesi ve karar kurallarının elde edilmesi yer almaktadır. Kaba küme yaklaşımında kullanılan her bir karar tablosu, aynı zamanda kurallar tabanını da oluşturmaktadır. Kurallar, "eğer ise", koşul yapıları içerisindeki sonuç ifadelerinden oluşmaktadır. Koşul, koşul niteliklerinin aldığı değerleri, sonuç ise karar niteliklerinin değerini göstermektedir. Bilgi sistemi indirgendiğinde sistemin temelini oluşturan kurallar ortaya çıkmış olur. Yeni bir nesnenin sınıflandırılmasında farklı durumlar söz konusu olabilmektedir.

Bu bölümde ele alınan kaba kümeleme yaklaşımının çözebileceği problem tipleri ve uygulama alanları ise; çok büyük veri kümeleri; karışık veri türleri (sürekli değer alan kümeler, sembolik veriler); belirsizlik içeren gürültülü veriler; eksiklik durumunu kapsayan eksik veri ve yapıları; veri değişimi durumları; eski, arşiv bilgilerinin kullanımının gerektiği durumlardır. Kaba küme teorisi

çok sayıda niteliğin meydana getirdiği, büyük veri topluluklarının analizinde sınıflandırma ve kural indirgeme yöntemleri ile karar kurallarının elde edilmesine olanak sağlamaktadır. Kaba küme tabanlı geliştirilen algoritmalar sayesinde büyük yapıdaki veritabanlarının analizi gerçekleştirilebilmektedir.

10.5 Kaynakça

Ananthanarayana, V. S., Narasimha, M. M., & Subramanian, D. K. (2003). Tree structure for efficient data mining using rough sets. Pattern Recognition Letters, 24(6), 851–862. doi:10.1016/S0167-8655(02)00197-6

Avşar, G. (2007). Extraction of Fuzzy Rules from Incomplete Data with Do not Care and Lost Value by Rough Sets (Master's Thesis). Fırat University, Graduate School of Natural and Applied Sciences, Department of Computer Engineering.

Aydoğan, E, K, 2008. Veri Madenciliğinde sınıflandırma problemleri için evrimsel algortima tabanlı yeni bir yaklaşım: Rough-Mep algoritması. GÜ Fen bilimleri (Basılmamış doktora tezi).

Beynon, M.J., Peel, M.J., '' Variable precision rough set theory and data discretion: an application to corporate failure prediction", Omega, V. 29(2001), 561-576.

Carey, G., Law, R., & Mok, H. M. K. (2008). Analyzing and Forecasting Tourism Demand: A Rough Sets Approach, Journal of Travel Research, 46(3), 327-338.

Changseok, B., Wei-Chang, Y., Yuk Ying, C., & Sin-Long, L. (2010). Feature selection with Intelligent Dynamic Swarm and Rough Set. Expert Systems with Applications, 37(10), 7026–7032. doi:10.1016/j.eswa.2010.03.016

Chen, J., Lin, Y., Li, J., Lin, G., Ma, Z., & Tan, A. (2016). A rough set method for the minimum vertex cover problem of graphs. Applied Soft Computing, 42(May), 360–367. doi:10.1016/j.asoc.2016.02.003

Chen, J., Lin, Y., Lin, G., Li, J., & Ma, Z. (2015). The relationship between attribute reducts in rough sets and minimal vertex covers of graphs. Information Sciences, 325, 87-97.

Chen, Y. S. (2012). Classifying credit ratings for Asian banks using integrating features election and the CPDA-based rough sets approach. Knowledge-Based Systems, 26, 259–270. doi:10.1016/j.knosys.2011.08.021

Derrac, J., Cornelis, C., Garcia, S., & Herrera, F. (2012). Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection'. Information Sciences, 186(1), 73–92. doi:10.1016/j.ins.2011.09.027

Fang, B. W., & Hu, B. Q. (2016). Probabilistic graded rough set and double relative quantitative decision-theoretic rough set. International Journal of Approximate Reasoning, 74(July), 1–12. doi:10.1016/j.ijar.2016.03.004

Guo, J., & Chankong, V. (2002). Rough set-based approach to rule generation and rule induction. International Journal of General Systems, 31(6), 601–617. doi:10.1080/0308107021000034353

Hu, B. (2016). Three-way decision spaces based on partially ordered sets and three- way decisions based on hesitant fuzzy sets. Knowledge- Based Systems, 91, 16–31. doi:10.1016/j.knosys.2015.09.026

Huang, B., Guo, C. X., Li, H. X., Feng, G. F., & Zhou, X. Z. (2016, September). An intuitinistic fuzzy graded covering rough set. Knowledge- Based Systems, 107(1), 155–178. doi:10.1016/j.knosys.2016.06.006

Huang, C. C. Tseng (Bill), T., & Tang, C.Y. (2016). Feature extraction using rough set theory in service sector application from incremental perspective. Computers & Industrial Engineering, 91, 30-41.

James, J. H. L., & Gwo-Hshiung, T. (2010). A Dominance-based Rough Set Approach to customer behavior in the airline market. Information Sciences, 180(11), 2230–2238. doi:10.1016/j.ins.2010.01.025

Lee, S., & Vatchsevanos, G. (2002). An application of rough set theory to defect detection of automotive glass. Mathematics and Computers in Simulation, 60(3-5), 225–231. doi:10.1016/S0378-4754(02)00017-4

Leung, Y., Fischer, M.M., Wu, W.Z., & Mi, J.S. (2008). A rough set approach for the discovery of classification rules in interval-valued information systems. International Journal of Approximate Reasoning, 47(2), 233-246.

Lin, S. H., Huang, C. C., & Che, Z. X. (2015). Rule induction for hierarchical attributes using a rough set for the selection off a green fleet. Applied Soft Computing, 37(December), 456–466. doi:10.1016/j.asoc.2015.08.016

Liu, D., Li, T., & Ruan, D. (2011). Probabilistic model criteria with decision-theoretic rough sets. Information Sciences, 181(17), 3709–3722. doi:10.1016/j.ins.2011.04.039

Macia-Perez, F., Berna-Martinez, J. V., Oliva, A. F., & Ortega, M. A. A. (2015). Algorithm for the detection of outliers based on the theory of rough sets. Decision Support Systems, 75(July), 63–75. doi:10.1016/j.dss.2015.05.002

Mehers, S.K., ''Excplicit rough-fuzzy pattern classification model", Pattern Recognition Letters, V. 36, 15, Jan. 2014, 54-61. Nauman, M., Azam, N., & Yao, J.T. (2016). A three-way decision making approach to malware analysis using probabilistic rough sets. Information Sciences, 374, 193-209.

Nina, F. R. C. (2007). " On Applications of Rough Sets Theory to Knowledge Discovery", Doctor pf Philosophy Thesis in Computing and Information Sciences and Engineering, University of Puerto Rico.

Pawlak, J., Grzymala-Busse, Z., Slowinski, R., & Ziarko, W. (1995). Rough sets. Communications of the ACM, 38(11), 89–95. doi:10.1145/219717.219791

Pawlak, Z. (1982). Rough Sets. International Journal of Computer and Information Sciences, 11(5), 341–356. doi:10.1007/BF01001956

Pawlak, Z. (1991). Rough Sets, Theoretical Aspects of Reasoning about Data. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Pawlak, Z., & Skowron, A. (1994). Rough Set Rudiments. The International Workshop on Rough Sets and Soft Computing, 72.

Pedrycz, W. (2013). Granular Computing: Analysis and Design of Intelligent Systems. Boca Raton, FL: CRC Press/Francis Taylor. doi:10.1201/ b14862

Pedrycz, W. (2014). Allocation of information granularity in optimization and decision making models: Towards building the foundations of granular computing. European Journal of Operational Research, 232(1), 137–145. doi:10.1016/j. ejor.2012.03.038

Polkowski, L., Tsumoto, S., & Lin, T. Y. (Eds.). (2010). Rough Set Methods and Applications. Berlin: Physica-Verlag.

Ramanna, S., Meghdadi, A.H., Peters, J.F., ''Nature-inspired framework for measuring visiual image resemblance: A near rough set approach", Theoritical Computer Science, V. 412, 42, September 2011, 5926-5938.

Rissino, S. Lambert-Torres (2009). "Rough Set Theory – Fundamental Concepts, Principals, Data Extraction, and Applications", Data Mining and Knowledge Discovery in Real Life Applications, Julio Ponce and Adem Karahoca, eds., Vienna, Austria: I-Tech, 438.

Salamo, M. & Loıpez-Sanchez, M., (2012). Rough set based approaches to feature selection for Case- Based Reasoning classifiers. Pattern Recognition Letters, 32(2), 280-292.

Shu, W., & Qian, W. (2015). An incremental approach to attribute reduction from dynamic incomplete decision systems in rough set theory. Data & Knowledge Engineering, 100, 116–132.

Spiric, J. V., Stankovic, S. S., Docic, M. B., & Popovic, T. D. (2014). Using the rough set theory to detect fraud committed by electricity customers. International Journal of Electrical Power & Energy Systems, 62(November), 727–734. doi:10.1016/j.ijepes.2014.05.004

Son, C.S., Kim, Y.N., Kim, H.S., Park, H.S., Kim, M.S., ''Decision-making model for early diagnosis of congestive heart failure using rough set and decision tree approaches", Journal of Biomedical Informatics, V. 45,5, 2012, 999-1008.

Sun, L., Xu, J. C., & Tian, Y. (2012). Feature selection using rough entropy-based uncertainty measures in incomplete decision systems. Knowledge- Based Systems, 36, 206–216. doi:10.1016/j.knosys.2012.06.010

Swiniarski, R. W., & Skowron, A. (2003). Rough set methods in feature selection and recognition. Pattern Recognition Letters, 24(6), 833–849. doi:10.1016/S0167-8655(02)00196-4

Tian, D., Zeng, X.J., Keane, J., '' Core-generating approximate minimum entropy discretization for rough set feature selection in pattern classification", International Journal of Approximate Reasoning, V. 92, 6, Sept. 2011, 863-880.

Tseng, T.L., Huang, C.C., Fraser, K., & Ting, H.W. (2016). Rough set based rule induction in decision making using credible classification and preference from medical application perspective. Computer Methods and Programs in Biomedicine, 127, 273-289.

Tsumoto, S. (1998). Automated extraction of medical expertsystem rules from clinical databases based on rough set theory. Information Sciences, 112(1-4), 67–84.

Turgay,S., Torkul, O., Turgay, T., Rough-Set-Based Decision Model for Incomplete Information Systems, Encyclopedia of Information Science and Technology, ed. Mehdi Khosrow-Pour, Fourth,IGI Publishing, 2017

Xu, H., Kilgour, D.M., Hipel, K.W., McBean, E.A., '' Theory and application of conflict resolution with hybrid preference in colored graphs", Applied Mathematical Modelling, V. 37, 3, 2013, 989-1003.

Wang, P.C., Su, C., T., Chen, K.H., Chen, N.H., ''The application of rough set and Mahalandois distance to enhance the quality of OSA diagnosis", Expert Systems with Applications, V. 8, 6, 2011, 7828-7836.

Wei, D., Zhao, Y., & Zhou, X. (2006). "A Rough Set Approach to Incomplete and Fuzzy Decision Information System", Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21 - 23, 2006, Dalian, China.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Levy flights. In Proc. of World Congresson Nature & Biologically Inspired Computing (pp. 210–214). IEEE Publications. doi:10.1109/NABIC.2009.5393690

Yao, J.T., Herbert, J.P., '' Financial time-series analysis with rough sets", Aüpplied Soft Computing, V. 9, 3 (2009), 1000-1007.

Yao, Y. Y. (2001). Information granulation and rough set approximation. International Journal of Intelligent Systems, 16(1), 87–104. doi:10.1002/1098-111X(200101)16:1<87::AIDINT7> 3.0.CO;2-S D Category: Decision Support Systems 2211

Yao, Y. Y. (2010). Three-way decisions with probabilistic rough sets. Information Sciences, 180(3), 341–353. doi:10.1016/j.ins.2009.09.021

Yao, Y. Y., & Zhao, Y. (2008). Attribute reduction in decision-theoretic rough set models. Information Sciences, 178(17), 3356–3373. doi:10.1016/j. ins.2008.05.010

Yumin, C., Duoqian, M., & Ruizhi, W. (2010). A rough set approach to feature selection based on ant colony optimization. Pattern Recognition Letters, 31(3), 226–233.

Zhang, Y., Li, T., Luo,C., Zhang, J., Chen, H. (2016). Incremental updating of rough approximation in interval-valued information systems under attribute generation. Information Sciences, 373, 461-475.

Zhong, N., Dong, J. Z., & Ohsuga, S. T. Y. L. (1998). An incremental probabilistic rough set approach to rule discovery. The IEEE International Conference on Fuzzy Systems, 2. doi:10.1109/ FUZZY.1998.686243

Zhong, N., & Skowron, A. (2001). A rough setbased knowledge discovery process. International Journal of Applied Mathematics and Computer Science, 11, 603–619.

https://www.mimuw.edu.pl/ son/datamining/RSDM/Intro.pdf

https://www.webpages.uncc.edu/ras/KDD-02/rough-sets.PPT

10.6 Yazar Hakkında

Dr. Safiye Turgay lisans derecesini, İstanbul Teknik Üniversitesi Endüstri Mühendisliği, yüksek lisans ve doktora derecesini Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Ana Bilim dalından almıştır. Abant İzzet Baysal Üniversitesi Bilgisayar Programcılığı, Bilgisayar ve Öğretim Teknolojileri ve Eğitimi, İşletme, Sakarya Üniversitesi Yönetim Bilişim Sistemler bölümlerinde öğretim üyeliği yapan Safiye Turgay halen Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği bölümünde öğretim üyesi olarak görev yapmaktadır. Çok etmenli sistemler, bulanık mantık, karar destek sistemleri, üretim sistemleri, çok kriterli karar verme teknikleri ve kaba kümeleme konularında çok sayıda yayını bulunmaktadır.

Prof. Dr. Orhan Torkul, Sakarya Üniversitesi Mühendislik Fakültesi Dekanı ve Endüstri Mühendisliği Bölümü Profesörüdür. 2011-2014 yılları arasında Endüstri Mühendisliği Bölümü Başkanı olarak görev yaptı. 1993 yılında İngiltere'de Cranfield Institute of Technology Enstitüsünde doktora unvanını almıştır. Üretim planlama ve kontrol, yönetim bilgi sistemleri ve uzaktan eğitim / öğretim alanlarında araştırmalar yapmıştır. Uluslararası dergiler, konferanslar ve ulusal sempozyumlarda çok sayıda yayınları bulunmaktadır. Birkaç uluslararası konferans ve sempozyum düzenledi. Davetli konuşmacı olarak sayısız konferans, atölye çalışması ve diğer bilimsel toplantılara katıldı. Prof. Dr. Orhan Torkul, Yönetim Bilgi Sistemleri, Bilgi Sistemleri Güvenlik ve Kontrol ve İleri Üretim Planlama ve Kontrol Sistemleri konularında çeşitli dersler vermektedir.

11. Derin Öğrenmeye Giriş

Derin Öğrenmeye Giriş

Arş. Gör. Muhammet Raşit CESUR, Prof. Dr. Orhan TORKUL, Yrd. Doç. Dr. Mümtaz İPEK, Yrd. Doç. Dr. İhsan Hakan SELVİ, Prof. Dr. İsmail Hakkı CEDİMOĞLU

11.1 Yapay Zekanın Gelişimi

Yapay Zekâ başlangıcı 1950 yıllarına dayanmaktadır. Turing Makinası ile bilgisayarın insan gibi cevap verip veremeyeceği test edilmiştir. 1960 yıllarında ise Yapay Sinir Ağları ortaya çıkmıştır. Makine öğrenme kullanım tarihi ise 1990 yıllarına kadar gitmektedir. 2006 yılında derin öğrenme kavramını ilk defa kullanılmıştır. 2012 yılında ilk derin öğrenme uygulaması gerçekleştirilmiştir.

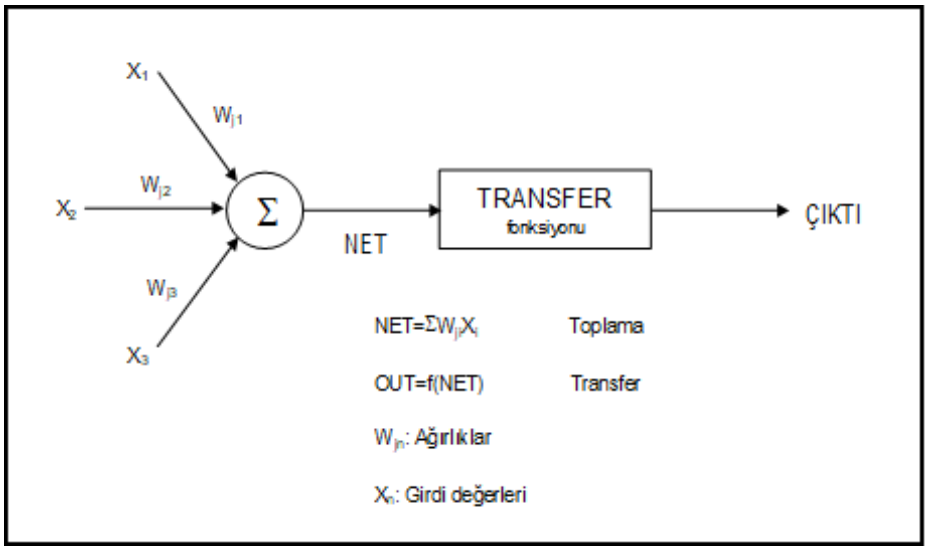
11.2 Yapay Sinir Ağları

Bir sinir ağı, insan beynine dayalı bir çıkarım mekanizması olarak tanımlanabilir (Negnevitsky, 2002). Diğer bir deyiş ile yapay sinir ağları, birbirine paralel bağlı basit elemanlar ile biyolojik sinir sistemlerinin yaptığı gibi gerçek dünya nesneleriyle etkileşiminin gerçekleştirildiği planlı hiyerarşik yapılardır (Kohonen, 1987). Bir genel sinir ağı modeli Şekil 11.1'de gösterilen proses elemanları ile karakterize edilebilir. Bu şekilde, bir proses elemanında beş bileşenin yer aldığı görülmektedir. Bunlar aşağıdaki gibi ifade edilebilir:

- Girdiler, proses elemanına bilgi getirmektedir. Bu bilgi diğer proses elemanları veya dış kaynaklardan sağlanabilir. Bazen proses elemanı kendi kendine bilgi verebilir. Ağırlıklar, belli bir girdinin proses elemanı üzerindeki etkisini belirlemektedir. Ağın doğru çıktı üretebilmesi için eğitme işlemi esnasında optimize edilmesi gereken ağırlık değerleridir.
- Toplama fonksiyonu, proses elemanının ağırlıklı girdilerini toplar. Literatürde çok fazla toplama fonksiyonları vardır. En yaygın olanı ağırlıklı toplamı bulmaktır. Burada gelen her girdi değeri, kendi ağırlığı ile çarpılarak toplanır (Öztemel, 2003).

Not: Kapak resmi https://en.wikipedia.org/wiki/Computer_vision adresinden alınmıştır.

- Transfer fonksiyonu, toplama fonksiyonunun sonucunu düzenleyerek proses elemanının çıktısını belirlemektedir. Literatürde çeşitli transfer fonksiyonları mevcuttur. Sigmoid fonksiyon, lineer fonksiyon ve step fonksiyon kullanılan transfer fonksiyonlarından birkaçıdır.
- Çıktı, transfer fonksiyonunun sonucunun, bağlı olan proses elemanlarına veya dış kaynaklara gönderilmesidir.



Şekil 11.1: Bir geriye yayılım sinir ağının topolojisi

Ağın topolojisi, ağı karakterize eden başka bir özelliktir. Tipik bir sinir ağı, birbirine bağlı üç katmandan oluşur. Bunlar, dışardan veri kabul eden girdi katmanı, girdi katmanından gelen bilgileri işleyerek çıktı katmanına gönderen gizli katman ve ağın kararını belirten çıktı katmanıdır. Bilgi ağın katmanları arasında veya katmanlar içinde akmaktadır (İpek, 2007).

11.2.1 Genel Özellikler

Yapay sinir ağının karakteristikleri ağın algoritmik olmayan, paralel ve dağıtılmış bilgi işleme yeteneklerine dayanmaktadır. Bu yetenekler sinir ağlarının kolaylıkla gerçek zamanlı olarak uygulanabilmesini sağlayabilir ve herhangi bir zorluk olmaksızın karmaşık, doğrusal olmayan hesaplamaları yapabilmesine, hızlı cevap verebilmesine sağlayabilmektedir. Yapay sinir ağlarının bazı karakteristikleri şu şekilde özetlenebilir:

- Örneklerden öğrenme. Sinir ağları sadece öğreneceği varsayılan girdi/çıktı ilişkileri örneklerine ihtiyaç duymaktadır. Bu örnekleri kullanarak genellemeler yapabilir.
- Örüntü tanıma veya sınıflama. Sinir ağları örüntüleri girdi olarak alırlar ve örneklerden alınan girdi/çıktı eşleştirmesiyle ilgili bilginin saklandığı dağıtılmış ilişki belleğini kullanarak uygun çıktıları üretebilirler.
- Örüntüleri yeniden yapılandırma. Yapay sinir ağları tam olmayan örüntüleri işleyebilir ve girdi örüntüsünü belleğindeki tam olan örüntü ile karşılaştırarak eksik bilgiyi örüntüye yerleştirebilir.
- Kendi kendini organize etme. Bazı sinir ağları kendi kendine organize edilebilir ve öğrenebilir. Bir şekilde çevre değiştiğinde bu ağlar kendilerini yeni bir duruma adapte edebilirler.
- Hata toleransı. Bilgi bütün ağ üzerinde dağıtıldığından, bilgideki küçük parçaların kaybı veya

hasarı ciddi olarak ağın performansına etki etmez.

- Bulanık veya gürültülü girdilerden etkilenmemektedir. Sinir ağları eksik veya bulanık veriyle karşılaştığı zaman sunulan girdi örüntüsü için en uygun çıktıyı seçer. Yapay sinir ağlarının bu özellikleri nedeni ile kullanımı artmaktadır (Wells, 1992).

11.2.2 Öğrenme

Bir sinir ağının öğrenme amacı, ağın belirli girdiler ile doğru çıktıları üretmesi için en uygun ağırlık değerlerini bulmaktır. Bilgi bütün bağlantılara dağıtıldığı için, tek bir bağlantı genellikle anlamlı bir bilgi içermeyebilir. Bilginin oluşturulması için bir grup bağlantının düşünülmesi gereklidir. Bir ağın genellikle, bahis konusu probleme bir cevap verebilmesi için, bütün bağlantılarında uygun ağırlık değerlerine sahip olması gerekmektedir. Bu, öğrenme veya eğitim adı verilen bir proses ile elde edilir. Öğrenme, ağırlık değerlerini değiştirme yolunu belirleyen bir öğrenme kuralına dayanır. Bir öğrenme kuralının temel prensipleri, kullanılan öğrenme stratejisiyle belirlenir. Literatürde tartışılan üç tip öğrenme stratejisi aşağıda verilmiştir.

Öğretmenli öğrenmede ağı denetleyecek bir öğretmene ihtiyaç duyulur. Öğretmen basitçe ağa çıktı katmanında ne üretmesi gerektiğini söyler (Rumelhart ve diğerleri, 1986).

Destekleyici öğrenmede yine bir öğretmene ihtiyaç vardır. Fakat ağa ne üretmesi gerektiği söylenmez. Sadece, üretilen çıktının doğru veya yanlış olduğu söylenir (Wells, 1992).

Öğretmensiz öğrenmede ağa yardımcı olacak herhangi birine gerek yoktur. Bu durumda ağ, girdi/çıktı eşleştirmelerini organize etmek için kendi kriterlerini geliştirir. Bu stratejiyi kullanan ağlar " kendi kendine organize olan ağlar" olarak adlandırılır (Carpenter ve Grossberg, 1987), (Nelson ve Illingworth, 1991).

11.2.3 Geriye Yayılım Ağları

İlk defa Rumelhart ve diğerleri (1986) tarafından sunulan geriye yayılım modeli, en çok kullanılan sinir ağı modellerinden birisidir. Geriye yayılım, yapay sinir ağlarının uygulanabileceği problem aralığını büyük ölçüde genişletmiş ve özellikle çeşitli girdi/çıktı eşleştirme problemlerinde başarılı sonuçlar üretmiştir (Wasserman, 1989, Rabelo ve Alptekin, 1990).

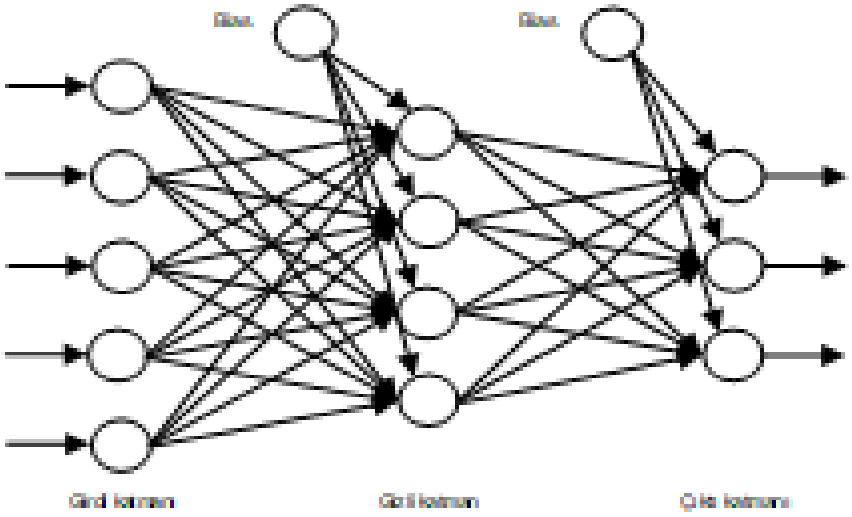
Geriye yayılım ağı, temel olarak girdi örüntüsüyle birlikte ağa sunulan sistemin beklenen çıktılarına gerek duyan öğretmenli öğrenme stratejisini kullanır. Öğrenme mekanizması, ağın gerçek ve beklenen çıktıları arasındaki hatayı minimize eden yinelemeli eğim azalması (iterative gradient descent) metoduna dayanır.

11.2.4 Bir Geriye Yayılım Ağının Yapısı

Geriye yayılım ağları, çeşitli katmanlar içeren tam bağlantılı ileri besleme ağlardır (Şekil 11.2). Her bir proses elemanı bir sonraki katmandaki her bir proses elemanıyla bağlantılıdır. Fakat bağlantılar geri besleme yoktur. Ayrıca aynı katmandaki proses elemanları arasında herhangi bir bağlantı mevcut değildir. Ağ üç tip katmandan oluşur: Girdi örüntülerini sonraki katmana aktaran bir girdi katmanı, bir veya daha fazla gizli katman ve bir çıktı katmanı.

11.2.5 Geriye Yayılım Öğrenme İşlemi

Verini girdi katmanında ve gizli katmanda işlenmesi ile, ağın çıktıları hesaplanır ve istenen çıktılarla karşılaştırılır. Bu çıktılar arasındaki hata, azalan eğim algoritması kullanılarak ağın bağlantılarına doğru geriye yayılır. Öğrenme metodu çoğunlukla genelleştirilmiş delta kuralı olarak adlandırılır. Standart bir geriye yayılım ağının proses elemanları toplama fonksiyonu ve transfer fonksiyonu



Şekil 11.2: Bir geriye yayılım sinir ağının topolojisi

olarak sırasıyla ağırlıklı girdilerin toplamı ve sigmoid fonksiyonuna sahiptir. Öğrenme işlemine etki eden birkaç faktör vardır. Bunlar, öğrenme katsayısı, momentum ve bias terimidir.

11.3 DERİN ÖĞRENME

Makine öğrenme, yapay zekânın bir koludur. Makine öğrenme, veri ile eğitilebilir. Daha sonra, elde ettiği bilgi ile tahminlerde bulunabilir. Derin öğrenme ise makine öğrenmenin bir dalıdır. Büyük veri üzerinde çalışan, ön eğitimli (farklı işler için özelleşmiş) çok sayıda yapay zekâdan oluşan, çok sayıda düğüme sahip ağlar derin öğrenme olarak adlandırılır. Derin öğrenme uygulamaları büyük ve etiketsiz veri üzerinde çalışırken öğretmensiz ön eğitim (pre- unsupervised trained), ardından öğretmenli öğrenme süreçlerinin çalıştırıldığı sistemlerdir. Bu uygulamaların temel özelliklerin biri de ağdaki düğüm miktarını çok büyük olmasıdır. Örneğin, Google Brain'de 1 milyar düğüm mevcuttur. Böyle bir yapının en büyük eksiği bir objenin öğrenilmesinin uzun zaman almasıdır. Örneğin, bir obje, 10 milyon videodan, 16000 bilgisayarla 3 günde derin öğrenme ile öğrenilmiştir (ALPAYDIN, 2011).

Derin öğrenme sistemlerinin hem öğrenme hem de karar verme aşamasında çıkarılan özelliklerden yakın olanların bütünleştirilmesi aşaması alt küme (Pooling) oluşturmadır. Milyonlarca farklı verinin (görüntü, ses ve bunun gibi) tüm özellikleri birebir aynı olmayacağı için bu safha önemlidir. Oluşturulan alt kümelerin stokastik olarak elenmesine silme (dropout) denir. Bu sayede genellikle zayıf ilişkiye ait olan bağlar koparılır. Silme katmanları sadece kıvrımlı ağ için kullanılmazlar. Diğer ağlar için de kullanılabilir (Zocca ve diğerleri, 2017). Yapılan işlem gürültü giderme işlemine benzemektedir. Verideki gürültü ise girdi sayısı çıktı sayısına eşit olan yapay sinir ağlarıyla (autoencoder) sağlanır. Derin öğrenme sistemlerinde öğrenme ve karar verme karar verme süreçleri göz önünde bulundurulduğunda aşağıdaki temel bileşenlerin her derin öğrenme ağında bulunduğu anlaşılmaktadır.

- Parametreler
- Katmanlar
- Aktivasyon fonksiyonları

- Zarar fonksiyonları
- Optimizasyon yöntemleri
- Hiper parametreler (Patterson ve Gibson, 2017).

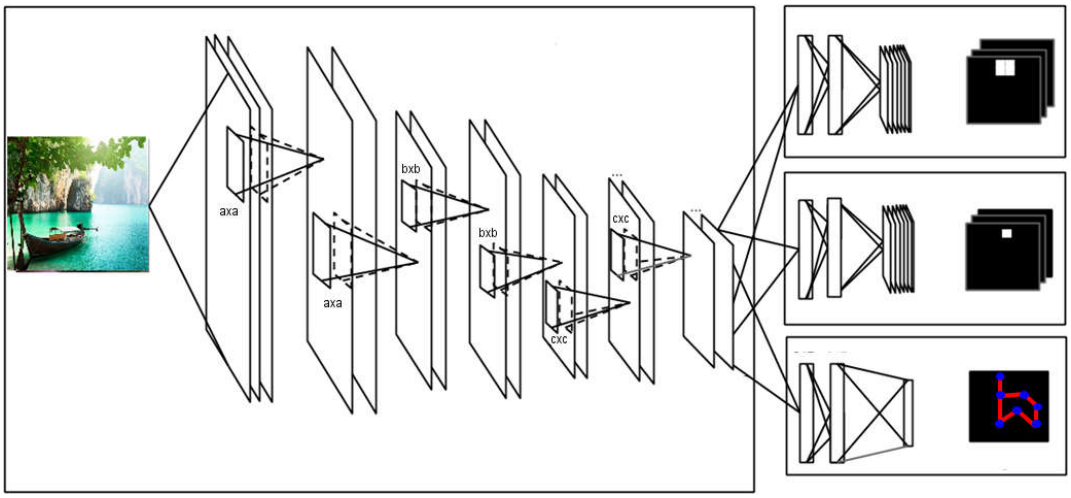
11.3.1 Derin Anlama Ağları

Derin Anlama Ağları(Deep Belief Nets), Kısıtlı Boltzman Makine Network'üne dayanmaktadır. Boltzman makinesi 2 kademeli, düğümlerin birbirine tam bağlı olduğu bir yapay sinir ağıdır. Bu yüzden hesaplamalar uzun zaman alabilir. Kısıtlı Boltzman makinesinin, aynı katmandaki düğümlerde bağlantı yoktur. Bu yüzden hesaplamalar, Boltzman makinesine göre daha kısa zamanda tamamlanabilir.

11.3.2 Kıvrımlı Yapay Sinir Ağları

Kıvrımlı Yapay sinir ağlarında, düğümler üç boyutlu olarak organize edilmiştir. Bu üç boyut en, boy yükseklik veya en, boy ve renk olabilir. Kıvrımlı katmandaki düğümler bir önceki katmandaki lokal bölge ile bağlantılıdır. Kıvrımlı katman 3 boyutlu veriyi işleyip, yeni bir üç boyutlu bilgi elde etmektedir (Buduma, 2017). Kıvrımlı yapay sinir ağları (Convolutional Neural Network), görüntü işleme problemleri için kullanılır. Konuşma tanıma için de kullanılabilir. Şekil 11.3'te görülen kıvrımlı sinir ağlarının özellikleri aşağıdaki verildiği gibidir.

- Birden fazla katman mevcuttur.
- Pooling katmanı mevcuttur.
- Tam bağlantılı katman vardır.



Şekil 11.3: Kıvrımlı yapay sinir ağlarının yapısı

11.3.3 Derin Öğrenmenin Standart İleri Beslemeli Yapay Sinir Ağların'dan Farkı

Bir katmanlı, ileriye doğru yapay sinir ağı, doğrusal olarak ayrılabilir veri ile çalışır. Gizli katmanlı yapay sinir ağlarının karmaşık problemlerdeki performans düşüklüğü, birden fazla gizli katmanlı modellerin geliştirilmesine yol açmıştır. Ayrıca tekrarlayan yapay sinir ağları ile, bilginin geriye doğru beslenmesi sağlanmıştır. Diğer ağlarda ise hafıza oluşturma süreci ortaya çıkmıştır (Patterson ve Gibson, 2017). Derin öğrenme sistemleri YSA'larda kullanılandan daha fazla düğüme sahiptir ve

çok sayıda düğüme rağmen başarılı olarak çalışmaktadır. Düğüm sayısı fazla olduğu için Katmanları bağlamak için daha karmaşık yollar belirlenmiştir. Bu durum, sistemin yüksek bir veri işleme gücüne ihtiyaç duymasına neden olur. Derin öğrenme sistemlerinin bir diğer farkı da otomatik özellik çıkarımı yapabilmesidir.

11.3.4 Python Derin Öğrenme Kütüphaneleri

Python kullanılarak geliştirilen çok sayıda derin öğrenme kütüphanesi bulunmaktadır. Genel olarak 2 temel kütüphane üzerine inşa edilen bu kütüphaneler aşağıda verilmiştir.

- **Theano** matematiksel ifadeler için optimizasyon ve yapay zeka fonksiyonlarını içeren kütüphanedir. GPU'yu kullandığı için çok boyutlu ve elemanlı diziler üzerinde etkin çalışmaktadır. Alt düzey bir kütüphanedir. Uzmanlık istemektedir (Theano 0.9.0, 2017).
- **Lasagne** kütüphanesi Theano üzerinde geliştirilmiş bir derin öğrenme kütüphanesidir. Kütüphanenin katkısı ağ modellerinin özelleştirilmesi ve kullanıcıya katmanlar üzerinde çalışma imkanı sağlamasıdır. Model Lasagne requires little sacrifice in terms of flexibility while providing a wealth of common components to help with layer definition, layer initialization, modelin düzenlenmesi, eğitimi ve izlenmesi için biraz kod yazmak gerekmektedir (Lasagne, 2017).
- **Blocks**, Lasagne gibi Theano üzerinde geliştirilmiş bir kütüphanedir. Theano üzerinde geliştirilen ilk yüksek seviyeli yapay sinir ağı kütüphanesidir. Lasagne'dan daha esnektir. Blocks yinelemeli yapay sinir ağları üzerinde gayet iyi bir performans göstermektedir (Blocks, 2017).
- **TensorFlow**, Theano ve Lasagne gibi kütüphanelere alternatif olarak geliştirilmiş bir kütüphanedir. Fonksiyonların tasarımı açısından Theano ve Lasagne'ın karşımı bir kütüphanedir denebilir. Google Brain tarafından desteklenmektedir. GPU desteği çok güçlüdür ve çoklu GPU desteği vardır. Geniş bir geliştirici topluluğu vardır (TensorFlow, 2017).
- **Keras**, oldukça kullanıcı dostu, üst seviye bir derin öğrenme kütüphanesidir. Backend olarak Theano veya TensorFlow seçilebilmesi kütüphanenin en önemli avantajlarından birisidir. Çok geniş ve aktif bir geliştirici topluluğu vardır. Yakında TensorFlow projesinin bir parçası olarak geliştirilmeye devam edecektir (Keras, 2017).
- **MXNet**, Amazon'un geliştirmiş olduğu bir derin öğrenme kütüphanesidir. Çoklu GPU desteği vardır. Kütüphanenin tasarımı Lasagne ve Blocks'a benzemektedir. Çok geniş bir donanım yelpazesi üzerinde çalıştırılabilir. MXNet kütüphanesi Pyhton, R, Julia, C++, Scala, Matlab, and Javascript dilleri ile kullanılabilir (MxNet, 2017).
- **PyTorch** yeni yayınlanmış bir kütüphanedir. Facebook yapay zeka araştırma ekibi tarafından yayınlanmıştır. Dinamik İşlem Grafikleri ile çalışabilmesi en önemli özelliklerinden biridir. Bu özellik kod geliştirmek yerine akış diyagramlarıyla proje geliştirmeye olanak sağlamaktadır. Bu özellik TensorFlow'da da bulunmaktadır (PyTorch, 2017).

Python kütüphanelerinin dışında çevrimiçi kullanılabilecek derin öğrenme sistemleri (bulut api) bulunmaktadır. Bu sistemler aşağıda verilmiştir.

- MetaMind
- PhotoTag
- Place2
- Clarifai

Yaygın olarak bilinen derin öğrenme uygulamaları ise şu şekilde sıralanabilir;

- Yüz tanıma (Facebook)
- Siri (Apple)
- Anlık çeviri (Skype)

- Otonom araç (Tesla)
- Otonom araç, DeepMind (Google).
- Watson Platformu (IBM)
- AskADoctor, Resim arama (Baidu arama motoru)
- JetPack SDK derin öğrenme kütüphanesi, cuDNN, resim renklendirme (nVdia)

11.3.5 Derin Öğrenme Uygulama Tasarımı

Derin öğrenme uygulamaları klasik yapay zeka teknolojilerinin aksine veriyi çok boyutla analiz ederek karar sürecinde bu analizler arasındaki benzerlikler ve farklılıklara odaklanır. Özellikle silme işlemi zayıf ilişkileri ortadan kaldırmak için kullanılırken, oluşturulan alt kümelerde benzer girdiler toplanmış olur. Alt kümelerin oluşturulması ve ilişkilerin silinmesi öğretmensiz öğrenme yaklaşımıyla yapılmaktadır. Bu işlemlerin sonucunda elde edilen alt kümelerin sonuçla ilişkilendirilmesi ile sınıflandırma işlemidir. Bu durum beynin fonksiyonlarına benzetilebilir. Beyinde görme için bir merkez çalışırken ses algılama, koku algılama ve sinir sistemine bağlı diğer işlemlerin gerçekleştirildiği merkezler vardır. Bu merkezlerin çıktıları ara beyine gönderilerek öğrenme sağlanmaktadır. Derin öğrenme süreci de bu duruma benzemektedir. Derin öğrenme uygulamalarının temel özellikleri aşağıda verilmiştir (Zocca ve diğerleri, 2017).

- Büyük veri üzerinde çalışır.
- Öğretmensiz öğrenme ile eğitilmiş yapay zekalardan oluşur.
- Öğretmenli öğrenme yöntemleriyle eğitilebilir.
- Ağdaki düğüm sayıları çok yüksektir.

Derin öğrenme uygulamaları büyük veri üzerinde çalıştığı için uygulamada çok geniş aralıktaki veri setleri kullanılmaktadır. Derin öğrenme uygulamalarında sıklıkla kullanılan girdiler aşağıda verilmiştir.

- Görüntü
- Ses
- Metin
- Yapısal Veri

11.3.6 Görüntü

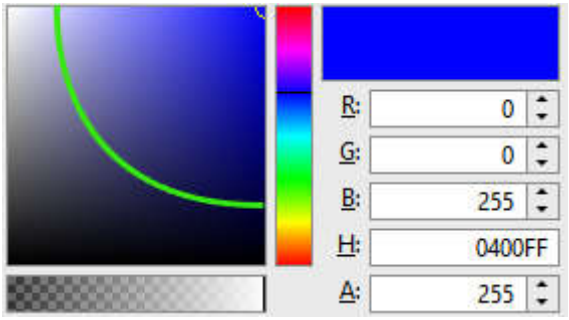
Dijital resimlerdeki renk verisi saydamlık (Alpha), kırmızı (Red), yeşil (Green) ve mavi (Blue) renk değerlerini birlikte içerir. Bu değerler ton (Hue), doygunluk (Saturation) ve parlaklık (Value) değerlerine dönüştürülebilir. Kameradan gelen görüntü içerisinde ise saydamlık değeri olmaz.

Rengin ayrıştırılabilmesi için kırmızı, yeşil mavi ve parlaklık değerlerine bakılmalıdır. Renk verisinin siyah, gri ve beyaz aralıktan uzak olduğu noktalar (Şekil 11.4) kırmızı, yeşil ve maviden birisine veya ikili renk çiftine belirli uzaklıkta bir dairesel alanda yer alır. Renkleri ayırmanın en etkili yollarından biri kırmızı, yeşil, mavi ve parlaklık (RGBV) değerlerine göre kümelemedir.

Kümelenen değerlere en hızlı ulaşma yolu ikili ağaç (binary tree) kullanmaktır. Veriler ağaca yazılırken RGBV değerleri karşılaştırılacaktır. İki renk verisinden birinin tüm değerleri küçükse karşılaştırma sonucu bellidir. RGBV değerlerinin bazıları büyük bazıları da küçükse?

Resimden çıkarılacak özellikler aşağıdaki gibidir (Li C., et.al., 2017).

- Renk değerleri arasındaki toplam fark.
- Resimler arasındaki farklı noktaların sayısı.
- Resimler arasındaki farklı noktaların ortalaması (pozisyon değerleri).
- Resim içindeki nesneleri orta noktalarının (skeleton) pozisyonları.



Şekil 11.4: Renk verisi

$$\frac{r_1 - r_2}{255} + \frac{g_1 - g_2}{255} + \frac{b_1 - b_2}{255} + v_1 - v_2 \begin{cases} > 0 \Rightarrow renk1 > renk2 \\ < 0 \Rightarrow renk1 < renk2 \end{cases}$$

- Noktaların merkezden ya da bir referans noktasından farkları.
- Kenar noktaları.
- Köşe noktaları.
- Filtreden geçen noktaların sayıları ve pozisyonları.

11.3.7 Metin ve Ses

Metin ve ses verileri sinyal olarak işlenebilecek tek boyutlu dizilerden oluşur. Metin belirli byte değerlerine sahip harflerden oluşurken, ses -128 ile +127 arasındaki tüm değerleri alabilen, saniyede 2 defa elde edilen 8000 elemanlı dizilerden oluşur. Metin ve ses verilerinin saklanması için en uygun iki veri yapısı ağ (map) veya ağaçtır (tree).

Metin madenciliğindeki en önemli faktörler harf ve kelimelerin dizilimleri ile yan yana geçme sıklıklarıdır. Çünkü kelimelerin ve cümlelerin oluşma biçimleri böyledir. Kelimedeki harflerin birlikte bulunma sıklığından kelimenin hangi dile ait olduğundan, cümle içindeki görevine kadar bir sürü özellik çıkarılabilmektedir. Metin verilerin işlenmesi sırasında çıkarılacak özellikler aşağıda verilmiştir (Christopher D., et. al., 2017).

- Metinlerin yan yana geçme sıklıkları.
- Yazım hataları;
  - Fazla harf yazma
  - Eksik harf yazma
  - Harflerin yerini değiştirme
  - Yanlış harf yazma
- Kelimelerin türleri (eylem, sıfat, zarf, zamir vb. . .)
- Kelimelerin ekleri.
- Harflerin okunuş ve şive özellikleri.
- Dile özgü harfler veya harf dizilimleri.
- Cümle içinde kelime dizilimleri.

Ses verileri de metne benzer mantıkla işlenmesine rağmen, ses sinyalleri kelimelerden göre çok daha büyük boyutlarda olduğu ve sensör verilerini içerdiği için arada bazı temel farklılıklar oluşmaktadır. Bu farklılıklardan birisi aynı kelimenin farklı kişiler tarafından birebir aynı sinyal değerlerini üretecek biçimde söylenememesidir. İkincisi de yarım saniyede ortalama 6 harfli bir

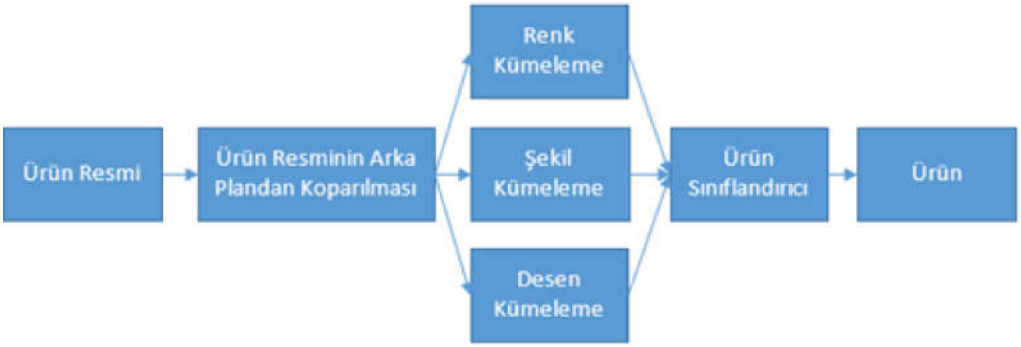
kelime söylenebildiği için bir harfin yaklaşın 1000 - 1500 bytelık bir sinyal tarafından temsil edilmesidir. Ayrıca seste yanlış yazım gibi problemler de olmamaktadır. Ses işlemede böyle problemler olmamakla birlikte, çözülmeye çalışılan problem, sinyal değerlerinin doğru harfle eşleştirilmesidir (sınıflandırılmasıdır). Ses verilerin işlenmesi sırasında çıkarılacak özellikler aşağıda verilmiştir.

- Sinyal verisindeki değerler.
- Referans sinyal dizisi ile aradaki fark toplamı, ortalaması ve standart sapması.
- Sinyal değerinin sıklığı.
- Sinyal değerlerdeki değişim miktarı.
- Fourier gibi sinyal dönüşümü sonuçları.
- Sinyal verisi ortalama ve sapması gibi istatistiksel özellikleri.
- Ses dizisi (8000 adet sinyal) bazında güç tahmini.

11.3.8 Derin Öğrenme Uygulaması

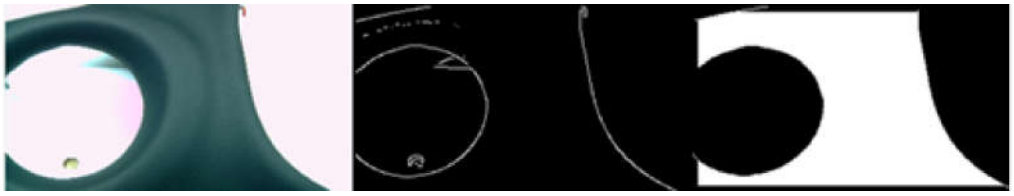
Torkul ve arkadaşlarının geliştirdiği uygulamada, kamera ile görünen ürünün tanınması hedeflenmektedir (Torkul, et.al., 2017). Ürünlerin yapay zeka tarafından tanınması esnasında Şekil 11.5'te görülen süreç için aşağıdaki işlemler gerçekleştirilmiştir.

- Ürünün görüntü üzerinden seçilmesi.
- Renk, şekil, desen analizleri.
- Veri tabanıyla eşleştirme işlemleri gerçekleştirilmiştir.



Şekil 11.5: Uygulama akışı

Arka planın temizlenip sadece ürün görüntüsünün elde edilmesi için A* algoritması kullanılmıştır. Geliştirilen sezgisel model, Şekil 6'da görüldüğü gibi, Canny filtresinden geçirilen resimdeki noktaların hangilerinin ürün kenar noktaları olduğuna karar vermektedir.



Şekil 11.6: Ürünün arka plandan ayrılması (Torkul, et.al., 2017)

Ürünün elde edilen görüntüsü renk, şekil ve desen özellikleri çıkarılarak kümelenmektedir. Görüntü üzerinde parlama ve gölgelenmeler olduğu için renk değerleri ürünün her tarafında aynı

olmamaktadır. Kümeleme arka plandan şekli olarak tam alınmadığında, desenler birebir aynı şekle sahip olmadığında ve renk analizinde gölgeyle parlamanın ayrıştırılmasında çok önemli bir katkı sağlamaktadır. Belirlenen kümelerin özellikleri ise bir sınıflandırıcı aracılığıyla sınıflandırılmakta ve ürün veri tabanındaki ürünlerle eşleştirilerek tahmin edilmektedir.

11.4 KAYNAKLAR

ALPAYDIN, E., Yapay Öğrenme, Boğaziçi Üniversitesi Yayınevi, 2011

Blocks, Blocks 0.2.0 Documentation, https://blocks.readthedocs.io/en/latest/, 10.08.2017

BUDUMA, N., Fundamentals of Deep Learning, Designing Next-Generation Machine Intelligence Algorithms, O'Reilly Media, 2017.

CARPENTER, G. A., GROSSBERG, S., " ART2: Self Organisation of Stable Category Recognition Codes for Analog Input Patterns", Applied Optics, No. 26-23, s. 4919-4930, 1987.

CHRISTOPHER D. M., Prabhakar R., and Hinrich S., Introduction to Information Retrieval, Cambridge UP, 2009

İPEK, M., " Dinamik Atölye Çizelgelemede Yapay Sinir Ağı İle Teslim Tarihi Belirlenmesi", Doktora Tezi, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Temmuz 2007, Sakarya.

KERAS, Keras Tutorial: Deep Learning in Python, https://www.datacamp.com/community/tutorials/deep-learning-python, 10.08.2017

KOHONEN, T., " State of the Art in Neural Computing", IEEE First International Conference on Neural Networks, No. 1, s. 79-90, 1987.

LASAGNE, Lasagne Documentation, http://lasagne.readthedocs.io/en/latest/user /installation.html, 10.08.2017

LI, Chao; WANG, Xinggang; LIU, Wenyu. Neural features for pedestrian detection. Neurocomputing, 2017, 238: 420-432.

MXNET, MxNet Tutorials, https://mxnet.incubator.apache.org/tutorials/index.html , 10.08.2017

NEGNEVITSKY, M., " Artificial Intelligence: A Guide to Intelligent Systems", Addison-Wesley, 2002.

NELSON, M. N., ILLINGWORTH, W. T., " A Practical Guide to Neural Nets", Addison-Wesley, 1991.

ÖZTEMEL, E., " Yapay Sinir Ağları", Papatya Yayıncılık, 2003.

PATTERSON, J., GIBSON, A., Deep Learning, O'Reilly Media, Inc., 2017.

PYTORCH, PyTorch Documentation, http://pytorch.org/docs/master/, 10.08.2017

RABELO, L. C., ALPTEKIN, S., " Adaptive Scheduling and Control Using Artificial Neural Networks for a Hierarchical/Ditributed FMS Architecture, IEEE Second International Conference on Computer Integrated Manufacturing, s. 538-545, 1990.

RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J., " Learning Representations by Back Propagating Errors", Nature, No. 323, s. 533-536, 1986.

TENSORFLOW, TensorFlow Tutorials, https://www.tensorflow.org/tutorials/ wide_and_deep, 10.08.2017

THEANO 0.9.0, Theano 0.9.0 Documentation, http://deeplearning.net/software/ theano/, 10.08.2017

TORKUL, O., CEDİMOĞLU, İ. H., CESUR, M. R., OKUYAN, A., A Real - Time Cyber Platform Enabling Digital Factory, Internation Symposiom for Production Research, 2017, 13 - 1 5 September 2017, Vienna, 775 - 779

WASSERMAN, P. D., " Neural Computing: Theory and Practice, Van Nostrand Reinhold, New York, 1989.

WELLS, G., " An Introduction to Neural Networks", Applications of AI in Process Control, 1992.

ZOCCA, V., SPACAGNA, G., SLATER, D., ROELANTS, P., Python Deep Learning, Packt Publishing, 2017.


WWW.YAZSUM.SAKARYA.EDU.TR