Tellekt

Yapay Öğrenme

Yeni Yapay Zekâ

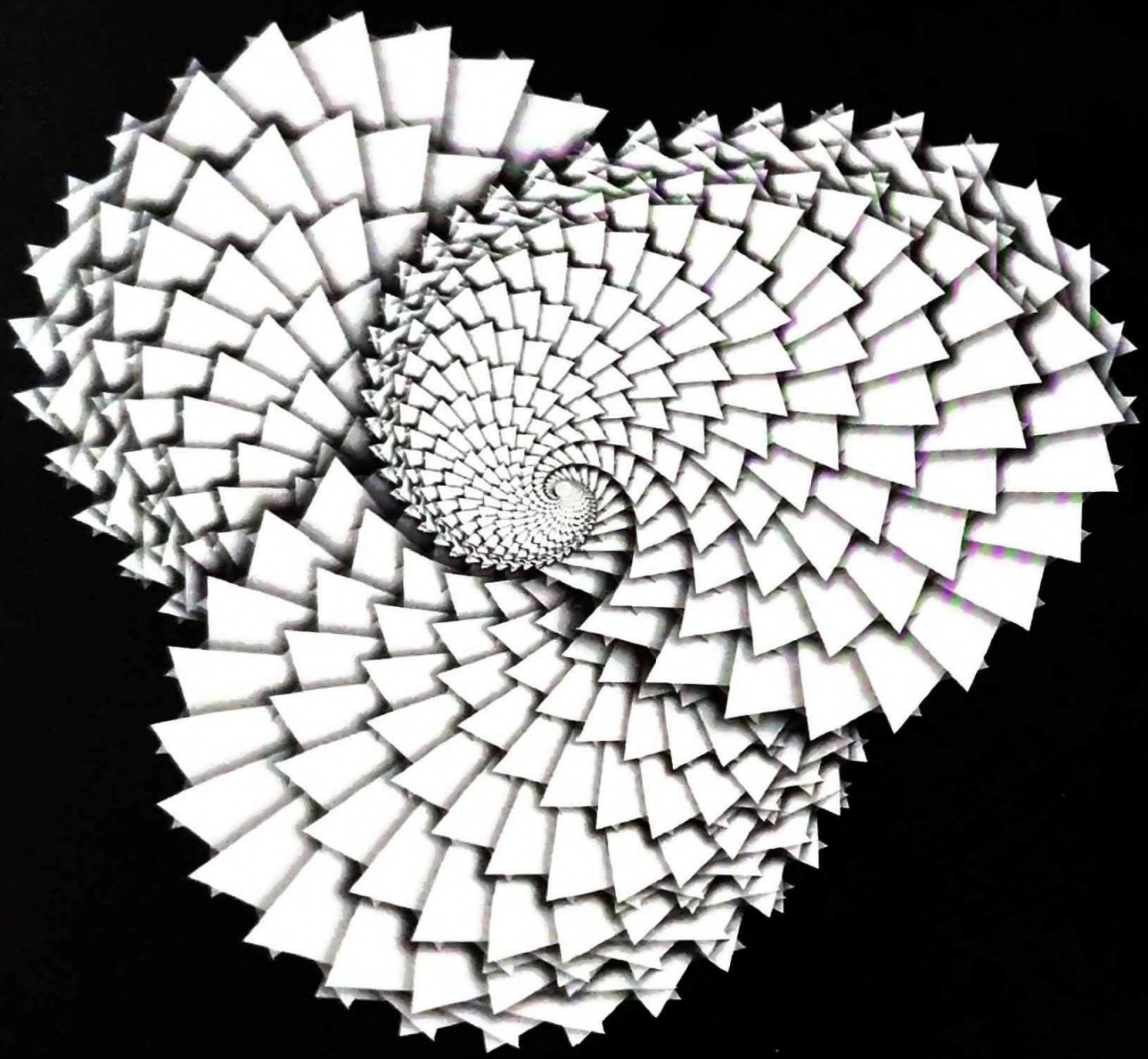


Ethem Alpaydın

2. BASKI



ÇEVİRİ: AYLİN AĞAR



YAPAY ÖĞRENME:

YENİ YAPAY ZEKÂ

Tellekt_20

Yapay Öğrenme: Yeni Yapay Zekâ
Çeviri: Aylin Ağar

Machine Learning: The New AI (The MIT Press Essential Knowledge Series)
İlk basım (çeviriye kaynak alınan basım): MIT Press, 2018
© 2018, Massachusetts Institute of Technology
© 2019, Can Sanat Yayınları A.Ş.
Bu kitap ilk kez 2019 yılında QNB Finansbank işbirliğiyle yapılan Ufuk Açan Yayınlar dizisinde yayımlanmıştır.
Bu eserin Türkçe yayın hakları Kayı Telif Hakları ajansı aracılığıyla alınmıştır.
Tüm hakları saklıdır. Tanıtım için yapılacak kısa alıntılar dışında yayıncının yazılı izni olmaksızın hiçbir yolla çoğaltılamaz.

1. baskı: 2020
2. baskı: Mart 2021, İstanbul
Bu kitabın 2. baskısı 1000 adet yapılmıştır.

Yayına hazırlayan: Didem Bayındır
Düzelti: Melis Oflas
Mizanpaj: Bahar Kuru Yerek

Kapak Tasarımı ve Uygulama: Bora Başkan
İç Kapak Görseli: Frank Cone

Baskı ve cilt: Türkmenler Matbaacılık Reklam San. ve Tic. Ltd. Şti.
Maltepe Mah. Gümüşsuyu Cad. No: 16-18
Topkapı, İstanbul
Sertifika No: 43087

ISBN 978-625-7118-09-5

Tellekt
tellekt.com • bilgi@tellekt.com

Maslak Mah. Eski Büyükdere Cad. İz Plaza, No: 9/25 Sarıyer / İstanbul
Telefon: (0212) 252 56 75 / 252 59 88 / 252 59 89 Faks: (0212) 252 72 33
Sertifika No: 43514
Tellekt, Can Sanat Yayınları Yapım ve Dağıtım Ticaret ve Sanayi A.Ş.'nin markasıdır.

twitter.com/tellekt • facebook.com/tellekt • instagram.com/tellekt

YAPAY ÖĞRENME:

YENİ YAPAY ZEKÂ

ETHEM ALPAYDIN

ÇEVİRİ:
AYLİN AĞAR

Tellekt

ETHEM ALPAYDIN, lisans derecesini 1987'de Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü'nden, doktora derecesini ise 1990'da İsviçre, École Polytechnique Fédérale de Lausanne, Bilgisayar Bilimleri Bölümü'nden almıştır. 1991-2019 yılları arasında Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü'nde öğretim üyesi ve bu süre içinde yurtdışında farklı üniversitelerde misafir araştırmacı olarak çalışmıştır. *Yapay Öğrenme: Yeni Yapay Zekâ* kitabı 2016 yılında MIT Press tarafından basılmış; Çince, Japonca, Korece ve Rusçaya çevrilmiştir. *Introduction to Machine Learning* (Yapay Öğrenmeye Giriş) adlı kitabının dördüncü baskısı 2020 yılında yapılmış, bu kitabın önceki baskıları da Çince, Almanca ve Türkçeye çevrilmiştir. Ethem Alpaydın, Bilim Akademisi üyesidir ve Özyeğin Üniversitesi Bilgisayar Mühendisliği Bölümü'nde öğretim üyesi olarak çalışmaktadır.

AYLİN AĞAR, 1982'de İstanbul'da doğdu. İstanbul Üniversitesi İngilizce Mütercim Tercümanlık Bölümü'nden mezun olduktan sonra yüksek lisans eğitimini Boğaziçi Üniversitesi'nde çeviribilim alanında tamamladı. Mesleki İngilizce dersleri vermekte ve yükseköğretim kurumlarında okutman olarak görev yapmaktadır.

İÇİNDEKİLER

TEŞEKKÜR	9
ÖNSÖZ	11
1. YAPAY ÖĞRENMEYLE NEDEN İLGİLENİYORUZ?	17
2. YAPAY ÖĞRENME, İSTATİSTİK VE VERİ ANALİZİ	37
3. ÖRÜNTÜ TANIMA	55
4. YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME	77
5. ÖBEKLERİ VE ÖNERİLERİ ÖĞRENME	93
6. EYLEM YAPMAYI ÖĞRENME	103
7. GELECEKTE BİZİ NE BEKLİYOR?	113
SÖZLÜKÇE	133
İLERİ OKUMALAR	143
KAYNAKÇA	145
DİZİN	149

TEŞEKKÜR

Yapıcı yorum ve tavsiyeleri için anonim hakemlere minnettarım. MIT Press'le çalışmak her zaman olduğu gibi bir zevkti. Ayrıca Kathleen Caruso, Kathleen Hensley ve Marie Lufkin Lee'ye destekleri için teşekkür ederim.

Bu kitabı Türkçeye çeviren Aylin Ağar'a ve başta Didem Bayındır olmak üzere kitabı yayımlayan Tellekt çalışanlarına teşekkür ederim.

ÖNSÖZ

Son yirmi yıldır bilgisayar bilimlerinde sessiz bir devrim gerçekleşiyor. Bugünlerde, giderek daha çok sayıda "öğrenen bilgisayar programı", yani görevinin gereklerine daha iyi uyması için davranışlarını otomatik olarak uyarlayabilen yazılım görüyoruz. Artık insanları yüzlerinden tanımayı öğrenen, konuşmayı anlayan, otomobil süren ya da film öneren programlarımız var. Gelecekte daha da fazlasını yapmayı vaat ediyorlar.

Bir zamanlar, bilgisayara ne yapması gerektiğini bir programlama dilinde program yazarak tanımlayan programcıydı. Şimdi ise bazı görevler için program yazmak yerine veri topluyoruz. Veri, ne yapılması gerektiğini söyleyen örnekler içeriyor; öğrenme algoritması da öğrenen programı, bu verinin tanımladığı gerekliliklerle uyuşacak biçimde otomatik olarak ayarlıyor.

Bilgisayarların geçen yüzyılın ortasında ortaya çıkışından bu yana, yaşamlarımız giderek artan biçimde sayısallaşıyor. Artık bilgisayarlar, bir zamanlar oldukları gibi salt hesap makinesi değiller. Veritabanları [*database*] ve sayısal ortam, bilgi saklamanın ana ortamı olarak kâğıda baskı almanın yerine geçti ve bilgisayar ağları üstünden sayısal iletişim, bilgi aktarımının ana biçimi olarak postayı gölgede bıraktı. Önce kullanımı kolay grafik arayüze sahip kişisel bilgisayarlar, ardından telefon ve diğer akıllı aygıtlar içinde gömülü halde, bilgisayar her yerde bulunan, televizyon ya da mikrodalga fırın

gibi bir ev eşyası haline geldi. Şimdilerde, yalnızca rakamlar ve harfler değil, görüntü, video, ses ve benzeri her tür bilgi sayısal olarak saklanıyor, işleniyor ve çevrimiçi bağlantı sayesinde bir yerden bir yere aktarılabiliyor. Tüm bu sayısal işlem çok miktarda veri doğuruyor. Veri analizi ile yapay öğrenmeye olan yaygın ilginin tetiklenmesine yol açan işte bu veri taşmasıdır; buna "veri sarsıntısı" [*dataquake*] da diyebiliriz.

Görüntü tanımadan konuşma tanımaya, çeviriden robotiğe çok sayıda uygulama için 1950'lerden beri yapılan onlarca yıllık araştırmaya rağmen yeterince iyi algoritma bulamadık. Fakat tüm bu görevler için veri toplamak kolay ve buradaki düşünce, programcıları öğrenen programlarla değiştirmek ve bu görevler için gereken algoritmaları veriden otomatik olarak öğrenmektir. Bu yapay öğrenmenin alanıdır ve son yirmi yılda yalnızca veri sürekli olarak büyümekle kalmadı, bu veriyi bilgiye dönüştürmeye yönelik yapay öğrenme teorisi de önemli ölçüde gelişti.

Bugün, perakende ve finanstan imalata, farklı iş türlerinde sistemlerimiz bilgisayarlaştıkça, durmaksızın daha fazla veri üretilip toplanıyor. Bu durum astronomiden biyolojiye kadar bilimin çeşitli alanları için de geçerli. Günlük yaşamlarımızda da sayısal teknoloji varoluşumuza sızdıkça, sayısal ayak izimiz derinleştikçe, yalnızca tüketiciler ve kullanıcılar olarak değil, sosyal medya aracılığıyla da yaşamlarımızın giderek artan bir kısmı kaydediliyor ve veri haline geliyor. Kaynağı ne olursa olsun –ticari, bilimsel ya da kişisel– pasif bir şekilde öylece duran veri bir işe yaramaz ve akıllı insanlar her geçen gün bu veriyi işe yarar kılmanın ve kullanışlı bir ürüne ya da hizmete dönüştürmenin yeni yollarını buluyorlar.

Tüm bu görünüşte karmaşık ve hacimli verinin berisinde basit bir açıklamanın yattığı inancındayız. Verinin, büyük olmasına rağmen, az sayıda saklı etken ve bu etkenlerin etkileşimini içeren görece basit bir modelle açıklanabileceğini düşünüyoruz. Çevrimiçi olarak ya da yerel süpermarketlerden her gün binlerce ürün alan milyonlarca tüketiciyi düşünelim. Bu oldukça büyük bir işlem veritabanı demektir. Fakat bizi kurtaran ve yardımcı olan gerçek, bu verilerin bir örüntüye sahip olmasıdır. İnsanlar rasgele alışveriş yapmaz.

Parti veren biri belli bir ürün altkümesini, evde bebeği olan biriyse farklı bir altkümeyi satın alır. Tüketici davranışını açıklayan bu tür saklı etkenler vardır. Yapay öğrenmenin merkezinde olan da bu, yani gözlenen veriden saklı bir modelin –altta yatan etkenlerin ve onların veriyi oluşturmadaki etkileşiminin– çıkarılmasıdır.

Yapay öğrenme, yalnızca veriden bilgi çıkaran ticari uygulama yöntemleri değildir: Öğrenme ayrıca zekânın bir gereğidir. Zeki bir sistem çevresine uyum sağlayabilir olmalıdır; hatalarını tekrar etmemeyi, başarılarını tekrar etmeyi öğrenmelidir. Önceleri araştırmacılar yapay zekânın gerçekliğe dönüşmesi için yeni bir paradigmaya, yeni bir düşünme tarzına, yeni bir hesaplama [*computation*] modeline ya da bütünüyle yeni bir algoritma kümesine ihtiyacımız olduğuna inanıyorlardı. Farklı alanlarda yapay öğrenmenin yakın zamanlı başarıları dikkate alındığında, ihtiyacımız olanın yepyeni algoritmalar değil, içinden doğru kuralları çıkarmayı sağlayacak kadar çok veri ve bu kadar çok veriyi işleyebilecek yeterlilikte bilgisayar gücü olduğu iddia edilebilir.

Yapay çeviri ve planlama gibi uygulamaların, göreli olarak basit ancak büyük miktarda veri örneğinden öğrenebilen böyle yöntemlerle gerçeklenebileceği varsayılabilir. Derin öğrenmedeki [*deep learning*] yakın zamanlı başarılar da bu iddiayı destekliyor. Zekâ, bazı karmaşık formüllerden değil; basit, anlaşılır algoritmaların sabırlı ve sürekli kullanımından kaynaklanıyor gibi görünüyor.

Teknoloji geliştikçe ve biz daha hızlı bilgisayarlara ve daha fazla veriye sahip oldukça, öğrenen algoritmalar yavaş yavaş daha yüksek zekâ üretecek ve gittikçe daha zeki aygıtlarda ve yazılımlarda kullanım alanı bulacaklar gibi görünüyor. Bu tür öğrenilen zekânın, bu yüzyıl sona ermeden insan zekâsı düzeyine erişmesi şaşırtıcı olmayacaktır.

Ben bu kitap üzerinde çalışırken, en prestijli bilim dergilerinden biri olan *Science*, 15 Temmuz 2015 tarihli sayısında (cilt 349, sayı 6245), yapay zekâ üzerine özel bir bölüm yayımladı. Başlık, yapay zekâya odaklanılacağını duyursa da asıl tema yapay öğrenmeydi. Bu da yapay öğrenmenin artık yapay zekâda itici güç olduğunu göste-

ren bir başka işarettir. Mantık temelli, programlanmış uzman sistemlerin 1980'lerde yarattığı hayal kırıklığının ardından yapay öğrenme, kayda değer sonuçlar sağlayarak alanı yeniden canlandırmıştır.

Bu kitabın amacı okura, yapay öğrenmenin ne olduğu, bazı önemli öğrenme algoritmalarının temelleri ve bir dizi örnek uygulama hakkında genel bir fikir vermektir. Kitap genel okura yöneliktir ve herhangi bir matematiksel ya da programlama ayrıntısı verilmeksizin yalnızca öğrenme yöntemlerinin anafikirleri tartışılmaktadır. Kitapta yapay öğrenme uygulamaları da aşırı ayrıntılı olarak verilmemekte ve bir dizi örnek, yalnızca konunun ana noktalarını aktarmaya yetecek biçimde ele alınmaktadır.

Okur, yapay öğrenme algoritmaları hakkında daha fazla bilgi edinmek için, aynı konu üzerine yazdığım ve büyük oranda bu kitabın da dayanağı olan ders kitabıma başvurabilir.*

Elinizdeki kitabın içeriği şu şekilde düzenlenmiştir:

Birinci bölümde yapay öğrenmeye olan ilgiyi yaratan şu anki durumu bir bağlama yerleştirmek için, bilgisayar bilimi ile uygulamalarının zaman içindeki gelişimini kısaca ele alıyoruz. Yani sayısal teknolojinin yoğun hesaplamalı büyük bilgisayarlardan [*number-crunching mainframes*] masaüstü kişisel bilgisayarlara ve ardından çevrimiçi ve mobil akıllı aygıtlara evrimini inceliyoruz.

İkinci bölüm, yapay öğrenmenin temellerini tanıtıyor ve bazı basit uygulamalar üzerinde model uydurma ve istatistikle nasıl ilişkili olduğunu ele alıyor.

Çoğu yapay öğrenme algoritması gözetimlidir [*supervised*] ve üçüncü bölümde böyle algoritmaların yüz ve konuşma gibi örüntüleri tanımada nasıl kullanıldığından bahsediyoruz.

Dördüncü bölüm, insan beyninden esinlenen yapay sinir ağlarını [*artificial neural networks*], nasıl eğitildiklerini ve "derin" çok kat-

* Ethem Alpaydın, *Introduction to Machine Learning*, MIT Press, Cambridge, MA, 2014. [Aksi belirtilmedikçe dipnotlar yazara aittir. (Y.N.)]

manlı ağların farklı soyutlama düzeylerindeki hiyerarşileri nasıl öğrenebildiklerini ele alıyor.

Başka bir yapay öğrenme türü, örnekler arasındaki bağıntıların öğrenilmesini amaçlayan gözetimsiz öğrenmedir. Beşinci bölümde, bu türün yaygın uygulamalarına örnek olarak bölütleme ve önerilerin öğrenilmesinden bahsediyoruz.

Altıncı bölüm, otonom bir etmenin –sözgelimi sürücüsüz [*self-driven*] bir otomobilin– belli bir çevre içinde ödülü en yüksek, cezayı en az yapacak biçimde eylem yapmayı öğrendiği pekiştirmeli öğrenme üzerinde duruyor.

Yedinci bölüm, yapay bulut hesaplamayı da içeren "veri bilimi" alanını tartışarak kitabı sonlandırıyor. Bu bölümde ayrıca bu alanın, veri mahremiyeti ve güvenliği gibi etik ve yasal boyutlarını ele alıyoruz.

Bu kitabın amacı, yapay öğrenmede günümüzde ne olup bittiği üzerine kısa bir giriş sunmaktır ve umudum gelecekte ne yapılabileceği konusunda okuru düşünmeye teşvik etmektir. Yapay öğrenme kesinlikle günümüzün en heyecan verici bilimsel alanlarından biridir, çeşitli alanlarda teknolojiyi geliştirmiştir ve halihazırda tüm yaşam alanlarını etkileyen çarpıcı uygulamalar üretmiştir. Ben bu kitabı yazmaktan çok keyif aldım, umarım siz de okurken keyif alırsınız!

1

YAPAY ÖĞRENMEYLE NEDEN İLGİLENİYORUZ?

Sayısalın Gücü

Son yarım yüzyılda yaşamlarımızdaki en büyük dönüşümlerden bazıları bilgi işlem ve sayısal teknoloji sayesinde oldu. Önceki yüzyıllarda icat ettiğimiz ve geliştirdiğimiz araç, aygıt ve hizmetler, her gün artan oranda bilgisayarlaşmış "e-" sürümleriyle yer değiştirmekte ve biz de aynı zamanda bu yeni sayısal çevreye uyum sağlamaktayız.

Bu dönüşüm çok hızlı oldu: Bir zamanlar –her şeyin ışık hızında gerçekleştiği sayısal alanda elli yıl efsanevi bir geçmiştir– bilgisayarlar pahalıydı ve yalnızca hükümetler, büyük şirketler, üniversiteler, araştırma merkezleri gibi büyük kuruluşlar onlara sahip olabiliyordu. O zamanlar yalnızca bu kuruluşların bilgisayar satın alıp işletmenin getirdiği yüksek maliyeti haklı çıkarabilecek ölçüde önemli işleri vardı. Ayrı katlar ya da binalardaki bilgisayar "merkez"leri bu güç düşkünü canavarlara ev sahipliği yaparken büyük salonların içinde manyetik bantlar döndürülüyor, kartlar deliniyor, sayılar hesaplanıyordu; o zamanlar hatalar gerçekten böcekti.*

* Yazar burada bilgisayar biliminde "hata" anlamına gelen "bug" sözcüğünün asıl anlamıyla (böcek) sözcük oyunu yapıyor. (Ç.N.)

Bilgisayarlar ucuzladıkça nüfusun daha büyük bir kesimi için erişilebilir oldular ve bununla birlikte uygulama alanları genişledi. Başlangıçta bilgisayarlar yalnızca hesap makineleriydi. Yeni sayılar elde etmek için sayıları topluyor, çıkartıyor, çarpıyor ve bölüyorlardı. Hesaplama teknolojisinin büyük itici gücünün, her bilgi [*information*] parçasının sayı olarak temsil edilebileceğinin anlaşılması olduğunu söyleyebiliriz. Bu da o zamana dek yalnızca sayıları işlemekte kullanılan bilgisayarın *her* türden bilgiyi işlemek için kullanılabileceğini göstermiştir.

Daha kesin bir ifadeyle, bir bilgisayar her sayıyı 0 ya da 1'den oluşan belirli bir ikili hane [*digit*] (bit) dizisi olarak temsil eder ve bu tür bit dizileri başka bilgi türlerini de temsil edebilir. Örneğin "101100", ondalık sayı 44'ü temsil etmek için kullanılabilir *ve* ayrıca virgülün de kodudur. Benzer şekilde, "1000001" hem 65 hem de büyük "A" harfidir.* Bilgisayar programı bağlama göre diziyi nasıl yorumlayacağını bilir ve ona uygun olarak işler.

Aslında böyle bit dizileri yalnızca sayıları ve metinleri değil, bambaşka bilgi türlerini, örneğin bir fotoğraftaki renkleri ya da bir şarkıdaki tonları da temsil edebilir. Bilgisayar programlarının kendileri de bit dizileridir. En az bunun kadar önemli bir başka nokta da bir görüntüyü daha parlak yapmak ya da bir fotoğraftaki yüzü bulmak gibi bu bilgi türleriyle ilişkili işlemlerin de bit dizilerini işleyen komutlara çevrilebilmesidir.

Bilgisayarlar Veri Saklar

Bilgisayarın gücü, her bilgi parçasının sayısal –yani bir bit dizisi– olarak temsil edilebilmesi ve her türden bilginin işlenmesinin böylesi sayısal gösterimleri [*representation*] işleyen bilgisayar komutları olarak yazılabilmesi gerçeğinde yatar.

* Burada İngilizce alfabe ve noktalama işaretleri için tasarlanan ASCII kodunu kullanıyoruz. Günümüzde kullandığımız karakter kodlama şemaları, farklı dillerin farklı alfabelerini kapsıyor.

Bunun sonuçlarından biri 1960'larda ortaya çıkan *veritabanlarıdır.* Bunlar *veriyi* –sayısal olarak temsil edilmiş bilgiyi– saklayıp işleyen özelleştirilmiş bilgisayar programlarıdır. Bant ya da disk gibi çevre aygıtları bitleri manyetik olarak kaydettiği için bilgisayarın elektriği kesildiğinde içerikleri silinmez.

Veritabanlarıyla birlikte bilgisayarlar, işlemenin ötesine geçmiş ve sayısal gösterimli bilgiyi saklama araçları haline gelmiştir. Sayısal ortam zamanla öylesine hızlı, ucuz ve güvenilir oldu ki, insanlığın ana bilgi saklama aracı olan kâğıda baskının yerine geçti.

Mikroişlemcinin [*microprocessor*] icadı ve elektronik devrelerde küçükleştirmeyle birlikte maliyetler düştü ve bu sayede 1980'lerin başlarından itibaren kişisel bilgisayarlar giderek daha erişilebilir oldu. *Kişisel bilgisayar*, hesaplamayı küçük işletmeler için erişilebilir kıldı. Fakat en önemlisi, kişisel bilgisayar ev aleti olmasına yetecek kadar küçük ve ucuzdu. Artık büyük bir şirket olmanıza gerek yoktu, bilgisayar sizin de hayatınızda yardımcı bir rol üstlenebilirdi. Kişisel bilgisayar herkesin bilgisayarda yapılmaya değer işleri olduğunu gösterdi ve sayısal teknolojinin demokratikleştiği bu dönemde uygulamalar da bu teknolojiyle birlikte büyüdü.

Grafik kullanıcı arayüzleri [*graphical user interface*] ve fare, kişisel bilgisayarın kullanımını kolaylaştırdı. Bunlar sayesinde bilgisayarı kullanmak için programlama öğrenmemize, farklı bir sözdizimindeki komutları hatırlamamıza gerek kalmıyordu. Dosyaları, ikonları ve hatta çöp kutusuyla bir sanal masaüstünü gösteren ekran, çalışma alanımızın sayısal bir canlandırmasıdır. Fare ise okumak ya da düzenlemek için bu unsurları tutan sanal elimizdir.

Bu gelişmelere paralel olarak kişisel bilgisayar yazılımları, daha çok veri türünü işleyerek ve yaşamlarımızı daha fazla sayısallaştırarak ticari uygulamalardan kişisel uygulamalara yöneldi. Artık mektuplarımız ve diğer kişisel belgelerimiz için bir sözcük işlemcimiz, evimizle ilgili hesaplarımız için bir elektronik tablomuz, müzik ya da fotoğrafçılık gibi hobilerimiz için yazılımlarımız var. İstersek oyun bile oynayabiliriz! Hesaplama gündelik ve eğlenceli hale geldi.

Böyle bir gündelik uygulamalar paletine sahip zevkli ve davet-

kâr kullanıcı arayüzüyle kişisel bilgisayar, insanlarla bilgisayarların, o zamana dek bildiğimiz yaşamımızla sayısal dünyanın yakınlaşmasında büyük bir adım niteliği taşıyordu. Bir yandan bilgisayarlar yaşamlarımıza biraz daha iyi uyacak şekilde programlanırken öte yandan biz de onlarla yaşamaya biraz daha uyum sağladık. Zamanla bilgisayar kullanmak, otomobil sürmek gibi temel bir beceri haline geldi.

Kişisel bilgisayar, bilgisayarları kitleler için erişilebilir kılma yolunda atılan ilk adımdı. Sayısal teknolojiyi yaşamımızın daha büyük bir parçası haline getirdi ve bu kitaptaki öykümüz açısından en önemlisi, yaşamlarımızın daha büyük kısmının sayısal olarak kaydedilmesini mümkün kıldı. Bu özelliğiyle kişisel bilgisayar, yaşamlarımızı veriye, analiz edip içinden bir şeyler öğrenebildiğimiz veriye dönüştürme sürecinde önemli bir sıçrama tahtasıydı.

Bilgisayarlar Veri İletir

Hesaplamada yaşanan bir sonraki büyük gelişme, bağlanırlık [*connectivity*] konusunda oldu. Bilgi iletişimi için bilgisayarları veri linkleriyle birbirine bağlamak daha önce yapılmış olsa da kişisel bilgisayarları telefon ya da özel hatlarla birbirine ya da merkezî sunuculara bağlayan ticari sistemler 1990'larda yaygınlaşmaya başladı.

Bilgisayar ağı, bir bilgisayarın artık yalıtılmış olmaması ve uzaktaki bir bilgisayarla veri iletişimi yapabilmesi anlamına gelir. Bir kullanıcı artık kendi bilgisayarındaki kişisel verisiyle kısıtlanmış değildir, herhangi bir yerdeki veriye erişebilir ve eğer isterse kendi verisini de başka kullanıcıların erişimine açabilir.

Bilgisayar ağlarının gelişmesiyle çok hızlı biçimde internet ortaya çıktı. İnternet, tüm küreyi sarmalayan *ana* bilgisayar ağıdır. İnternet, dünyada bilgisayara erişimi olan herhangi bir kişinin başka herhangi birine bilgi, örneğin e-posta göndermesini mümkün kıldı. Zaten artık verilerimiz ve aygıtlarımız sayısal olduğundan, paylaşabildiğimiz bilgi sadece metinler ve sayılarla da sınırlı değil; görüntü, video, müzik, konuşma ya da başka herhangi bir şey gönderebiliriz.

Sayısal olarak temsil edilen bilgi nerede olursa olsun herhangi bir kişiye bilgisayar ağları üzerinden ışık hızında gönderilebilir. Bilgisayar artık yalnız verinin işlendiği ve saklandığı bir makine değildir, bilgi aktarmanın ve paylaşmanın da bir aracı haline gelmiştir. Zaman içinde bağlanırlık öyle hızlı büyüdü, sayısal iletişim öylesine ucuz, hızlı ve güvenilir hale geldi ki, sayısal aktarım bilgi aktarımının ana teknolojisi olarak postanın yerini almakta.

"Çevrimiçi" olan biri, kendi bilgisayarındaki kişisel veriyi ağ üzerinde başkaları için erişilebilir kılabilir; Dünya Çapında Ağ da [*World Wide Web – www*] böyle doğmuştur. İnsanlar web üzerinde "gezinebilir" ve bu paylaşılan bilgiye göz gezdirebilir. Gizli bilgileri paylaşmak ve böylece web üzerinden çevrimiçi alışveriş veya bankacılık gibi ticari işlemleri yapabilmek için güvenli protokoller hızla ortaya çıktı. Çevrimiçi bağlanırlık, sayısal teknolojinin nüfuzunu daha da artırdı. "www" hizmet sağlayıcı geçidini kullanarak çevrimiçi bir hizmet aldığımızda bilgisayarımız, mağazanın, bankanın, kütüphanenin ya da üniversitenin sayısal sürümüne dönüşür. Bu da daha fazla veri yaratır.

Mobil Bilgi İşlem

Zaman içinde bilgisayarlar gittikçe küçüldü ve 1990'ların ortalarında pil teknolojisinde yaşanan gelişmelerle birlikte taşınabilir –"laptop" ya da "notebook" denilen– bilgisayarlar yaygınlaşmaya başladı ve böylece yeni bir mobil bilgi işlem [*mobile computing*] çağı başladı. Cep telefonları da yaklaşık olarak aynı zamanlarda yaygınlaşmaya başladı ve 2005 yılı civarında bu iki teknoloji akıllı telefonlarda birleşti.

Akıllı telefon, aynı zamanda bilgisayar da olan bir telefondur. Zamanla akıllı telefon daha akıllı –daha fazla bilgisayar, daha az telefon– hale geldi; o kadar ki, günümüzde telefon, akıllı telefondaki birçok uygulamadan yalnızca biri ve nadiren kullanılanlardandır. Geleneksel telefon akustik bir cihazdı: Konuşurdunuz ve öteki taraftaki kişinin konuştuğunu duyardınız. Akıllı telefon bugünse daha

çok görsel bir cihazdır: Büyük bir ekranı vardır ve zamanımızın çoğunu konuşarak değil bu ekrana bakarak ya da duyarlı yüzeyine dokunarak geçiririz.

Akıllı telefon daima çevrimiçi olan bir bilgisayardır* ve kullanıcısının hareket halindeyken her türlü bilgi için internete erişebilmesini sağlar. Bu nedenle –sözgelimi seyahat ederken– başka bilgisayarlardaki verilere daha büyük bir erişim olanağı sağlayarak bağlanırlığımızı artırır ve ayrıca bizi ve verilerimizi başkaları için daha erişilebilir kılar.

Akıllı telefonu özel yapan şey, onun aynı zamanda bir mobil algılama cihazı olmasıdır ve hep üzerimizde bulunduğu için kullanıcısı hakkındaki bilgileri, örneğin konumunu sürekli olarak kaydeder ve bu verileri kullanıma sunabilir. Akıllı telefon bizi tespit edilebilir, izlenebilir, kaydedilebilir kılan bir mobil sensördür.

Bilgisayarın bu artırılmış hareketliliği [*mobility*] yeni bir olgudur. Bir zamanlar bilgisayar büyüktü ve bir "bilgisayar merkezi"ndeydi; sabitti ve bizim ona yürümemiz gerekiyordu. Bilgisayarı kullanmak için bir terminalin önüne oturuyorduk, buna "terminal"** deniyordu çünkü bilgisayar orada bitiyordu. Sonra departmanımıza daha küçük bir bilgisayar geldi ve ardından daha küçüğü ofisimizdeki veya evimizdeki masamıza kuruldu; daha sonra bundan da küçük olanı kucağımızdaydı ve şimdi cebimizde ve hep bizimle.

Bir zamanlar çok az sayıda bilgisayar vardı; binlerce kişiye, örneğin şirket ya da kampüs başına bir tane bilgisayar düşüyordu. Sözünü ettiğimiz, kişi başına düşen bilgisayar oranı çok hızla arttı ve kişisel bilgisayarla herkesin bir bilgisayar sahibi olması hedeflendi. Bugün kişi başına düşen pek çok bilgisayarımız var. Artık tüm cihazlarımız aynı zamanda birer bilgisayar ya da içlerinde bir bilgisayar bulunuyor. Telefonunuz aynı zamanda bir bilgisayardır, televizyonu-

* Hesaplama gücü, saklama kapasitesi ya da bağlanırlık kendi başına katma değer üretmez; tıpkı daha yüksek bir nüfusun ille de daha büyük bir işgücü anlamına gelmemesi gibi. Gelişmekte olan ülkelerdeki çok sayıda akıllı telefon zenginlik anlamına gelmiyor.

** "Terminal" sözcüğü Latince "*terminus*" sözcüğünden türetilmiş olup "son", "son ek", "son yer", "son durak", "uç", "hedef", "uçbirim" gibi anlamlara gelmektedir. (Ç.N.)

nuz da bir bilgisayardır, otomobilinizin içinde farklı işlevler gören birçok bilgisayar vardır ve müzik çalarınız, tıpkı kameranız veya kol saatiniz gibi özelleşmiş bir bilgisayardır. *Akıllı cihaz*, ilk başta yaptığı şeyin sayısal sürümünü gerçekleyen bir bilgisayardır.

Her yerde her an bilgi işlem [*ubiquitous computing*] giderek yaygınlaşan bir terimdir ve farkında bile olmadan bilgisayar kullanmak anlamına gelir. Onlara açıkça bilgisayar adını vermeden, her zaman her türlü amaç için çok sayıda bilgisayar kullanmak demektir. Sayısal sürümün, hız, doğruluk ve kolay uyarlanabilirlik gibi genel üstünlükleri vardır. Ancak bir başka üstünlüğü de cihazın sayısal sürümünün artık tüm verilerine sayısal formatta sahip olmasıdır. Dahası, eğer çevrimiçi ise, başka çevrimiçi cihazlarla konuşabilir ve verilerini neredeyse ânında kullanılabilir hale getirebilir. Onlara "akıllı nesneler" veya sadece "şeyler" diyoruz ve "Nesnelerin İnterneti" (çevrimiçi eşya) [*Internet of Things*] deyimini kullanıyoruz.

Sosyal Veri

Birkaç bin yıl önce, resminizin çizilmesini, heykelinizin yapılmasını veya hikâyenizin hatırlanıp anlatılmasını istiyorsanız, bir tanrı veya tanrıça olmanız gerekiyordu. Bin yıl önce bir kral veya kraliçe, birkaç yüzyıl önce ise zengin bir tüccar olmanız ya da böyle birinin evinde yaşamanız gerekiyordu. Artık herkesin, hatta bir çorba konservesinin bile resmi yapılabilir. Benzer bir demokratikleşme bilgisayar ve veride de yaşandı. Bir zamanlar sadece büyük kuruluşlar ve işletmeler bilgisayarda yapılmaya değer işlere sahipti ve bu yüzden de sadece onların verileri vardı; kişisel bilgisayarla birlikte insanlar ve hatta nesneler veri üreticileri haline geldi.

Yeni bir veri kaynağı da sosyal etkileşimlerimizin sayısal hale geldiği sosyal medyadır. Bunlar, artık toplanabilen, saklanabilen ve analiz edilebilen başka bir veri türü oluşturur. Sosyal medya, agorada, meydanda, pazarda, kahvehanede ve barda yapılan tartışmaların ya da çeşme, su kuyusu ve su sebili etrafında toplanmanın yerini alıyor.

Sosyal medyayla birlikte her birimiz artık hayatı takip edilmeye değer birer ünlüyüz ve kendi kendimizin paparazzisiyiz. Artık bize yalnızca on beş dakikalık bir ün vaat edilmiyor, çevrimiçi olduğumuz her an ünlüyüz. Sosyal medya, sayısal otobiyografimizi yaşarken yazmamıza izin veriyor. Eski zamanlarda kitaplar ve gazeteler pahalıydı ve dolayısıyla nadir bulunurdu; bu yüzden yalnızca önemli hayatların izini sürüp hikâyesini anlatabilirdik. Şimdiyse veri ucuz ve hepimiz kendi küçük çevrimiçi krallıklarımızın kral ve kraliçeleriyiz. Bugün, aygıtlara meraklı ebeveynlerin eline doğan bir bebek henüz ilk ayında, Homeros'un Odysseus'un tüm maceralarını anlatmak için kullandığından daha fazla veri üretebilir.

Tüm Bu Veri: Veri Depremi

Bütün bu bilgisayarlı makinelerimiz ve hizmetlerimiz tarafından üretilen veriler eskiden sayısal teknolojinin bir yan ürünüydü ve bilgisayar bilimciler, büyük miktarda veriyi verimli saklamak ve işlemek için veritabanları üstüne birçok araştırma yaptılar. Veriyi sakladık çünkü buna zorunluyduk. Son yirmi yılda ise tüm bu veri bir kaynağa dönüştü; artık daha fazla veri bir nimet olarak kabul ediliyor.

Sözgelimi, her gün binlerce ürünü milyonlarca müşteriye, tümü ülkeye yayılmış sayısız fiziksel mağazalarının birinde ya da web üzerindeki sanal bir mağaza aracılığıyla satan bir süpermarket zincirini düşünelim. Satış noktası terminalleri sayısaldır ve her işlemi ayrıntılarıyla kaydederler: örneğin, müşteri kimliği (bazı sadakat programları aracılığıyla), satın alınan mallar, ne kadara alındığı, harcanan toplam para, vb. Mağazalar çevrimiçi olarak bağlıdır ve tüm mağazalardaki bütün terminallerden gelen veriler ânında merkezdeki bir veritabanında toplanabilir. Bu, her gün çok yüksek miktarda (ve çok güncel) veri demektir.

Özellikle yaklaşık son yirmi yılda insanlar giderek kendilerine tüm bu veriyle ne yapabileceklerini sormaya başladılar. Bu soruyla bilgi işlemenin yönü tersine çevrildi. Önceden veri, programların işlediği ve çıkardığı şeydi, yani veri pasifti. Bu soruyla birlikte veri

operasyonu yönlendirmeye başladı; böylece ne yapılacağını tanımlayan artık programcılar değil veri oldu.

Bir süpermarket zinciri, hangi müşterinin hangi ürünü satın almaya yatkın olduğunu bilmek ister. Bu bilgiyle daha verimli bir şekilde stok yapabilir, bu da satışları ve kârı artırır. Bununla ayrıca müşteri memnuniyeti de artacaktır, çünkü bu sayede müşteriler de ihtiyaçlarına en uygun ürünleri daha hızlı ve ucuza bulabilecektir.

Bu işin kesin bir tanımı yoktur. Tam olarak hangi insanların şu dondurma aromasını ya da şu yazarın yeni kitabını satın alacağını, şu filmi seyredeceğini ya da şu şehri ziyaret edeceğini bilemeyiz. Müşteri davranışı zaman içinde değişir ve coğrafi konuma bağlıdır.

Fakat yine de umut vardır, çünkü müşteri davranışının tamamen rasgele olmadığını da biliyoruz. İnsanlar süpermarkete gidip rasgele şeyler satın almaz. Veride belli örüntüler vardır: bira alırken cips de alırlar; yazın dondurma, kışın boza alırlar. Müşteri davranışında bir düzen bulunur ve veri bu noktada devreye girer.

Müşteri davranışı örüntülerini bilmememize rağmen, onları toplanan veride görmeyi bekleriz. Bu tür örüntüleri geçmiş veride bulabilirsek geleceğin, en azından yakın geleceğin bu verinin toplandığı geçmişten çok farklı olmayacağı varsayımıyla söz konusu örüntülerin geçerliliklerini sürdürmesini bekleyebiliriz ve onlara dayanarak tahminlerde bulunabiliriz.

Süreci tam olarak tanımlayamasak da iyi ve kullanışlı bir yaklaşıklama [*approximation*] oluşturabileceğimize inanırız. Bu yaklaşıklama her şeyi olmasa da verinin önemli bir kısmını açıklayabilir. Bütün süreci belirlemek mümkün olmasa da bazı örüntüleri saptayabileceğimize inanırız. Tahmin yürütmek için bu örüntüleri kullanabiliriz; bunlar ayrıca süreci anlamamıza da yardım edebilir.

Buna *veri madenciliği* denir. Büyük hacimde toprağın ve hammaddenin işlenip içinden az miktarda çok değerli madenin çıkarılması gibi veri madenciliğinde de yüksek hacimde veri işlenerek işe yarayan, örneğin yüksek tahmin başarısına sahip olan bir model oluşturmak amaçlanmaktadır.

Veri madenciliği bir tür yapay öğrenmedir. Müşteri davranışlarını açıklayan kuralları bilmiyoruz ve bu yüzden doğrudan bir program yazamayız, fakat bilgisayar veriden böylesi kuralları çıkararak "öğrenebilir".

Kuralları bilmediğimiz ancak çok fazla veriye sahip olduğumuz birçok uygulama var. İşletmelerimizin bilgisayar ve sayısal teknoloji kullanması, her tür alanda büyük miktarda veri bulunduğunu gösterir. Günlük sosyal hayatımızda da bilgisayar veya akıllı cihazlar kullanıyoruz, dolayısıyla bunun için de verimiz var.

Öğrenen modeller örüntü tanımada, örneğin bir kamera tarafından kaydedilen görüntüleri veya bir mikrofon tarafından kaydedilen konuşmayı tanımada kullanılır. Bugünlerde akıllı telefonlarla insan eylemlerinin tanınmasından otomobillerdeki sürüş destek sistemlerine kadar farklı uygulamalar için kullandığımız farklı türdeki sensörlere sahibiz.

Başka bir veri kaynağı da bilimdir. Daha iyi sensörler oluşturdukça, astronomide, biyolojide, fizikte vb. algı ve saptama kapasitemiz artar yani daha fazla veri elde ederiz ve giderek daha büyük hale gelen bu verileri anlamak için öğrenen algoritmalara ihtiyaç duyarız. İnternetin kendisi büyük bir veri deposudur ve aradığımızı bulmak için akıllı algoritmalara ihtiyacımız vardır. Bugün sahip olduğumuz verinin önemli bir özelliği farklı kiplerden gelmesidir, yani veri çoğulortamlıdır [*multimedia*]. Metinlerimiz yanında resimlerimiz veya videolarımız, ses kayıtlarımız vb. olabilir ve bunların hepsi de ilgilendiğimiz aynı nesne veya olayla ilgili bilgi verebilir; bugün yapay öğrenmedeki önemli araştırma alanlarından biri bu tür farklı kaynaklardan gelen bilgileri karar verirken birleştirmektir. Örneğin tüketici verileri analizinde, geçmiş işlemlere ek olarak aynı zamanda kullanıcının web günlüklerine, yani bir kullanıcının yakın zamanda ziyaret ettiği web sayfalarının bilgisine de sahip olabiliriz ve bu günlükler oldukça bilgilendirici olabilir.

Günlük yaşamımızda bize sürekli yardımcı olan akıllı makinelerin sayısı arttıkça hepimiz veri üreticisine dönüşüyoruz. Ne zaman bir ürün alsak, bir film kiralasak, bir web sayfasını ziyaret etsek, bir

blog yazsak ya da sosyal medyada paylaşım yapsak, hatta yürürken veya otomobilimizde gezerken bile veri üretiyoruz. Ve bu veri, onu toplamak ve analiz etmekle ilgilenen birileri için değer taşıyor. Müşteri, daima haklı olmasının yanında ilgi çekici ve takip edilmeye değerdir.

Her birimiz verinin yalnızca üreticisi değil aynı zamanda tüketicisiyiz. Bizim için özelleştirilmiş ürün ve hizmetlere sahip olmak istiyoruz. İhtiyaçlarımızın anlaşılmasını ve ilgilerimizin tahmin edilmesini istiyoruz.

Programlama Yerine Öğrenme

Bilgisayarda bir problemi çözmek için bir algoritmaya ihtiyacımız vardır.* *Algoritma*, girdiyi çıktıya dönüştürmek için yürütülen bir komut dizisidir. Örneğin sıralama [*sorting*] için bir algoritma geliştirilebilir. Girdi bir sayı kümesidir ve çıktı bu sayıların sıralı listesidir. Aynı görev için çeşitli algoritmalar olabilir ve bunlar içinden en az sayıda komut ve/veya bellek gerektiren en verimli algoritmayı bulmaya çalışabiliriz.

Gelgelelim, bazı problemler için bir algoritmamız yoktur. Müşteri davranışını tahmin etmek bunlardan biridir; bir diğeri ise istenmeyen e-postaları geçerli olanlardan ayırmaktır. Girdinin ne olduğunu biliyoruz: En basit durumda metin mesajı olan bir e-posta belgesi. Çıktının ne olması gerektiğini de biliyoruz: Mesajın istenmeyen olup olmadığını belirten bir evet/hayır çıktısı. Fakat böyle bir girdiyi böyle bir çıktıya nasıl dönüştüreceğimizi bilmiyoruz. İstenmeyen e-postanın ne olduğu, zaman içinde ve kişiden kişiye değişir.

Eksik bilgiyi veriyle telafi ederiz. Bir kısmı istenmeyen e-posta, bir kısmı geçerli e-posta olan büyük bir örnek veriyi kolayca derleye-

* Bir bilgisayar programı, görevi tanımlayan bir algoritmadan ve işlenen bilgilerin sayısal gösterimine yarayan veri yapılarından oluşur. Bilgisayar programlamaya ilişkin ufuk açıcı bir kitabın adı şudur: Niklaus Wirth, *Algorithms + Data Structures = Programs*, [Algoritmalar + Veri Yapıları = Programlar], Prentice Hall, 1976.

biliriz ve hedeflediğimiz şey, bu örneklem içinden bazı e-postaları istenmeyen kılan şeyi "öğrenmek"tir. Başka bir deyişle, bilgisayarın bu görevin algoritmasını otomatik olarak çıkarmasını isteriz. Sayıları sıralamayı öğrenmeye gerek yoktur (bunun için zaten algoritmalarımız var) ancak algoritması bulunmayan ama çok miktarda verisini kolayca toplayabileceğimiz birçok uygulama vardır.

Yapay Zekâ

Yapay öğrenme sadece bir veritabanı veya programlama problemi değildir, aynı zamanda yapay zekâ için bir gerekliliktir. Değişen bir çevrede bulunan bir sistem öğrenme yeteneğine sahip olmalıdır; aksi takdirde ona zeki denemez. Sistem bu değişiklikleri öğrenip onlara uyum sağlayabiliyorsa, sistem tasarımcısının olası tüm durumları öngörmesi ve her biri için ne yapılacağını tanımlaması gerekmez.

Bizim için sistem tasarımcısı evrimdi ve vücut şeklimizin yanı sıra yerleşik içgüdülerimiz ve reflekslerimiz milyonlarca yıl içinde evrildi. Hayatımız boyunca davranışlarımızı değiştirmeyi de öğreniyoruz. Bu, içinde yaşadığımız ortamda evrimin öngöremediği değişikliklerle başa çıkmamıza yardımcı olur. İyi tanımlanmış bir çevrede kısa bir yaşama sahip olan organizmaların tüm davranışları sabit olarak tanımlı olabilir; ancak evrim, yaşamımızda karşılaşabileceğimiz herhangi bir durum için gereken davranışı donanım içinde gömülü vermek yerine büyük bir beyin ve deneyim yoluyla kendimizi güncelleyebileceğimiz ve böylece kendimizi farklı ortamlara uyarlayabileceğimiz bir öğrenme mekanizması verdi. Bu, insanların yeryüzünün farklı bölgelerinde, oldukça farklı iklim ve koşullarda hayatta kalmasının ve başarılı olmasının nedenidir. Belirli bir durumdaki en iyi stratejiyi öğrendiğimizde bu bilgi beynimizde saklanır ve aynı durum yeniden ortaya çıktığında uygun stratejiyi hatırlar ve ona göre davranırız.

Hepimiz, aslında her hayvan, bir veri bilimcidir. Sensörlerimizle içinde yaşadığımız ortamı algılayıp veri toplarız ve ardından bu verileri işleyerek bu ortamdaki eylemlerimizi acıyı en aza indirmek

ve/veya hazzı en çok yapmak üzere kontrol eden soyut kurallara ulaşmayı isteriz. Bu kuralları beynimizde saklamak için bir belleğimiz vardır ve gerektiğinde bunları hatırlar ve kullanırız. Öğrenme yaşam boyu sürer. Kuralları, artık geçerli olmadıklarında unutur ya da ortam değiştiğinde güncelleriz.

Ancak öğrenmenin sınırları vardır. Beynimizin sınırlı kapasitesiyle asla öğrenemeyeceğimiz şeyler olabilir, tıpkı üçüncü bir kol ya da kafamızın arkasında bir göz çıkarmayı asla "öğrenemeyeceğimiz" gibi, ki bunlar, genetik yapımızda değişiklik gerektiren şeylerdir. Kabaca söylemek gerekirse genetik, binlerce nesil boyu çalışan donanımı tanımlarken öğrenme, bir kişinin ömrü boyunca bu donanım üzerinde çalışan (ve bununla kısıtlanan) yazılımı tanımlar.

Yapay zekâ beyinden ilham alır. Amaçları beynin işleyişini anlamak olan bilişsel bilimciler ve sinirbilimciler var; bu amaca yönelik olarak sinir ağı modelleri kuruyor ve benzetim çalışmaları yapıyorlar. Ancak yapay zekâ bilgisayar biliminin bir parçası olduğu için amacımız herhangi bir mühendislik alanında olduğu gibi kullanışlı sistemler oluşturmaktır. Bu yüzden, beyin bize ilham verse de nihayetinde geliştirdiğimiz algoritmaların biyolojik açıdan makul olup olmadıklarını pek umursamayız.

Beyne ilgi duyuyoruz, çünkü onun daha iyi bilgisayar sistemleri kurmamıza yardımcı olabileceğine inanıyoruz. Beyin, görme, konuşma, tanıma ve öğrenme gibi müthiş yetenekleri olan ve çoğu alanda şu an var olan mühendislik ürünlerini aşan bir bilgi işleme cihazıdır. Bu yetenekler makinelerde gerçeklendiğinde bariz bir ekonomik faydaya sahip olacaktır. Beynin bu işlevleri nasıl yerine getirdiğini anlayabilirsek, bunları algoritmalar olarak tanımlayabilir ve bilgisayarlar üstünde gerçekleyebiliriz.

Bilgisayara bir zamanlar "elektronik beyin" deniyordu ancak bilgisayar ve beyin birbirinden farklıdır. Bir bilgisayar genellikle bir veya birkaç işlemciye sahipken beyin, nöron denilen paralel olarak işleyen çok sayıda işlem biriminden oluşur. Ayrıntılar tam olarak bilinmemekle birlikte bu işlem birimlerinin bilgisayardaki tipik bir işlemciden çok daha basit ve yavaş olduğuna inanılmaktadır. Beyni

farklı kılan ve ona hesaplama gücünü sağladığına inanılan özelliklerden biri de büyük bağlanırlığıdır. Beyindeki nöronların on binlerce başka nöronla sinaps adı verilen bağlantıları vardır ve bunların hepsi paralel olarak çalışır. Bir bilgisayarda işlemci aktif, bellek ise ayrı ve pasiftir; ancak beyinde hem işlemenin hem de belleğin ağ üzerinde birlikte dağıldığı düşünülmektedir. İşleme nöronlar tarafından gerçeklenirken bellek nöronlar arasındaki sinapslardadır.

Beyni Anlamak

Marr'a (1982) göre, bir bilgi işlem sistemini anlamak üç *analiz düzeyinde* gerçekleşir:

- *Hesaplama teorisi* [*computational theory*], hesaplamanın amacına ve görevin soyut bir tanımına karşılık gelir.
- *Gösterim ve algoritma* [*representation and algorithm*], girdi ile çıktının nasıl gösterildiği ve girdiden çıktıya dönüştürme algoritmasının tanımlanmasıyla ilgilidir.
- *Donanım gerçeklenmesi* [*hardware implementation*], sistemin fiziksel olarak gerçeklenmesidir.

Bu analiz düzeylerindeki temel düşünce, aynı hesaplama teorisi için, çok sayıda gösterim ve bu gösterimdeki simgeleri işleyen çok sayıda algoritma bulunabilmesidir. Benzer şekilde, herhangi bir gösterim ve algoritma için birden fazla donanım yapılabilir. Herhangi bir teori için çeşitli algoritmalardan birini kullanabildiğimiz gibi aynı algoritmanın farklı donanım gerçeklemeleri de olabilir.

Bir örnek ele alalım: "6", "VI" ve "110", altı sayısının üç farklı gösterimidir; bunlar sırasıyla Arap rakamı, Roma rakamı ve ikili gösterimlerdir. Kullanılan gösterime bağlı olarak toplama işlemi için farklı bir algoritma vardır. Sayısal bilgisayarlar ikili gösterimi kullanırlar ve bu gösterimde toplama yapmak için belirli bir donanım olan elektronik devrelere sahiptirler. Bir abaküste rakamlar farklı şekilde gösterilir ve toplama farklı bir komut dizisine karşılık gelir, bu da başka bir donanım uygulamasıdır. İki sayıyı "kafamızda" topladığı-

mızda başka bir gösterim ve bu gösterime uygun bir algoritma kullanırız, ki bu da nöronlar tarafından uygulanır. Ancak tüm bu farklı donanım uygulamaları –yani biz, abaküs, sayısal bilgisayar– aynı hesaplama teorisini uygular: Bu da toplama işlemidir.

Klasik örnek, doğal ve yapay uçma makineleri arasındaki farktır. Bir serçe kanatlarını çırpar; bir uçak kanatlarını çırpmaz ancak jet motorları kullanır. Serçe ve uçak, farklı amaçlar için ortaya çıkmış, farklı sınırlamalara uyan iki farklı donanımdır. Fakat ikisi de aynı teoriye, yani aerodinamiğe dayanır.

Bu açıdan bakıldığında beynin öğrenmeyi gerçekleştiren özel bir donanım olduğunu söyleyebiliriz. Ters mühendislik yaparak yukarıya doğru çıkıp bu özel donanımdan kullandığı gösterim ile algoritmayı çıkarırsak ve bundan da soyutlama yapıp bir hesaplama teorisi elde edebilirsek ardından yeniden aşağıya doğru gidip bu teoriden önce bizim elimizdeki araç ve kısıtlara uygun başka bir gösterim ve algoritma, bunun ardından da elimizdeki malzemeye uygun farklı bir donanım geliştirebiliriz. Bunu yaparken sonuç donanımımızın daha ucuz, daha hızlı ya da daha başarılı olacağını umarız.

Tıpkı ilk uçan makine yapma girişimlerinin biz aerodinamik teorisini keşfedene kadar kuşlara benzemesi gibi, beynin yeteneklerine sahip yapılar gerçeklemeye yönelik ilk girişimlerin de çok sayıda işlem birimi ağına sahip olan beyin gibi görünmesi normaldir. Dördüncü bölümde, birbirine bağlı işlem birimlerinden oluşan yapay sinir ağlarını ve bu ağların nasıl öğrenebileceğini tartışacağız – bu da gösterim ve algoritma düzeyidir. Zamanla, zekânın hesaplama teorisini keşfettiğimizde, tıpkı tüylerin uçuş için uygulama ayrıntıları olduğunu fark ettiğimiz gibi nöron ve sinapsların da zekâ için ille de gerekmeyen uygulama ayrıntıları olduğunu keşfedebiliriz.

Örüntü Tanıma

Uzun yıllar boyunca bilgisayar biliminde birçok işi, elle belirlenen kurallar ve algoritmalarla programlanmış "uzman sistemler"le çözmeye çalıştık ama onlarca yıllık çalışma son derece sınırlı bir ba-

şarı sağladı. Bu görevlerden bazıları yapay zekâyla ilişkilendirilir çünkü zekâ gerektirdiklerine inanılır. Son zamanlarda büyük bir ilerleme kaydettiğini gördüğümüz şu anki yaklaşım, veriler üzerinde yapay öğrenme kullanmayı temel almaktadır.

Yüz tanıma örneğini ele alalım: Zahmetsizce gerçekleştirdiğimiz bir görevdir bu; her gün aile üyelerimizi ve arkadaşlarımızı poz, ışık, saç modeli vb. farklılıklarına rağmen yüzlerine veya fotoğraflarına bakarak tanıyoruz. Yüz algılama bizim için önemlidir çünkü dostu düşmandan ayırt etmeye yarar. Hayatta kalışımız için önemi yalnızca kimliklendirme bakımından değildir; yüz, ruh halimizin de gösterge panosudur. Mutluluk, öfke, şaşkınlık ve utanç gibi hisler yüzümüzden okunabilir ve bizler hem bu durumları sergilemek hem de başkalarında algılamak üzere evrildik.

Böylesi bir tanıma bizim için kolay olsa da bunu bilinçsizce yapıyoruz ve nasıl yaptığımızı açıklayamıyoruz. Uzmanlığımızı açıklayamadığımız için doğrudan buna karşılık gelen bir bilgisayar programı yazamıyoruz.

Öğrenen bir program, bir kişinin farklı yüz görüntülerini analiz ederek o kişiye özgü örüntüyü yakalar ve daha sonra verilen bir görüntüde o örüntüyü kontrol eder. Bu, *örüntü tanıma* örneklerinden biridir.

Bunu yapabilmemizin nedeni, bütün doğal görüntüler gibi bir yüz görüntüsünün de rasgele bir piksel derlemesi olmadığını bilmemizdir (rasgele bir görüntü, karlı bir televizyon gibidir). Bir yüzün yapısı vardır. Simetriktir. Gözler, burun ve ağız yüzün belli yerlerinde bulunur. Her insanın yüzü, bunların özel bir birleşiminden oluşan bir örüntüdür. Aydınlatma veya pozumuz değiştiğinde, saçlarımızı uzattığımızda veya gözlük taktığımızda ya da yaş aldığımızda, yüz görüntüsünün bazı kısımları değişir ancak bazı kısımlar değişmez. Bu, müşteri davranışına benzer çünkü düzenli olarak aldığımız ürünler ve ayrıca anlık satın alımlar vardır. Öğrenen algoritma belli bir insanın yüzünü tanımlamak üzere o kişiye ait bir dizi görüntünün üzerinden geçerek değişmeyen *ayırt edici* özellikleri ve bunların bir araya gelme biçimini bulur.

Öğrenme Hakkında Konuştuğumuzda Konuştuğumuz Şey

Yapay öğrenmede amaç, eldeki verilere uyan bir program oluşturmaktır. Bir öğrenen program sıradan bir bilgisayar programından farklı olarak *değiştirilebilir parametrelere sahip bir genel şablondur* ve aynı program bu parametrelere atanan değerlere göre farklı şeyler yapabilir. Öğrenen algoritma *model şablonun parametrelerini veri üstünde tanımlanmış bir başarım iyileyecek biçimde ayarlar.*

Örneğin, bir yüz tanıyıcı için parametreler, bir kişinin eğitim görüntü kümesinde en yüksek tahmin doğruluğunu elde edecek biçimde ayarlanır. Bu öğrenme genellikle tekrarlı ve artımlıdır. Öğrenen program birbiri ardına çok sayıda örnek görüntü görür ve parametreler her örnekte bir miktar güncellenir, böylece başarım zaman içinde yavaş yavaş artar. Ne de olsa öğrenme budur: Tenis, geometri veya yabancı dil olsun, bir işi öğrendikçe onu daha iyi yapar hale geliriz.

İkinci bölümde, şablonun ne olduğuna (aslında göreceğimiz gibi, görevin türüne bağlı olarak farklı şablonlarımız var) ve parametreleri öğrenme verisi üstünde en yüksek başarımı sağlayacak şekilde ayarlayan farklı öğrenme algoritmalarına daha ayrıntılı değineceğiz.

Bir öğrenme uygulaması tasarlarken bir dizi önemli konuya dikkat etmek gerekir.

İlk olarak şunu aklımızdan çıkarmamalıyız ki yalnızca çok fazla veriye sahip olmamız altta yatan, öğrenilebilecek kuralların var olduğu anlamına gelmez. Altta yatan süreçte bağımlılıklar olduğundan ve toplanan verilerin bunların kabul edilebilir doğrulukla öğrenilmeleri için yeterli bilgi sağladığından emin olmalıyız. İnsanların isimlerini ve telefon numaralarını içeren bir telefon rehberimiz olsun. İsimler ve telefon numaraları arasında, verili bir telefon rehberi (ne kadar büyük olursa olsun) üzerinde eğitim yaptıktan sonra yeni bir isim gördüğümüzde ona denk gelen telefon numarasını tahmin etmemizi sağlayacak genel bir ilişki olduğuna inanmak mantıklı değildir.

İkincisi, öğrenen algoritma verimli olmalıdır çünkü genelde

çok miktarda verimiz vardır ve hesaplama ile belleği etkin kullanarak öğrenmeyi olabildiğince hızlı yapmak isteriz. Birçok uygulamada, uygulamanın altta yatan özellikleri zamanla değişebilir; böyle bir durumda, önceden toplanan veriler eskir ve eğitilen modelin yeni veriyle devamlı ve etkili biçimde güncellenmesi ihtiyacı doğar.

Üçüncüsü, öğrenen bir model eğitilip tahmin için kullanılmaya başlandığında, o da bellek ve hesaplama açısından verimli olmalıdır. Belli bazı uygulamalarda bu son modelin verimliliği, tahmin doğruluğu kadar önemli olabilir.

Tarihçe

Neredeyse tüm bilimler veriye model uydurur. Galilei, Newton ve Mendel gibi biliminsanları deneyler tasarlayıp gözlemler yaptılar ve veri topladılar. Daha sonra geliştirdikleri teorilerle bunlardan bilgi çıkarmaya çalıştılar, yani gözlemledikleri verileri açıklayan modeller tasarladılar.* Daha sonra bu teorileri tahminlerde bulunmak için kullandılar ve işe yaramadıklarında daha fazla veriyle güncellediler. Bu veri toplama ve teori/model oluşturma süreci, yeterli açıklama gücüne sahip modeller elde edilinceye kadar devam etti.

Artık bu tür bir analizin elle yapılamayacağı bir noktadayız çünkü böyle bir analiz yapabilecek çok sayıda insan yoktur; ayrıca veri miktarı da çok büyüktür ve elle analiz mümkün değildir. Bu nedenle verileri analiz edip otomatik olarak bilgi çıkarabilen, başka bir deyişle öğrenen bilgisayar programlarına gereksinim her geçen gün artıyor.

* İlk biliminsanları, fiziksel dünyayı açıklayan bu tür kuralların varlığının, yalnızca bir tanrı sayesinde ortaya çıkmış olabilecek düzenli bir evrenin işareti olduğuna inanıyorlardı. Doğayı gözlemlemek ve doğal olayları açıklayan kurallar bulmaya çalışmanın antik Mezopotamya'ya dek uzanan bir tarihçesi vardır. Önceleri sahte bilimler bilimden ayrılamıyordu. Geriye dönüp baktığımızda, eski insanların astrolojiye inanması şaşırtıcı gelmiyor: Eğer Güneş'le Ay'ın hareketleri hakkında, sözgelimi tutulmaları tahmin edebilmemizi sağlayan bir düzen ve kurallar varsa, bunlara kıyasla son derece önemsiz görünen insanların hareketlerinde de düzenliliklerin ve kuralların var olduğunu iddia etmek o kadar zorlama görünmüyor.

Ele aldığımız yöntemlerin kökleri farklı bilimsel alanlara dayanmaktadır. Bazen aynı veya çok benzer algoritmaların farklı tarihsel yolları izleyen farklı alanlarda birbirinden bağımsız olarak icat edilmesi görülmemiş şey değildir.

Yapay öğrenmenin altında yatan ana teori istatistiğe dayanır; istatistikte, özel gözlemlerden genel açıklamalara geçmeye çıkarım [*inference*], öğrenmeye ise kestirim [*estimation*] denir. Sınıflandırma, istatistikte ayırtaç [*discriminant*] analizi olarak adlandırılır. Eskiden istatistikçiler küçük örneklemler üzerinde çalışmaya alışkındılar ve matematikçi oldukları için çoğunlukla matematiksel bakımdan analiz edilebilen basit modeller üzerinde çalışıyorlardı. Mühendislikteyse sınıflandırma, örüntü tanıma olarak adlandırılır ve yaklaşım daha gözlemseldir.

Bilgisayar biliminde, yapay zekâ çalışmalarının bir parçası olarak öğrenen algoritmalar üzerine araştırmalar yapıldı. Paralel ancak neredeyse bağımsız bir çalışma hattına ise *veri bilgi keşfi* deniyordu. Elektrik mühendisliğinde sinyal işleme araştırmaları, uyarlamalı görüntü işleme ve konuşma tanıma programlarına varan yolu açtı.

1980'lerin ortasında çeşitli disiplinlerden yapay sinir ağı modellerine yönelik büyük bir ilgi patlaması gerçekleşti. Bilgisayar bilimleri, elektrik mühendisliği ve uyarlanabilir kontrolün [*adaptive control*] yanı sıra fizik, istatistik, psikoloji, bilişsel bilim, sinirbilim ve dilbilim de bu araştırmalarda yer alıyordu. Yapay sinir ağları üzerine yapılan araştırmaların belki de en önemli sonucu, özellikle istatistik ve bilgisayar bilimleri olmak üzere çeşitli disiplinleri birleştiren bu sinerjidir. Daha sonra yapay öğrenme alanına dönüşecek olan yapay sinir ağı araştırmalarının 1980'lerde başlaması rastlantı değildir. O zamanlar VLSI (çok büyük ölçekli entegrasyon [*very-large-scale integration*]) teknolojisindeki ilerlemelerle binlerce işlemci içeren paralel donanım oluşturma kapasitesini kazandık ve yapay sinir ağları, hesaplamayı hepsi de paralel olarak çalışan çok sayıda işleme birimine dağıtacak olası bir teori olarak ilgi çekiciydi. Dahası, bu birimler öğrenebildikleri için programlanmaya da gerek duymayacaklardı.

Bu farklı alanlarda yürütülen araştırmalar geçmişte farklı vurgular ve yollar izlemiştir. Bu kitaptaki amacımız, birleşik ve tanıtıcı bir değerlendirme yapmak için bunları, bazı ilginç uygulamalardan da bahsederek bir araya getirmektir.

2

YAPAY ÖĞRENME, İSTATİSTİK VE VERİ ANALİZİ

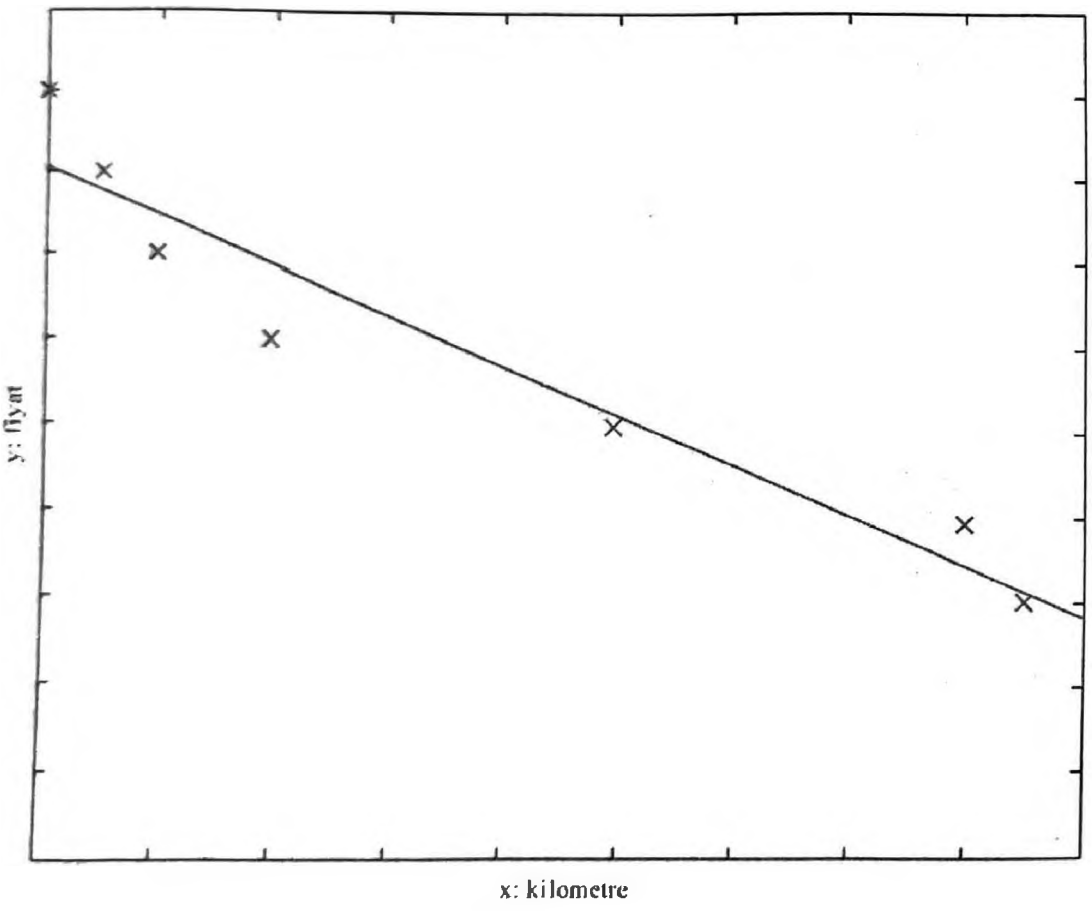
Kullanılmış Bir Otomobilin Fiyatını Tahmin Etmeyi Öğrenmek

Önceki bölümde gördüğümüz gibi yapay öğrenmeyi, ilgilendiğimiz gözlemler arasında bir ilişki olduğuna inandığımız ancak bunun şeklini tam olarak bilmediğimiz durumlarda kullanıyoruz. İlişkiyi tam olarak bilmediğimiz için doğrudan bilgisayar programı yazamıyoruz. Dolayısıyla yaklaşımımız, örnek gözlem verileri toplamak ve ilişkiyi keşfetmek üzere bu verileri analiz etmek olacaktır. Şimdi, bir ilişkiden ne kastettiğimizi ve onu verilerden nasıl çıkardığımızı daha ayrıntılı olarak ele alalım. Her zaman olduğu gibi, tartışmayı somutlaştırmak için örnek kullanmak iyi bir fikir olacaktır.

Kullanılmış bir otomobilin fiyatını tahmin etmek istediğimizi düşünelim. Bu, yapay öğrenme için iyi bir örnek uygulamadır, çünkü bu da kesin formülünü bilmediğimiz bir ilişkidir. Bununla birlikte, bazı kurallar olması gerektiğini de biliyoruz: Fiyat, otomobilin markası gibi birtakım özelliklere, kat edilmiş kilometre gibi kullanıma bağlı unsurlara, hatta ekonominin güncel durumu gibi doğrudan otomobille ilgili olmayan öğelere bağlıdır.

Bunları etkenler olarak tanımlayabilsek bile fiyatı nasıl etkilediklerini bilmiyoruz. Sözgelimi, kat edilmiş kilometre arttıkça fiyatın azaldığını biliyoruz, ancak bunun ne kadar hızlı gerçekleştiğini bilmiyoruz. Bilmediğimiz ve öğrenmek istediğimiz şey, fiyatı belirlemek üzere bu etkenlerin nasıl birleştiğidir. Bu amaca yönelik olarak, şu an ikinci el piyasasında olan bir dizi örnek otomobile bakıp bu örnek arabaların özelliklerini ve fiyatlarını kaydederek bir veri kümesi toplarız ve ardından, bir otomobilin özellikleriyle fiyatı arasındaki ilişkiyi bu veriden çıkarmaya çalışırız (bakınız Şekil 2.1).

Bunu yaparken ilk önemli konu, girdi gösterimi olarak neyin kullanılacağına, yani kullanılmış bir otomobilin fiyatını etkilediğine



Şekil 2.1 Kullanılmış bir otomobilin fiyatının tahmin edilmesi bir regresyon problemidir. Her bir çarpı, verimizdeki bir otomobildir ve x ekseni otomobilin kaç kilometre kullanılmış olduğunu, y ekseni de şu anki ederini gösterir. Bunlar öğrenme kümemizi oluşturur. Kullanılmış bir otomobilin fiyatını tahmin etmek için bu verilere uyan (olabildiğince yakınlarından geçen) bir model öğrenmek isteriz. Burada kalın çizgi doğrusal bir modeli gösteriyor. Bu modelle, kilometresi bilinen başka bir otomobilin fiyatı tahmin edilebilir.

inandığımız özelliklere karar vermektir. İlk akla gelenler otomobilin markası, modeli, üretim yılı ve kilometresidir. Başka özellikler de düşünebilirsiniz, ancak bunlar kolayca gözlemlenebilir olmalıdır.

Her zaman akılda tutulması gereken önemli bir gerçek, iki farklı otomobilin bu özellikler için tam olarak aynı değerlere sahip olmasına rağmen fiyatlarının farklı olabilmesidir. Bunun nedeni, sonradan eklenmiş aksesuvarlar gibi fiyatı etkileyen başka etkenlerin bulunmasıdır. Doğrudan gözlemleyemediğimiz ve dolayısıyla girdilere dahil edemediğimiz etkenler de olabilir – örneğin aracın geçmişte kullanıldığı koşulları ve ne kadar iyi bakılmış olduğunu bilemeyiz.

Kritik nokta şudur: Girdi olarak ne kadar özellik listelesek de çıktıyı etkileyen başka etkenler olabilir. Olası etkenlerin hepsini kaydedip girdi olarak kullanamayız ve ihmal ettiğimiz tüm diğer etkenler bir belirsizlik doğurur.

Bu belirsizliğin sonucu olarak artık kesin bir fiyat tahmin edemeyiz, ancak bilinmeyen fiyatın büyük olasılıkla yer alacağı bir *aralığı* tahmin edebiliriz ve bu aralığın genişliği belirsizliğin miktarına bağlıdır. Bu aralık, girdi olarak almadığımız ya da alamadığımız etkenlere bağlı olarak fiyatın ne kadar değişebileceğini gösterir.

Rastsallık ve Olasılık

Matematik ve mühendislikte belirsizliği, *olasılık teorisini* kullanarak modelleriz. Belirlenimci [*deterministic*] bir sistemde aynı girdiler için çıktı hep aynıdır, ama rastsal bir süreçte çıktı, rastsallığı oluşturan kontrol edilemez etkenlere de bağlıdır.

Yazı tura attığımızı düşünelim. Madeni paranın fiziksel özellikleri, başlangıç konumu, para atılırken ona uygulanan kuvvetin yönü, nerede ve nasıl yakalandığı gibi bilgilere erişebilseydik, atışın sonucunu tam olarak tahmin edebileceğimiz söylenebilir ancak tüm bu bilgiler bizden saklandığı için, yalnızca olası sonuçların olasılığı hakkında konuşabiliriz. Sonucun yazı mı yoksa tura mı olduğunu bilmeyiz, ancak her bir sonucun olasılığı hakkında bir şeyler söyleyebiliriz; bu da o sonucun olasılığına ne kadar inandığımıza dair bir ölçüdür.

Örneğin, para *bileviz* ise yazı görme olasılığı ile tura görme olasılığı aynıdır – aynı parayı birçok kez atarsak, kabaca tura sayısı kadar yazı görmeyi bekleriz.

Bu olasılıkları bilmiyorsak ve kestirmek istiyorsak, *istatistiğin* alanına girmişiz demektir. Ortak terminolojiyi izliyor ve her bir veri noktasına bir "örnek" [*example*] diyoruz, "örneklem" [*sample*] sözcüğünü ise böyle örneklerin *bütünü* için kullanıyoruz. Amaç, örneklemi kullanarak kestirmek istediğimiz değerleri hesaplamak için bir *model* oluşturmaktır. Yazı tura atma durumunda, parayı çok kez atarak bir örneklem toplarız ve sonuçları –yazı ya da tura– gözlemlerimiz olarak kaydederiz. Daha sonra, tura olasılığı için kestirimimiz, örneklemdeki turaların oranı olabilir. Eğer parayı altı kez atarsak ve örneklemimizde dört tura ile iki yazı görürsek, tura olasılığını ⅔ (ve böylece yazı olasılığını ⅓) olarak kestirebiliriz. Bu yüzden bir sonraki deneyin sonucunu tahmin etmemiz istenirse, tahminimiz tura olacaktır, çünkü olasılığı daha yüksektir.

Şans oyunlarının altında bu tür bir belirsizlik yatar, bu da kumarı bazı insanlar için heyecan verici kılar. Fakat çoğumuz belirsizliği sevmeyiz ve günlük yaşamlarımızda, gerektiğinde bir bedel ödeyerek bundan kaçınmaya çalışırız. Örneğin ağır bir riskle karşı karşıya olduğumuzda sigorta yaptırırız. Küçük bir prim ödeme pahasına, büyük miktarda bir parayı (yüksek değerde bir malı) asla kaybetmemenin kesinliğini tercih ederiz.

Kullanılmış bir otomobilin fiyatı, bir otomobilin değer kaybını rastsal bir süreç haline getiren kontrol edilemeyen etkenlerin var olması bakımından benzerdir. Üretim hattında birbirini izleyen iki otomobil, o noktada tamamen aynıdır ve dolayısıyla tam olarak aynı değerdedir. Ama bir kere satın alınıp kullanılmaya başlandıktan sonra her türden etken değerlerini etkilemeye başlar: sahiplerinden biri daha titiz olabilir, otomobillerden biri daha iyi hava koşullarında sürülebilir, birinden biri kazaya karışmış olabilir, vb. Bu etkenlerin her biri fiyatı değiştiren rasgele bir yazı tura atışı gibidir.

Benzer bir akıl yürütme, müşteri davranışı için de ileri sürülebilir. Genel olarak müşterilerin, hane halkının bileşimi, zevkleri, ge-

lirleri ve bunlar gibi etkenlere bağlı olarak alımlarında belirli örüntüleri takip etmelerini bekleriz. Yine de çeşitliliği ortaya çıkaran ek rastsal etkenler de bulunur: tatil, hava durumu değişikliği, akılda kalıcı reklamlar vb. Bu rastsallığın bir sonucu olarak daha sonra hangi ürünlerin satın alınacağını tam olarak tahmin edemeyiz, ancak bir ürünün satın alınma olasılığını hesaplayabiliriz. Sonrasında tahminde bulunmak istediğimizde olasılığı en yüksek ürünleri seçebiliriz.

Genel Bir Model Öğrenmek

Her veri topladığımızda bunları genel eğilimleri öğrenecek şekilde toplamamız gerekir. Örneğin bir otomobili temsil ederken markayı bir girdi niteliği olarak kullanırsak, bu çok özel bir gösterim olur. Ama bunun yerine koltuk sayısı, motor gücü, bagaj hacmi gibi özellikleri kullanırsak, daha genel bir tahmin modeli öğrenebiliriz. Bunun nedeni, farklı markadaki otomobillerin müşteri segmenti olarak adlandırılan aynı müşteri kitlesine hitap etmesi ve aynı segmentteki otomobillerin benzer bir biçimde değer kaybetmesini beklememizdir. Markayı göz ardı edip türü tanımlayan temel niteliklere odaklanmak, gürültülü olmakla birlikte aynı veri örneğini aynı türdeki tüm otomobiller için kullanmakla eşdeğerdir. Bu da verimizin büyüklüğünü önemli oranda artırır.

Çıktı için de benzer bir iddia öne sürülebilir. Fiyatı olduğu gibi, örneğin doğrudan Türk lirası olarak tahmin etmek yerine otomobilin asıl değerinin yüzdesini, yani değer kaybının etkisini tahmin etmek daha mantıklı olur. Bu da yine daha genel bir model öğrenmemizi sağlar.

Genel modelleri öğrenmek iyi olsa da çok genel modelleri öğrenmeye de çalışmamalıyız. Örneğin otomobiller ve kamyonlar son derece farklı özelliklere sahip olduklarından ikisi için birlikte veri toplayıp tek bir model öğrenmeye çalışmak yerine her biri için ayrı veri toplayıp farklı modeller öğrenmek daha uygun olur.

Bir diğer önemli gerçek de verinin altında yatan sürecin zaman içinde değişebilmesidir. Örneğin bir otomobilin fiyatı yalnızca oto-

mobilin niteliklerine, geçmiş kullanımını temsil eden niteliklere veya sahibinin niteliklerine değil; aynı zamanda ekonominin durumuna, yani başka ürünlerin fiyatına da bağlıdır. Tüm alım satımlarımızı yaptığımız ortam olan ekonomi önemli değişikliklere maruz kalırsa, önceki eğilimler geçerliliğini sürdüremez. İstatistiksel olarak konuşursak, artık verinin altında yatan rastsal sürecin özellikleri değişmiştir — attığımız paralar değişmiştir. Bu durumda, önceden öğrenilmiş model artık geçerli değildir ve yeni veri toplayarak yeniden öğrenmemiz gerekir. Ya da bir biçimde tahmin başarısı hakkında geribildirim alabilirsek modeli kullanmaya devam ettikçe başarının düşük olduğu durumlar için modele ayarlamalar yaparız.

Model Seçimi

Öğrenmede en kritik noktalardan biri girdilerle çıktılar arasındaki ilişki şablonunu tanımlayan *modeldir*. Örneğin çıktıyı niteliklerin ağırlıklı bir toplamı olarak yazabileceğimize inanıyorsak *doğrusal bir model* kullanabiliriz. Sözgelimi, her ek koltuk otomobilin değerini X lira artırır ve fazladan kat edilen her bin kilometre değeri Y lira düşürür, vb.

Her bir niteliğin ağırlığı (yukarıdaki X ile Y *değerleri*) örneklemden hesaplanabilir. Bir ağırlık pozitif veya negatif olabilir, yani bunlara karşılık gelen niteliklerden daha fazlası, sırasıyla fiyatı artırır veya azaltır. Bir ağırlığın sıfıra çok yakın olduğu kestirilirse, ilgili niteliğin önemli olmadığı sonucuna varıp modelden çıkarabiliriz. Bu ağırlıklar modelin *parametreleridir* ve değerleri veri kullanılarak hesaplanır. Model hep sabittir. Ayarlanabilir olan parametrelerdir ve öğrenme dediğimiz şey veriyle model daha iyi eşleşsin diye parametrelerde yapılan bu ayarlama işlemidir.

Doğrusal model son derece basit olduğundan çok yaygın kullanılır. Az sayıda parametresi vardır ve ağırlıklı toplama hesabı basittir. Böyle bir modeli anlamak ve yorumlamak da kolaydır. Dahası, çok sayıda farklı uygulamada şaşırtıcı derecede başarıyla çalışır.

Her model sabit bir şablon belirlediği için, parametrelerini nasıl

değiştirirsek değiştirelim, her model yalnızca belli bir problem kümesini öğrenmek için kullanılabilir. *Model seçimi*, modeller arasında seçim yapmak demektir. Doğru modeli seçmek, verilmiş modelin parametrelerini ayarlamaktan daha zor bir iştir ve uygulama hakkındaki bilgi bize bu seçimde yardımcı olur.

Örneğin otomobil fiyatlarının tahmininde, doğrusal model her zaman uygulanamayabilir. Gözlemsel olarak otomobilin yaşının etkisinin aritmetik değil geometrik olduğu görülebilir: Otomobilin yaşında örneğin her ek yıl fiyatı aynı oranda düşürmez; tipik bir otomobilin her yıl yüzde 15 değer kaybettiği söylenir.* Gelecek bölümlerde, çeşitli uygulamalarda kullanılabilmeleri anlamında daha güçlü, doğrusal olmayan modelleri kullanan yapay öğrenme algoritmalarını ele alacağız.

Gözetimli Öğrenme

İstatistikte bir küme girdiden bir çıktı değerini tahmin etme görevine *regresyon* denir. Doğrusal model için bu doğrusal regresyon olarak adlandırılır. Yapay öğrenmede regresyon bir gözetimli [*supervised*] öğrenme türüdür. Her otomobil için bize istenen çıktıyı (yani fiyatı) sağlayabilen bir gözetmen vardır. Örneğin, piyasada satılan ikinci el otomobillere bakarak verileri topladığımızda otomobillerin niteliklerini, yanında satış fiyatlarını da gözlemleyebiliriz.

Yukarıda bahsettiğimiz gibi doğrusal model, olası tek model değildir. Her model girdiler ile çıktılar arasındaki belirli bir bağımlılık varsayımına karşılık gelir. Öğrenme, modelin veri örnekleri üzerinde en doğru tahminleri yapması için parametrelerin ayarlanmasına karşılık gelir. Genel durumda öğrenme, bir başarım kriterine göre daha iyi hale gelmek demektir ve regresyonda başarım, modelin çıktı tahminlerinin öğrenme verisindeki gözlenmiş çıktı değerlerine ne kadar yakın olduğuna bağlıdır. Buradaki varsayım, öğrenme verilerinin altta yatan işin özelliklerini yeterince iyi yansıttığıdır; bu ne-

* https://en.wikipedia.org/wiki/Depreciation.

denle öğrenme verisi üzerinde yeterince doğru çalışan bir modelin o işi öğrendiği söylenebilir.

Literatürdeki farklı yapay öğrenme algoritmaları, ya kullandıkları model, ya kullandıkları başarım kriteri ya da başarımı artırmak için parametrelerin nasıl ayarlandığına göre farklılaşır.

Bu noktada, yapay öğrenmenin amacının öğrenme verisini taklit etmek değil, öğrenme sırasında görülmemiş yeni durumlar için doğru tahminler yapmak olduğunu hatırlamalıyız. Piyasada yalnızca belirli sayıda otomobil olsaydı ve hepsinin fiyatını bilseydik, o zaman tüm bu değerleri bir tabloda saklayıp gerektiğinde o tabloya bakabilirdik. Fakat bu, *ezberlemek* olurdu. Ancak, sıklıkla olası tüm örneklerin sadece küçük bir altkümesini görürüz (öğrenmeyi ilginç yapan da budur) ve bu veriden *genelleme yapmak* isteriz. Yani öğrenme sırasında görülmemiş girdiler için de iyi tahminler yapabilmek üzere, öğrenme örneklerinin ötesine geçmiş genel bir model öğrenmek isteriz.

Bizim örneğimizde, olası tüm otomobillerin yalnızca küçük bir altkümesini görmesine rağmen bu öğrenme verisinin dışında yer alan ve doğru çıktısı öğrenme kümesinde verilmemiş olan bir otomobilin fiyatını da yeterince doğru tahmin edebilmek isteriz. Bir modelin böylesi yeni örnekler için doğru çıktıyı ne kadar iyi tahmin ettiği, modelin ve öğrenen algoritmanın *genelleştirme yeteneği* olarak adlandırılır.

Burada yaptığımız temel varsayım (öğrenmeyi mümkün kılan da bu varsayımdır), kullanmayı seçtiğimiz girdi özellikleri bakımından benzer otomobillerin benzer fiyatlara sahip olduğudur. Bu girdi özelliklerinin değerleri yavaşça değiştikçe –örneğin kilometre değiştikçe– fiyatın da yavaşça değişmesi beklenir. Çıktıda, girdi bakımından düzgünlük [*smoothness*] vardır ve genelleştirmeyi mümkün kılan da budur. Böylesi bir düzenlilik olmadan özel durumlardan genel bir modele ilerleyemeyiz, çünkü bu durumda öğrenme kümesinin içindeki ve dışındaki tüm durumları kapsayan genel bir model olduğu inancı temelsiz kalır.

Sadece kullanılmış bir otomobilin fiyatını tahmin ederken de-

ğil, aynı zamanda gerçek dünyadan veri kullanan birçok işte, örneğin ticari uygulamalarda, örüntü tanımada veya bilimde bu düzgünlüğü görürüz. Dünyada düzenlilikler olduğu için yapay öğrenme ve tahmin mümkündür. Dünyadaki şeyler düzgün bir şekilde değişir. A noktasından B noktasına "ışınlanmayız", bir dizi ara konumdan geçmemiz gerekir. Nesneler dünyada sürekli bir uzam parçasını kaplar. Görme alanımızdaki yakın noktalar aynı nesneye aittir ve bu nedenle çoğunlukla aynı rengin tonlarına sahiptir. Bir şarkıda veya konuşmada ses düzgün bir şekilde değişir. Süreksizlikler sınırlara karşılık gelir ve nadirdir. Duyu sistemlerimizin çoğu bu düzgünlüğü kullanır. Kanizsa üçgeni gibi görsel yanılsamalar, duyu organlarımızın ve beynimizin dış dünyayla ilgili olarak yaptığı böyle düzgünlük varsayımlarından kaynaklanmaktadır.

Böyle bir varsayım gereklidir çünkü öğrenme veya verilere model yerleştirme, *kötü konumlanmış bir problemdir* – toplanan veri tek bir model bulmak için yeterli değildir. Her öğrenme algoritması tek bir model bulabilmek için veri hakkında ek birtakım varsayımlarda bulunmak zorundadır ve bu varsayımlara, öğrenme algoritmasının yanlılığı [*inductive bias*] adı verilir (Mitchell 1997).

Bu genelleme yeteneği, yapay öğrenmenin temel gücüdür ve öğrenme örneklerinin ötesine geçmeyi sağlar. Elbette bir yapay öğrenme modelinin doğru biçimde genelleme yapmasının garantisi yoktur –bu, modelin verinin altında yatan işe ne kadar uygun olduğuna, ne kadar öğrenme verisi bulunduğuna ve model parametrelerinin ne kadar iyi ayarlanabildiğine bağlıdır– ancak iyi genelleme yaparsa, veriden çok daha fazlasına karşılık gelen bir modelimiz olur. Yalnızca öğretmeninin daha önce sınıfta çözdüğü alıştırmaları çözebilen bir öğrencinin konuyu öğrendiğini söyleyemeyiz; öğrencilerin bu örneklerden hareketle aynı konudaki yeni soruları çözebilmesini sağlayacak genel bir anlayışa erişmiş olmasını isteriz.

Bir Diziyi Öğrenmek

Çok basit bir örnek görelim. Size bir dizi sayı veriliyor ve dizide bir sonraki sayıyı bulmanız isteniyor olsun. Diyelim ki dizi şu:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Herhalde bunun *Fibonacci dizisi* olduğunu fark etmişsinizdir. İlk iki terim 0 ve 1'dir ve izleyen her terim, önceki iki terimin toplamıdır. Modeli bir kez belirledikten sonra onu tahmin yapmak için kullanabilirsiniz – bir sonraki sayı 89 olacaktır. Ardından aynı modeli kullanarak tahmin etmeye devam edebilir ve istediğiniz uzunlukta bir dizi oluşturabilirsiniz.

Bu modele varmamızın nedeni, bilinçsizce bu veriler hakkında *basit bir açıklama* bulmaya çalışıyor olmamızdır. Bu günlük hayatta her zaman yaptığımız bir şeydir. Felsefede *Occam'ın usturası* bize gereksiz karmaşıklığı kesip atarak daha basit açıklamaları tercih etmemizi söyler. Bu dizi örneğinde önceki iki terimin toplandığı doğrusal bir kural yeterince basittir.

Dizi daha kısa olsaydı,

0, 1, 1, 2.

Hemen Fibonacci dizisini tercih etmezdiniz – benim tahminim 2 olurdu. Kısa diziler söz konusu olduğunda birçok olası ve daha basit kural vardır. Ardışık her bir terimi gördüğümüzde, bir sonraki değeri öncekilere uymayan kurallar elenir. Model uydurma [*model fitting*], temelde bir eleme sürecidir: Her fazladan gözlem (öğrenme örneği), kendisine uymayan tüm aday modelleri eleyen bir kısıttır. Ve basit olanları bitirdikten sonra tüm terimleri kapsamak üzere adım adım daha karmaşık açıklamalara geçmemiz gerekir.

Modelin karmaşıklığı üst parametreler [*hyperparameters*] kullanılarak tanımlanır. Burada modelin doğrusal olması ve yalnızca önceki iki terimin kullanılması üst parametrelerdir.

Şimdi diyelim ki dizi şöyle:

0, 1, 1, 2, 3, 6, 8, 13, 20, 34, 55

Belki bu diziyi de açıklayan bir kural bulabilirsiniz ancak bunun karmaşık bir kural olacağını düşünüyorum. Bunun yerine, bu dizinin *iki hatalı* (5 yerine 6, 21 yerine 20) bir Fibonacci dizisi olduğunu söyleyip 89'u yine de sıradaki sayı olarak tahmin edebiliriz – veya sıradaki sayının [88, 90] aralığında bulunacağını tahmin edebiliriz.

Bu sayı dizisini hatasız açıklayan karmaşık bir model yerine gürültülü bir Fibonacci, eğer hata olabileceğine inanıyorsak, daha olası bir açıklama olabilir (bilinmeyen etkenlere bağlı rastsal etkiler konusunda yaptığımız önceki tartışmamızı hatırlayın). Ve gerçekten de hatalar olasıdır. Sensörlerimizin çoğu mükemmel olmaktan uzaktır, klavye kullananlar her zaman yazım hatası yapar ve genellikle makul ve rasyonel davrandığımıza inanmamıza rağmen fevri olarak da hareket eder ve gayriihtiyari satın alır/okur/dinler/tıklar/seyahat ederiz. İnsan davranışı bazen Apollon'cu olduğu kadar Dionysos'çudur da.

Öğrenme, *sıkıştırma* da yapar. Dizinin altında yatan kuralı bir kez öğrendiğimizde artık veriye ihtiyaç duymayız. Veriye bir kuralı uydurarak ondan daha basit olan bir açıklama elde ettiğimizde saklamak için daha az bellek ve işlemek için daha az hesap gerekir. Bir kez çarpma kurallarını öğrendikten sonra olası her sayı çifti için çarpım tablosunu hatırlamamız gerekmez.

Kredi Puanlama

Şimdi diğer bir yapay öğrenme algoritması türünü tartışmamıza yardımcı olacak başka bir uygulama görelim. *Kredi*, banka gibi bir finans kurumu tarafından borç verilen, genellikle taksitler halinde faiziyle geri ödenmesi gereken bir para miktarıdır. Bankanın bir borçla ilişkili riski önceden tahmin etmesi önemlidir, bu da müşterinin temerrüde düşüp tüm tutarı geri ödeyememe olasılığının ne kadar olduğu hakkında bir ölçüdür. Böylece hem bankanın kâr edeceğinden hem de müşteriyi finansal kapasitesinin üzerinde bir borçla zora sokmayacağından emin olmak isteriz.

Kredi puanlamada banka, kredi miktarı ve müşteriyle ilgili özelliklere dayanarak risk puanını hesaplar. Bu özellikler erişebildiğimiz verileri kapsar ve müşterinin finansal kapasitesini hesaplamaya yarayan gelir, birikimler, teminat, meslek, yaş, finansal geçmiş vb. gibi bilgilerden oluşur. Bu uygulamada da puanı hesaplamak için bilinen bir kural yoktur, yere ve zamana göre değişir. Bu yüzden en iyi yaklaşım yine veri toplamak ve puan hesaplama kuralını öğrenmeye çalışmaktır.

Kredi puanlaması bir regresyon uygulaması olarak tanımlanabilir; bir müşterinin puanının farklı niteliklerin ağırlıklı bir toplamı olarak yazıldığı doğrusal model geçmişte sıkça kullanılmıştır. Maaştaki her ek bin lira puanı X kadar artırır ve her ek bin lira borç puanı Y kadar azaltır. Tahmin edilen puana bağlı olarak banka farklı eylemlerde bulunulabilir. Örneğin, daha yüksek puana sahip bir müşterinin kredi kart limiti daha yüksek olabilir.

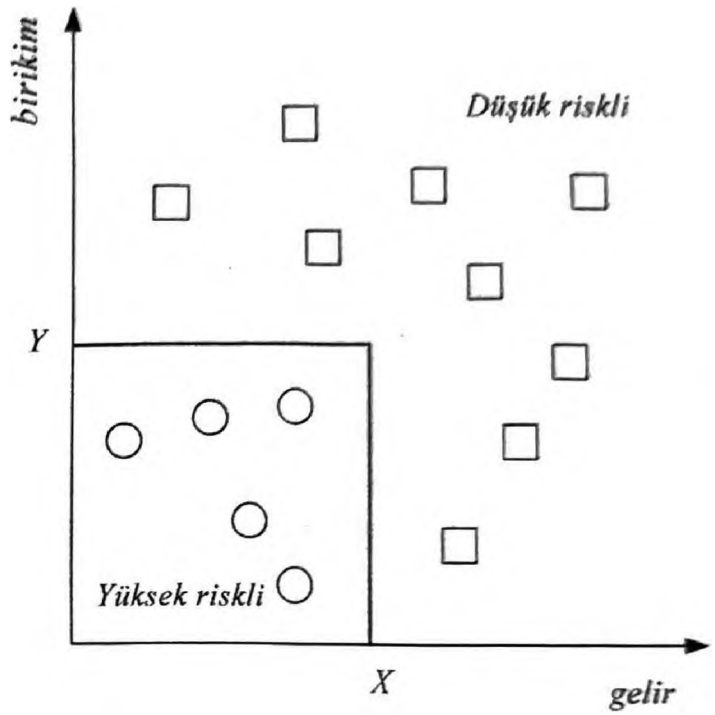
Kredi puanlaması, regresyon yerine, düşük riskli ve yüksek riskli olarak iki farklı müşteri sınıfının bulunduğu bir *sınıflandırma* problemi olarak da tanımlanabilir. Sınıflandırmada, regresyondaki sayısal değerin aksine, çıktı bir sınıf etiketidir ve bu da başka bir gözetimli öğrenme türüdür.

Bir *sınıf*, ortak bir özelliği paylaşan örneklerin kümesidir ve bunu bir sınıflandırma problemi olarak tanımladığımızda, tüm yüksek riskli müşterilerin düşük riskli müşteriler tarafından paylaşılmayan bazı ortak özellikleri paylaştığını ve her sınıfın bu özellikler cinsinden tanımlanmış *ayırtaç* adlı bir tanımı olduğunu varsayarız. Ayırtacı müşteri özellikleri tarafından tanımlanan uzayda iki sınıfın örneklerini ayıran bir sınır olarak hayal edebiliriz.

Her zaman olduğu gibi, burada da altta yatan asıl ayırtacı bilmiyoruz ve toplanan veriden öğrenmek istiyoruz.

Öğrenme verisini hazırlarken banka, geçmiş müşteri işlemlerini inceler ve borçlarını geri ödeyenleri düşük riskli, ödeyemeyenleri ise yüksek riskli olarak etiketler. İstediğimiz, bu verileri analiz ederek yüksek riskli müşteri sınıfını öğrenmektir; böylece gelecekte kredi için yeni bir başvuru olduğunda, müşterinin bu tanımla eşleşip eşleşmediğini kontrol ederek başvuruyu ret veya kabul edebiliriz.

Bir müşteriyle ilgili bilgi sınıflandırıcıya verilen girdiyi oluşturur; sınıflandırıcının görevi de bu girdiyi iki sınıftan birine atamaktır. Uygulama konusundaki bilgilerimizi kullanarak müşterinin gelirini ve birikimini girdi olarak almaya karar verdiğimizi düşünelim (bkz. Şekil 2.2). Bunları girdi olarak kullanmamızın nedeni bunların bize müşterinin güvenilirliği hakkında yeterli bilgi verdiğine inanmamızdır.



Şekil 2.2 Düşük ve yüksek riskli müşterileri ayırmak bir sınıflandırma uygulamasıdır. Gelir ve birikim eksenleri örneğin TL birimindendir. Her müşteri, gelir ve birikimine göre bu iki boyutlu alanda bir noktadır ve şekli sınıf etiketine bağlıdır; daire, o müşterinin yüksek riskli, kare de düşük riskli olduğunu gösterir. Tüm yüksek riskli müşterilerin geliri X, birikimi Y'den azdır ve şekilde kalın çizgiyle gösterilen ayırtaç bu iki sınıfa ait örnekleri ayırmak için kullanılabilir.

Ekonominin durumu hakkındaki tüm ayrıntılar ve müşteri hakkında olası tüm bilgiler de dahil olmak üzere, sonucu etkileyen tüm etkenleri tam bilmeyiz. Bu yüzden bir müşterinin düşük riskli mi yoksa yüksek riskli mi olacağı belirlenimci bir biçimde hesaplanamaz. Etkenlerin tümünün gözlemlenememesi rastsallık yaratır. Bu yüzden de gözlemleyebildiklerimizle yeni bir müşterinin düşük ya da yüksek riskli olup olmadığını tam olarak söyleyemeyiz; ancak bu iki sınıfın olasılıklarını hesaplayabilir ve sonra daha yüksek olasılığa sahip olanı seçebiliriz.

Olası bir model, ayırtacı eğer-ise [*if-then*] kuralları biçiminde tanımlar:

EĞER gelir <X VE birikim <Y İSE yüksek riskli

DEĞİLSE düşük riskli.

Burada X ve Y, bize veriyle kuralın yaptığı tahmini en iyi eşleştirmek üzere ayarlanacak parametrelerdir (bkz. Şekil 2.2). Önceki doğrusal modelde parametreler ağırlıklara karşılık geliyordu, bu eğer-ise modelinde ise parametreler bu eşik değerleridir.

Her eğer-ise kuralı, her biri girdi niteliklerinin birinde basit bir koşul olan terimlerden oluşan VE ile birleşik bir koşulu belirler. Kuralın öncülü VE ile bağlantılanmış terimleri içeren bir ifadedir. Yani kuralın uygulanması için tüm koşullar geçerli olmalıdır.

Bu kuraldan şunu anlıyoruz: Öncülü karşılayan –yani gelirleri X'ten *ve* birikimleri Y'den az olan– müşteriler altkümesi arasında düşük riskli müşterilerden çok daha fazla yüksek riskli müşteri vardır, bu yüzden bu tür müşteriler için yüksek risk olasılığı daha fazladır. Bu da kuralın sonucunun yüksek risk etiketli olarak tanımlanmasının nedenidir.

Bu basit örnekte yüksek riskli olmanın tek bir yolu vardır ve diğer tüm durumlar düşük risklidir. Başka bir uygulamada, her biri belirli bir bölgeyi sınırlayan eğer-ise kurallarından oluşan bir kural tabanı [*rule base*] bulunabilir ve her sınıf, bu gibi kuralların VEYA ile birleşimiyle belirtilir. Yüksek riskli olmanın farklı yolları vardır; bunların her biri bir kuralla belirlenir ve kurallardan herhangi birinin sağlanması yeterlidir.

Bu tür kuralları veriden öğrenmek *bilgi çıkarmayı* sağlar. Kural, verileri açıklayan basit bir modeldir ve bu modele bakarak verinin altında yatan süreçle ilgili bir açıklama elde ederiz. Örneğin, düşük riskli ve yüksek riskli müşterileri ayıran ayırtacı öğrendikten sonra düşük riskli müşterilerin özellikleri hakkında bilgimiz olur. Bu bilgileri potansiyel düşük riskli müşterileri daha verimli bir şekilde hedeflemek için, örneğin tüketiciye uyarlanmış reklamcılıkta kullanabiliriz.

Uzman Sistemler

Yapay öğrenme yaygın hale gelmeden önce uzman sistemler üstünde çalışılıyordu. 1970'lerde ortaya atılan ve 1980'lerde kullanılan

bu sistemler*, karar vermede insanlara yardım eden bilgisayar programlarıydı.

Bir *uzman sistem*, bir *bilgi tabanı* ve bir çıkarım [*inference*] motorundan oluşur. Bilgi, eğer-ise kuralları kümesi olarak gösterilir ve çıkarım motoru tümdengelim [*deduction*] için mantıksal çıkarım kurallarını kullanır. Kurallar, alan uzmanlarınca belirlendikten sonra programlanır ve sabittir. Alan bilgisini eğer-ise kurallarına dönüştürme işlemi zor ve maliyetliydi. Çıkarım motoru, özellikle mantıksal çıkarıma uygun olan Lisp ve Prolog gibi özel programlama dilleri kullanılarak programlanırdı.

Uzman sistemler 1980'lerde bir süreliğine tüm dünyada oldukça gözdeydi, yalnızca (Lisp'in kullanıldığı) ABD'de değil, (Prolog'un kullanıldığı) Avrupa'da da. Japonya'da, uzman sistemler ile yapay zekâ (YZ) için büyük ölçekli paralel mimariler üstüne Beşinci Nesil Bilgisayar Sistemleri Projesi vardı. Uygulamaları vardı ancak bulaşıcı hastalıkların tanısı için üretilen MYCIN gibi sınırlı alanlarda kullanılıyorlardı (Buchanan ve Shortliffe 1984). Bunların yanında ticari sistemler de bulunuyordu.

Araştırmaya ve yoğun ilgiye rağmen uzman sistemler asla çok başarılı olmadı. Bunun iki temel nedeni vardır. İlk olarak, verilerden öğrenme yoktu; bilgi tabanının çok zahmetli bir süreçle uzman insanlar tarafından elle oluşturulması gerekiyordu. İkinci neden ise, gerçek dünyayı temsil etmekte iki değerli mantığın elverişsizliğiydi. Gerçek hayatta şeyler doğru ya da yanlış değildir, farklı doğruluk derecelerine sahiptir: Bir kişi ya yaşlı ya da yaşlı olmayan değildir, yaşlılık yaşla birlikte artar. Mantıksal kurallar da farklı kesinlik dereceleriyle uygulanabilir: "Eğer X bir kuş ise X uçabilir" çoğunlukla doğru olsa da her zaman değildir.

Doğruluk derecelerini temsil etmek için bulanık üyelikler, bulanık kurallar ve çıkarım kullanan bulanık mantık [*fuzzy logic*] önerildi ve başlangıcından beri çeşitli uygulamalarda bir miktar başarı

* Çok başarılı bir yapay zekâ tarihçesi için, bkz. Nilsson 2009.

sağladı (Zadeh 1965). Belirsizliği temsil etmenin bir başka yolu, bu kitapta yaptığımız gibi olasılık teorisini kullanmaktır.

Bu kitapta ele aldığımız yapay öğrenme sistemleri, karar almada uzman sistemlerden iki açıdan daha güçlüdür. Birincisi, programlanmaları gerekmez ve örneklerden öğrenebilirler; ikincisi, olasılık teorisini kullandıkları için gürültüsü, istisnaları, muğlaklıkları ve sonuçta ortaya çıkan farklı belirsizlikleriyle gerçek dünya senaryolarını temsil etmekte daha başarılıdırlar.

Beklenen Değer Hesaplama

Bir karar aldığımızda –örneğin sınıflardan birini seçtiğimizde– doğru yapıyor ya da yanılıyor olabiliriz. Doğru kararların eşit derecede iyi olmaması ve yanlış kararların eşit derecede kötü olmaması sık rastlanan bir durumdur. Bir finans kurumu, kredi başvurusunda bulunan bir kişi hakkında karar alırken hem potansiyel kazancı hem de potansiyel zararı göz önünde bulundurmalıdır. Kabul edilen düşük riskli bir başvuru kârı artırırken, reddedilen yüksek riskli bir başvuru kaybı azaltır. Yanlışlıkla kabul edilen yüksek riskli bir başvuru sahibi kayba neden olabilir ve hatalı olarak reddedilen düşük riskli bir başvuru sahibi kaçırılmış bir kâr şansıdır.

Tıbbi tanı gibi alanlarda durum çok daha kritiktir ve simetrik olmaktan uzaktır. Burada girdiler, hasta hakkında sahip olduğumuz bilgiler, sınıflar ise hastalıklardır. Girdiler hastanın yaşını, cinsiyetini, geçmiş tıbbi tarihçesini ve o anki semptomlarını içerir. Bazı tetkikler hastaya uygulanmamış olabilir, yani bu girdiler eksik olabilir. Tetkikler zaman alır, masraflıdır ve hastaya rahatsızlık verebilir, bu yüzden bize değerli bilgi vereceklerine inanmadıkça uygulamak istemeyiz.

Tıbbi tanı durumunda yanlış bir karar, yanlış bir tedaviye ya da hiç tedavi uygulanmamasına yol açabilir ve farklı hata türleri aynı derecede kötü değildir. Diyelim ki hastanın belli bir hastalığı (örneğin bir kanser türü) olup olmadığına karar vermek istiyoruz. İki olasılık vardır: Hasta ya kanserlidir –buna pozitif sınıf diyelim– ya da değildir – negatif sınıf.

Benzer şekilde iki tür hata vardır. Sistem kanser tahmini yaptığı halde aslında hastada kanser yoksa bu bir *yanlış pozitiftir* – sistem pozitif sınıfı yanlışlıkla seçer. Bu kötü bir durumdur çünkü hem pahalı olan hem de hastaya rahatsızlık verecek gereksiz bir tedaviye neden olur. Eğer o kişi gerçekte hastayken sistem yok derse, bu da bir *yanlış negatiftir.*

Yanlış bir negatifin bedeli, yanlış bir pozitiften daha yüksektir, çünkü o zaman hasta gerekli tedaviyi alamaz. Bir yanlış negatifin bedeli, bir yanlış pozitifinkinden çok daha büyük olduğu için, pozitif sınıfın olasılığı görece düşük olsa bile –daha kapsamlı bir inceleme başlatmak üzere– pozitif sınıfını seçeriz. Bu, olasılığı ½'den yüksek olan sonucu –yazı ya da tura– seçtiğimiz bir yazı tura atışını tahmin etmeye benzemez.

Burada yaptığımız, olasılıkları kullanarak karar verdiğimiz ve kararımızın sonucu olarak karşılaşabileceğimiz olası zarar veya kazancı da dikkate alan *beklenen değer* hesaplamasının temelidir. Beklenen değer hesaplaması, sigortacılık gibi alanlarda sıklıkla yapılsa da insanların her zaman rasyonel davranmadıkları da bilinmektedir. Öyle olsaydı, kimse piyango bileti almazdı!

Max Frisch'in *Homo Faber* adlı romanında, yılan tarafından ısırılan bir kızın annesine endişelenmemesi söylenir, çünkü yılan ısırıklarından kaynaklı ölüm oranı yalnızca yüzde 3 ile 10 arasındadır. Kadın sinirlenir ve, "Yüz kızım olsaydı... o zaman sadece üç ila on kızımı kaybedecektim. İnanılmaz az! Oldukça haklısınız!" der ve devam eder: "Sadece bir çocuğum var." Etik meseleler söz konusu olduğunda, beklenen değer hesaplamasını kullanırken dikkatli olmamız gerekir.*

Hem yanlış pozitif hem de yanlış negatif yüksek bedele sahipse, olası bir üçüncü eylem kararı *reddetmek* ve kararı başka bir karar vericiye bırakmaktır. Örneğin, bilgisayar tabanlı tanı iki sonuç arasın-

* Beklenen değere veya beklenen faydaya dayalı kararın en iyi yol olamayabileceği bazı gerçek yaşam senaryoları için bkz. Sandel 2012. *Pascal'ın kumarı*, beklenen değer hesaplamasının yanlış bir konuya uyarlanmaya çalışılmasının bir başka örneğidir.

da seçim yapamazsa reddetmeyi seçebilir ve vakayla ilgili kararı doktora bırakabilir: İnsan uzman, sistem için doğrudan erişilebilir olmayan ek bilgileri kullanabilir. Örneğin, otomatik posta sıralayıcısında, sistem zarftaki sayısal posta kodunu tanımıyorsa, posta çalışanı adresi de okuyabilir.

3

ÖRÜNTÜ TANIMA

Okumayı Öğrenme

Farklı otomatik görüntü tanıma uygulamalarının farklı karmaşıklıkları vardır. En basitlerinden biri, bilginin farklı genişlikte, tanınması kolay şekiller olan çizgilerle gösterildiği *barkodların* okunmasıdır. Barkod, basit ve başarılı bir teknolojidir. Hem barkodları yazdırmak hem de onları okumak için tarayıcılar yapmak kolaydır. Bu yüzden hâlâ yaygın olarak kullanılıyorlar. Ancak barkod doğal bir gösterim değildir ve bilgi kapasitesi sınırlıdır. Son zamanlarda daha küçük bir alanda daha fazla bilginin kodlanabildiği iki boyutlu matris barkodlar da önerilmiştir.

Mühendislikte her zaman bir ödünleşim bulunur. Görevin çözülmesi zor olduğunda kısıtlayarak daha etkili çözümler geliştirebiliriz. Sözgelimi, tekerlek taşıma için çok iyi bir çözümdür, ancak düz yüzeyler gerektirir ve bu nedenle yol da yapmak gerekir. Kontrollü ortam görevi kolaylaştırır. Bacaklar çeşitli zeminlerde çalışır ancak üretilmeleri ve kontrolleri daha zordur ve düşük ağırlıktaki yükleri taşıyabilirler.

Optik karakter tanıma, basılı veya yazılı karakterleri görüntülerinden tanımaktır. Bu, barkodlardan daha doğal bir yaklaşımdır çünkü ek bir (çubuklar cinsinden) kodlama kullanılmaz. Tek bir font kullanılıyorsa her bir karakteri yazmanın tek bir yolu vardır; otomatik tanımayı kolaylaştırmak için özel olarak tanımlanmış OCR-A gibi standartlaştırılmış fontlar da bulunur.

Barkodlar veya tek bir font söz konusu olduğunda, her sınıf için tek bir görüntü şablonu bulunur ve öğrenmeye gerek yoktur. Her karakter için kolayca bellekte saklayabileceğimiz tek bir prototipimiz olur. Bu, karakterin ideal görüntüsüdür ve görülen girdiyi tüm prototiplerle tek tek karşılaştırıp prototiple en iyi eşleşen sınıfı seçeriz – buna *şablon eşleştirme* denir. Yazdırma veya algılamada ufak hatalar olabilir ancak en yakın eşleşmeyi bularak tanımayı hâlâ doğru olarak gerçekleştirebiliriz.

Çok sayıda font varsa ya da karakterler elle yazılmışsa, aynı karakteri yazmanın birden fazla yolu var demektir ve bu durumda bunların hepsini şablonlar olarak saklayamayız. Bunun yerine, aynı karakterin tüm farklı örneklerinin üzerinden geçerek sınıfı "öğrenmek" ve tüm örnekleri kapsayan genel bir tanım bulmak isteriz.

Yazı bir insan icadı olmasına rağmen ilginçtir ki "A"ların hepsini kapsayan ve "A" olmayanların hiçbirini kapsamayan bir "A" tanımımız yoktur. Bu olmadığı için, farklı yazı ve fontlardan örnekler alır ve bu örneklerden "A"lığın genel bir tanımını öğrenmek isteriz. Her ne kadar bir görüntüyü "A" sınıfının örneği yapan şeyin ne olduğunu bilmiyor olsak da farklı "A"ların hepsinde ortak bir şey olduğunu biliriz; örneklerden çıkarmak istediğimiz de bu şeydir.

Bir karakter görüntüsünün sadece rasgele noktalar ve farklı yönlerde çizgi parçalarından oluşmadığını, bir öğrenme programı kullanarak yakalayabileceğimize inandığımız bir düzenliliği olduğunu biliyoruz. Her karakter için farklı fontlarda (basılı metin için) veya yazılarda (elyazısı metin için) örnekler görüp genelleştirme yaparız; yani bir karakterin tüm örneklerince paylaşılan bir tanım buluruz: "A" belirli türlerde çizgileri birleştirmenin bir yoludur, "B" bir başka yoldur, vs.

Basılı karakterlerin tanınması, Latin alfabesi ve onun çeşitli türevlerinde nispeten kolaydır; daha fazla karakter, aksan ve yazma stilinin olduğu alfabelerde ise daha zordur. Bitişik elyazısında karakterler birbirine bağlıdır ve bu yüzden tanıma öncesinde bölütleme (karakter görüntülerini birbirinden ayırma) gerekir.

Birçok farklı font vardır, insanların elyazıları da birbirinden çok farklı olabilir. Karakterler ayrıca küçük veya büyük, eğimli, mürekkeple basılmış veya kalemle yazılmış olabilir ve bunun sonucunda aynı karaktere karşılık birçok olası görüntü ortaya çıkabilir. Yapılan tüm araştırmalara rağmen bugün hâlâ bu iş için bile insanlar kadar başarılı tanıma yapan bir bilgisayar programı bulunmamaktadır. Bu nedenle "*captcha*"lar hâlâ kullanılıyor. Bir *captcha*, web sayfalarında kullanıcının insan olduğunu ve bilgisayar programı olmadığını kanıtlamak için tanınması gereken sözcük veya sayılar içeren bozulmuş bir görüntüdür.

Model Karmaşıklığını Ayarlama

Yapay öğrenmede amaç, veriye bir model uydurmaktır. İdeal durumda, tüm örnekler için geçerli tek bir *küresel* [*global*] modelimiz vardır. İkinci bölümde gördüğümüz örnekte tüm otomobillerin fiyatını tahmin etmek için kullanabileceğimiz tek bir regresyon modelimiz vardı. Böyle bir durumda model tüm eğitim örnekleriyle eğitilmiştir ve tüm bu örneklerin model parametreleri üzerinde etkisi vardır. İstatistikte buna *parametrik kestirim* denir.

Parametrik model iyidir çünkü basittir –tek bir model saklar ve kullanırız– ve verilerin tümüyle eğitilir. Ne yazık ki bu model kısıtlayıcı olabilir, çünkü tüm durumlara uygulanabilir tek bir model varsayımı bütün uygulamalarda geçerli olmayabilir. Bazı uygulamalarda, her biri belirli türden örnekler için geçerli olan bir dizi *yerel* modelimiz olabilir. Bu, *yarı parametrik kestirimdir*. Hâlâ girdiyi çıktıyla eşleştiren bir modelimiz vardır ama yalnızca yerel olarak geçerlidir ve farklı girdi türleri için farklı modellerimiz olur.

Örneğin, kullanılmış otomobillerin fiyatını tahmin ederken

farklı otomobil türleri için değer kaybı davranışının farklı olduğuna inanmak için bir nedenimiz varsa, sedanlar için bir model, spor otomobiller için bir başka model ve lüks otomobiller için bir başka modelimiz olabilir. Bu tür bir yaklaşımda verilerin yerel alanlara gruplandırılması ve her bir yerel bölgedeki modellerin eğitimi birleştirilmiş biçimde bir arada yapılır ve her bir yerel model yalnızca kendi kapsamına giren eğitim verisiyle eğitilir. Yerel modellerin sayısı, model esnekliğini ve dolayısıyla karmaşıklığı tanımlayan üst parametredir.

Bazı uygulamalarda yarı parametrik varsayım bile yetmez; yani veriler birkaç yerel modelle açıklanamaz. Böyle bir durumda, genel ya da yerel olarak basit bir model kullanamadığımız yerlerde, diğer uçtaki *parametrik olmayan kestirimi* kullanırız. Kullandığımız tek bilgi, yapabileceğimiz en temel varsayım şudur: Benzer girdilerin benzer çıktıları vardır. Bu durumda, eğitim verisini model parametrelerine dönüştüren ayrı bir eğitim sürecimiz yoktur. Bunun yerine eğitim verisini geçmişteki örneklerin bir kaydı olarak saklar ve gerektiğinde kullanırız.

Bir örnek verildiğinde, öğrenme verisi içinde ona en çok benzeyen örnekleri buluruz ve bu örneklerin bilinen çıktılarını kullanarak bir çıktı hesaplarız. Örneğin, fiyatını tahmin etmek istediğimiz bir otomobil verildiğinde, tüm öğrenme örnekleri içinden ona en çok benzeyen üç otomobili (kullandığımız özellikler açısından) buluruz ve bu üç otomobilin fiyatlarının ortalaması tahminimiz olur. Bunlar, nitelikleri açısından girdiye en çok benzeyen otomobiller olduklarından, fiyatlarının da benzer olacağını düşünmek mantıklıdır. Buna en yakın k komşu kestirimi [*k-nearest neighbor estimation*] denir ve burada k üçtür. Bunlar, o anki girdiye en benzer üç geçmiş "vaka" olduğundan bu yaklaşıma bazen vaka temelli akıl yürütme [*case-based reasoning*] denir. En yakın komşu algoritması akla yakın bir yöntemdir: Benzer örneklerin anlamı da benzerdir. Hepimiz komşularımızı severiz, çünkü bize benzerler – ya da nefret ederiz, tam olarak aynı nedenden ötürü.

Üretici Modeller

Veri analizinde son zamanlarda yaygın olan bir yaklaşım, verilerin nasıl üretildiği hakkındaki inancımızı temsil eden bir *üretici model* düşünmektir. Gözlemlediğimiz verileri oluşturmak için etkileşime giren bir dizi *gizli neden kullanan* gizli bir model olduğunu varsayarız. Gözlemlediğimiz veri büyük ve karmaşık olsa da birkaç değişken tarafından kontrol edilen basit bir süreç aracılığıyla üretiliyor olabilir; saklı etkenler olan bu değişkenleri kestirebilirsek veri çok daha basit bir biçimde gösterilip anlaşılabilir. Böylesi bir model aynı zamanda doğru tahminlerde de bulunabilir.

Optik karakter tanımayı düşünün. Her karakter görüntüsünün iki tür üretici etkenden oluştuğunu söyleyebiliriz: Görüntüyü belirleyen kaynak olarak öncelikle karakterin bir kimliği/özdeşliği [*identity*] yani etiketi bulunur; buna ek olarak yazma ya da baskı alma sırasında (ve belki taranması/algılanmasında) gelen etkiler vardır.

Basılı bir metinde görünüm fonttan kaynaklanıyor olabilir; örneğin, Times Roman fontundaki karakterler, hepsi aynı genişlikte olmayan çizgilerle yazılır. Fontun estetik bir yanı da vardır; kaligrafide özellikle öne çıkan bu estetik kısımdır. Ancak görünüm nedeniyle eklenen bu özellikler, kimlik hakkında kafa karışıklığına neden olacak kadar büyük olmamalıdır. Tıpkı basılı metinlerdeki font seçimi gibi, yazarın elyazısı stili de yazılı metinde farklılıklar yaratır. Görünüm aynı zamanda yazarın kullandığı malzemeye (tükenmez ya da kurşunkalem) ve ortama (mermer levha ya da kâğıt) bağlı olabilir.*

Basılı veya yazılı karakter büyük ya da küçük olabilir ve genellikle bir ön işleme aşamasında karakter görüntüsü sabit bir büyüklüğe dönüştürülerek normalleştirilir – bunu yapabilmemizin nedeni büyüklüğün kimliği etkilemediğini bilmemizdir. Buna değişmezlik [*invariance*] denir. Büyüklüğe (metin 12 veya 18 punto olsun, içerik

* Burada girdinin bir görüntü olduğu optik karakter tanımadan bahsediyoruz. Ayrıca, yazmanın dokunmaya duyarlı bir tablette gerçekleştirildiği kalem tabanlı karakter tanıma da vardır. Böyle bir durumda girdi bir görüntü değildir, karakter yazılırken kalemin tablet üstünde temas ettiği (x,y) koordinatlarının dizisidir.

aynıdır), eğime (metin italik olduğunda görüldüğü gibi) ya da çizgilerin genişliğine (kalın karakterlerde olduğu gibi) karşı değişmezlik isteriz. Gelgelelim, istemediğimiz durumlar da olabilir: Örneğin, bir miktar döndürmeye karşı değişmezlik isteriz ama çoğuna değil: *q*, döndürülmüş bir *b*'dir.

Sınıfı tanımak için kimliğe odaklanmamız, yani kimliği temsil eden nitelikleri bulmamız ve bunları karakterleri tanımlamak için nasıl birleştireceğimizi öğrenmemiz gerekir. Görünüşle ilgili tüm özellikleri yani yazar, estetik, ortam ve algılamanın etkilerini görmezden gelmeyi öğreniriz. Fakat, farklı bir görevde bunların önemli şeyler olabileceği unutulmamalıdır; örneğin elyazısından kimlik doğrulaması veya imza tanıma yaparken önemli olan, yazılanlar değil yazara özgü özelliklerdir.

Üretici model, verinin saklı etkenler tarafından nasıl üretildiğini açıklamasından dolayı *nedenseldir*. Böyle bir modeli eğittiğimizde, onu ters yönde, yani gözlemden nedene doğru *tanı* için kullanmak isteyebiliriz. Tıbbi alan burada iyi bir örnektir: Hastalıklar nedenlerdir ve saklıdır; belirtiler, örneğin, tıbbi tetkiklerin sonuçları, hastanın gözlenen nitelikleridir. Hastalıktan semptoma gitmek nedensel yöndür – bu hastalığın yaptığı şeydir. Belirtiden hastalığa gitmek ise tanısaldır, bu ise doktorun yaptığı şeydir. Genel olarak söylersek tanı, gözlenen değişkenlerden saklı etkenlerin kestirilmesidir.

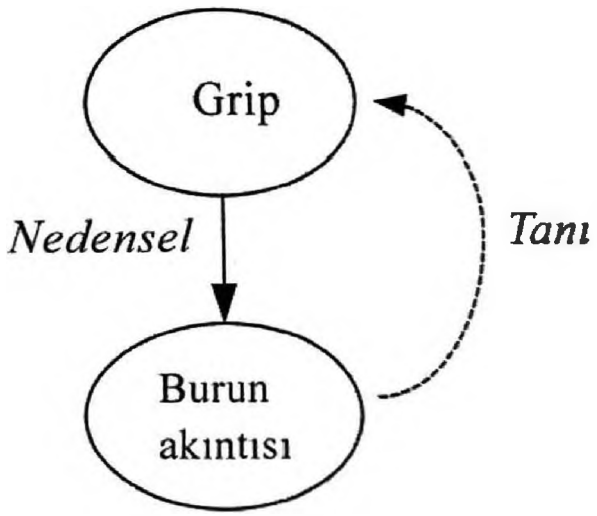
Üretici bir model, saklı ve gözlenen değişkenlere karşılık gelen düğümlerden oluşan bir çizge olarak temsil edilebilir ve düğümler arasındaki yaylar, aralarındaki nedensellikler gibi bağımlılıkları temsil eder. Bu tür çizgesel *modeller*, uygulamanın görselleştirilmesine izin vermesi bakımından ilgi çekicidir; ayrıca istatistiksel çıkarım ve kestirim yöntemlerinin, iyi bilinen ve etkili çizge işlemleriyle gerçeklenebilmesini sağlar (Koller ve Friedman 2009).

Örneğin, nedensel bir bağlantı, saklı etkenden gözlenen belirtiye doğru ilerlerken, tanı, bağlantı yönünü ters çevirir. Bağımlılığı modellemek için koşullu olasılık kullanırız. Hasta gripse burun akıntısı olması koşullu olasılığından söz ettiğimizde nedensel yön-

de gideriz: Grip (belli bir olasılıkla) burun akıntısına neden oluyordur.

Eğer bir kişinin burun akıntısı olduğunu biliyorsak grip olup olmadığına karar vermek için koşullu olasılığı ters yönde hesaplamamız gerekir – yani burun akıntısının verili olduğu koşulda grip olma olasılığını hesaplarız. (bkz. Şekil 3.1). Ters yönlerdeki bu iki koşullu olasılık, *Bayes kuralı** ile birbiriyle ilişkilidir ve bu yüzden çizgesel modellere bazen *Bayes'çi ağlar* da denir. İlerleyen bölümlerde Bayes'çi kestirime geri döneceğiz ve bu tür ağlara model parametrelerini de dahil edebileceğimizi ve bunun bize hem öğrenme hem çıkarımda ek esneklik sağladığını göreceğiz.

Bir metin okuyorsak, yararlanabileceğimiz kaynaklardan biri de



Şekil 3.1 Gribin burun akıntısının nedeni olduğunu gösteren çizge modeli. Hastanın burnunun aktığını biliyorsak grip olup olmama olasılığını hesaplamak için Bayes kuralını kullanarak ters yönde çıkarım yapar, yani tanı koyarız. Daha karmaşık bağlantıları göstermek için daha fazla düğüm ve bağlantıyla daha büyük çizgeler tanımlayabiliriz.

* G gribi, B ise burun akıntısını temsil etsin. Bayes kuralını kullanarak şunu yazabiliriz:
$P(G \mid B) = P(B \mid G)\, P(G) / P(B)$,
Burada, $P(B \mid G)$, grip olduğu bilinen bir hastanın burnunun akmasının koşullu olasılığıdır. $P(G)$, burun akıntısı olup olmadığına bakılmaksızın hastanın grip olma olasılığı ve $P(B)$ hastanın gribinin olup olmadığına bakılmaksızın burun akıntısına sahip olma olasılığıdır.

dilbilgisidir. Her sözcük bir karakter dizisidir ve nadiren rasgele bir karakter dizisi yazarız; genellikle söz konusu dilin sözlüğünden bir sözcük seçeriz. Bu sayede bir karakterini tanıyamasak bile sö?cüğü okuyabiliriz. Bu tür bağlamsal bağımlılıklar, daha yüksek düzeylerde, dilin sözdizimsel ve anlamsal kuralları tarafından tanımlandığı şekliyle sözcükler ve cümleler arasında da olabilir. Birazdan ele alacağımız gibi, yapay öğrenme algoritmaları doğal dil işlemede bu tür bağımlılıkları öğrenebilmemize yardımcı olur.

Yüz Tanıma

*Yüz tanıma*da girdi, kamera tarafından çekilen görüntü ve sınıflar da tanınacak kişilerdir ve öğrenme programı, yüz görüntülerini kimlikleriyle eşleştirmeyi öğrenir. Bu, optik karakter tanıma işleminden daha zordur çünkü girdi görüntüsü daha büyüktür. Ek olarak, bir yüz neredeyse üç boyutludur ve poz veya aydınlatmadaki farklılıklar görüntüde önemli değişikliklere neden olur. Yüzün bazı kısımları kapalı olabilir; gözlükler gözleri ve kaşları, sakal çeneyi gizleyebilir.

Karakter tanımada olduğu gibi, yüz görüntüsünü etkileyen iki ayrı etken kümesi düşünebiliriz: Kimliği tanımlayan özellikler yanında kimliği etkilemeyen ancak saç modeli veya ifade (yani nötr, gülümseyen, kızgın vb.) gibi görünümü etkileyen özellikler vardır. Bu görünüm özellikleri, kaydedilen yüz görüntüsünü etkileyen aydınlatma kaynağı ya da poz gibi saklı etkenlerden de kaynaklanabilir. Kimlikle ilgileniyorsak sadece birinci tür özellikleri kullanan, ikinci türdeki özelliklerden etkilenmeyen bir yüz tanımı öğrenmek isteriz.

Bununla birlikte, başka uygulamalarda ikinci tür özellikler önemli olabilir. Yüz ifadesini tanımak, kimlik yerine ruh halini veya duyguları tanımamızı sağlar. Örneğin, bir toplantının videosunu izlerken katılımcıların ruh halini takip etmek isteyebiliriz. Benzer şekilde, çevrimiçi eğitimde materyali sunma hızını daha iyi ayarlamak için öğrencinin kafasının karışıp karışmadığını anlamak önemlidir. Hızla popüler hale gelen bir alan olan duygulanımsal bilgiişlemde

[*affective computing*] amaç, kendilerini kullanıcının ruh haline uyarlayabilen bilgisayar sistemleri geliştirmektir.

Amaç (örneğin, güvenlik için), insanların kimliklerinin tespit edilmesi veya doğrulanması ise, yüz görüntüsünün kullanılması olasılıklardan yalnızca biridir. *Biyometri*, fizyolojik ve/veya davranışsal özellikler kullanılarak insanların tanınması veya kimliklerinin doğrulanmasıdır. Yüze ek olarak, olası fizyolojik özellikler, parmak izi, iris ve avuç içi olabilir. Davranışsal özelliklerin örnekleri arasında ise imza, ses, yürüyüş ve tuş vuruşu dinamikleri bulunur. Daha doğru kararlar için böylesi farklı kiplerden girdiler birleştirilebilir. Fotoğraf, imza veya şifre gibi yaygın kimlik tespit yöntemlerinin aksine çok sayıda farklı kipte girdi birlikte kullanıldığında, dolandırıcılık (sahtecilik) çok daha zor, tanıma çok daha doğru hale gelir; tabii ki bunu yaparken kullanıcılar için çok ek sıkıntı yaratmamaya dikkat etmek gerekir.

Konuşma Tanıma

Konuşma tanımada girdi, bir mikrofon tarafından algılanan akustik sinyal, sınıflar da belirli bir dilde söylenebilecek sözcüklerdir. Bu durumda öğrenilecek ilişki, akustik sinyal ile o dildeki bir sözcük arasındadır.

Her karakterin farklı açılı çizgi parçaları gibi temel öğelerden oluştuğunu düşündüğümüz gibi, sözcük de temel konuşma ses parçacıkları olan fonemlerin bir dizisi olarak tanımlanabilir. Konuşma durumunda girdi *zamansaldır*; sözcükler zaman içinde bu fonemlerin bir dizisi olarak söylenir ve bazı sözcükler diğerlerinden daha uzundur.

Farklı insanlar, yaş, cinsiyet veya aksandaki farklılıklar nedeniyle aynı sözcüğü farklı telaffuz eder ve yine her bir sözcüğe karşılık gelen akustik sinyalin iki etken kümesinden oluştuğunu düşünebiliriz: sözcükle ilişkili olanlar ve konuşmacıyla ilişkili olanlar. Konuşma tanıma, birinci türdeki özellikleri kullanırken; konuşmacı doğrulama, ikinci türdeki özellikleri kullanır. Bu arada, bu ikinci tür

(konuşmacıyla ilgili) özelliklerin tanınması veya yapay olarak üretilmesi kolay değildir; bu nedenle konuşma sentezleyicilerin çıktısı kulağa hâlâ "robotik" gelmektedir.*

Tıpkı biyometride olduğu gibi, burada da araştırmacılar çok sayıda veri kaynağını birleştirme fikrini kullanıyor. Akustik bilgiye ek olarak, sözcükleri seslendirirken konuşmacının dudaklarının ve ağız şeklinin video görüntüsünü de tanıma sırasında kullanabiliriz.

Doğal Dil İşleme ve Çeviri

Konuşma tanımada, optik karakter tanımada olduğu gibi bağlamsal bilgileri dikkate alan bir *dil modeli* kullanmak başarıyı önemli ölçüde artırır. Hesaplamalı dilbilimde, programlanmış kurallar üzerine yıllarca süren araştırmalar, (dilin sözcüksel, sözdizimsel ve semantik kurallarını tanımlayan) bir dil modeli oluşturmanın en iyi yolunun, onu büyük bir örnek veri kümesinden öğrenmek olduğunu gösterdi. Yapay öğrenmenin *doğal dil işlemeye* yönelik uygulamaları sürekli artmaktadır (son araştırmalar için bkz. Hirschberg ve Manning 2015).

En kolay uygulamalardan biri, *istenmeyen e-posta filtrelemedir*; burada bir yanda istenmeyen posta üreticileri diğer yanda ise filtreler birbirlerini alt etmek için ustalıklı yollar bulmaya devam etmektedir. Bu, istenmeyen ve geçerli e-postalar olarak ikiye ayrılabilecek bir sınıflandırma problemidir. Benzer bir uygulama da herhangi bir belgeyi, kültür, sanat, politika gibi sınıflardan birine atamak istediğimiz *belge sınıflandırmadır.*

Bir yüz bir görüntüdür ve söylenmiş bir cümle akustik bir sinyaldir, fakat bir metin neden oluşur? Metin, bir karakter dizisidir ve karakterler bir alfabeyle tanımlanır ama bir dil ile kullanılan alfabe arasındaki ilişki basit değildir. Bir cümlenin neredeyse hiçbir zaman tek başına bulunmadığı ya da tek başına yorumlanmaması gerektiği,

* Birçok bilimkurgu filminde robotlar görüntü ve konuşma tanıma veya otonom hareket açısından çok ileri düzeyde olsalar da ilginç bir şekilde "robotik" bir sesle konuşmaya devam ediyorlar.

aksine bir diyaloğun ya da genel bir bağlamın parçası olduğu gerçeği bir yana, insan dili mizah ve hiciv gibi incelikleriyle farklı düzeylerdeki sözcüksel, sözdizimsel ve anlamsal kurallar içeren çok karmaşık bir bilgi gösterim biçimidir.

En yaygın metin gösterim yöntemi, *sözcük torbası*dır. Önce bir sözcük dağarcığı tanımlar, ardından her belgeyi belgede *herhangi bir yerde* görülen sözcüklerin listesi olarak temsil ederiz. Yani belgede seçtiğimiz sözcüklerin hangilerinin göründüğünü, hangilerinin görünmediği bilgisini kullanırız. Sözcüklerin metin içinde nerede olduğu bilgisini kaybederiz ki bu uygulamaya bağlı olarak iyi ya da kötü olabilir. Bir dağarcık seçerken o uygulama için bilgi veren sözcükleri seçeriz; örneğin, istenmeyen e-posta filtrelemede "fırsat" ve "teklif" gibi sözcükler ayırt edicidir. Soneklerin (örneğin "-mak", "-lar") kaldırıldığı ve bilgi içermeyen sözcüklerin (örneğin "bu", "ile") görmezden gelindiği bir ön işleme adımı da vardır.

Sosyal medyadaki mesajların analizi son zamanlarda yapay öğrenmenin önemli bir uygulama alanı haline geldi. En çok konuşulan konuları [*trending topics*] çıkarmak bunlardan biridir: Bu, sosyal medya mesajlarında sıklıkla ortaya çıkmaya başlayan belirli bir özgün sözcük kombinasyonunu ifade eder. Başka bir görev de ruh hali tanıma, yani bir müşterinin bir üründen (örneğin bir politikacıdan) memnun olup olmadığını belirlemektir. Bunun için, sözcük torbası gösterimini kullanarak iki sınıfın –mutlu veya mutlu değil– göstergesi olan sözcükleri içeren bir sözcük dağarcığı tanımlanabilir ve bunların sınıf tanımlarını nasıl etkiledikleri öğrenilebilir.

Yapay öğrenmenin belki de en etkileyici uygulaması *yapay çeviridir* (ya da daha doğrusu olacaktır). Elle kodlanan çeviri kuralları üzerine onlarca yıllık araştırmanın ardından, en umut vaat eden yaklaşımın her iki dilde bolca metin çifti örneklemi toplamak ve birini diğeriyle eşleştiren kuralları bu veriden otomatik olarak çıkaracak bir öğrenen program kullanmak olduğu anlaşıldı. Kanada gibi iki dilli ülkelerde ve çok sayıda resmî dili olan Avrupa Birliği'nde aynı metni iki veya daha fazla dile özenle çevrilmiş olarak bulmak nispeten kolaydır. Bu tür veriler, çeviriye yönelik yapay öğrenme yaklaşımları tarafından yoğun olarak kullanılmaktadır.

Dördüncü bölümde, doğal dil işleme için gerekli olan farklı soyutlama katmanlarını otomatik olarak öğrenmesi bakımından fazlasıyla umut vaat eden *derin öğrenmeyi* tartışıyoruz.

Çok Sayıda Modeli Birleştirmek

Bir uygulamada çeşitli öğrenme algoritmalarından herhangi biri kullanılabilir ve genelde yapıldığı gibi hepsini deneyip en iyi olanı seçmek yerine hepsini kullanmak ve tahminlerini birleştirmek daha iyi bir yaklaşım olabilir. Bu, öğrenmedeki rastsallığı giderir ve daha iyi başarı sağlayabilir.

Amaç, birbirini tamamlayan olabildiğince farklı modeller bulmaktır. Bunu elde etmenin bir yolu, modellerin farklı bilgi kaynaklarına bakmalarını sağlamaktır. Bunu yüz, parmak izi vb. gibi farklı özelliklere baktığımız biyometride ve akustik konuşma sinyaline ek olarak görüntüden konuşmacının dudağını takip ettiğimiz konuşma tanımada görmüştük.

Günümüzde verilerimizin büyük kısmı çoğulortamlıdır ve çoklu görü [*multi-view*] modelleri, farklı ancak tamamlayıcı bilgiler sağlayan farklı sensörlere sahip olduğumuz uygulamalarda kullanılabilir. Görüntü erişiminde [*retrieval*], görüntünün kendisine ek olarak bir metin açıklaması veya bir etiket sözcük kümesi de bulunabilir. İki kaynağı bir arada kullanmak daha iyi başarı sağlar. Akıllı saatler ve akıllı telefonlar gibi akıllı cihazlarımızda sensörler bulunur ve bunların değerleri, örneğin *etkinlik tanıma* amacıyla birleştirilebilir.

Aykırılık [*outlier*] Tespiti

Yapay öğrenmenin bir başka uygulama alanı, amacın genel kurala uymayan örnekleri bulmak olduğu *aykırılık tespitidir*; bunlar belirli bağlamlarda bilgilendirici olan istisnalardır. Buradaki fikir, tipik örneklerin basitçe ifade edilebilecek özellikleri paylaşması ve bunlara uymayan örneklerin atipik olmasıdır.

Anna Karenina romanında Tolstoy, "Mutlu ailelerin hepsi birbi-

rine benzer, ancak her mutsuz aile kendine özgü bir biçimde mutsuzdur," diye yazar. Bu, sadece on dokuzuncu yüzyıl Rus aileleri için değil, pek çok alan için geçerlidir. Örneğin tıbbi tanıda da, tüm sağlıklı insanların aşağı yukarı aynı olduğunu ve sağlıksız olmanın farklı yollarının bulunduğunu söyleyebiliriz. Bunların her biri bir hastalıktır.

Böyle bir durumda model tipik örnekleri kapsar ve ardından dışarıda kalan bir örnek, istisna niteliği taşır. Bir *aykırılık*, örneklemdeki öteki örneklerden çok farklı olan bir örnektir. Bir aykırılık, sistemin anormal davranışını, örneğin bir kredi kartı işlemi söz konusu olduğunda *dolandırıcılığı* gösterebilir. Bir görüntüde aykırılık, bir tümör gibi dikkat gerektiren bir *anomaliyi* gösterebilir. Ağ trafiği örneğinde aykırı bir işlem, bir bilgisayar korsanlığı denemesine işaret edebilir. Bir sağlık hizmeti senaryosunda hastanın normal davranışından önemli bir sapmayı gösterebilir. Aykırılıklar, güvenilir istatistikler elde etmek için tespit edilip çıkarılması gereken (hatalı sensörler yüzünden ortaya çıkmış) kayıt hataları da olabilir. Bir aykırılık ayrıca daha önce görülmemiş fakat geçerli bir yenilik de olabilir; buna *yenilik tespiti* diyoruz. Örneğin daha önce görülmemiş ve şirket tarafından hedef alınmayı bekleyen yeni bir kârlı müşteri türü söz konusu olabilir.

Boyut İndirme

Herhangi bir uygulamada bilgi içerdiğine inandığımız gözlemlenen veri özellikleri girdi olarak alınmakta ve karar almada kullanılmaktadır. Bununla birlikte, bu özelliklerden bazılarının aslında bilgilendirici olmadığı ve göz ardı edilebileceği bir durum olabilir; örneğin kullanılmış bir otomobilin renginin fiyatı üzerinde önemli bir etkisi olmadığı ortaya çıkabilir. Veya iki farklı niteliğin birbiriyle yüksek ilintiye sahip olduğu ve temelde aynı bilgiyi verdiği söylenebilir (kullanılmış bir otomobilin üretim yılıyla kilometresi yüksek düzeyde ilintilidir) ve bu niteliklerden yalnızca birini tutmak yeterli olabilir.

Ayrı bir ön işleme adımında *boyut indirmek*, birkaç nedenden dolayı ilgimizi çeker.

İlk olarak, çoğu öğrenme algoritmasında hem modelin hem de eğitim algoritmasının karmaşıklığı girdi niteliklerinin sayısına bağlıdır. Burada karmaşıklık iki türlüdür: ne kadar hesaplama yaptığımızı gösteren zaman karmaşıklığı ve gerek duyduğumuz bellek miktarını gösteren uzam karmaşıklığı. Girdi sayısının azaltılması her zaman her ikisini de azaltır, ancak ne kadar azaldıkları söz konusu modele ve öğrenme algoritmasına bağlıdır.

İkincisi, bir girdinin gereksiz olduğu anlaşıldığında onu ölçmenin maliyetinden de tasarruf ederiz. Örneğin tıbbi tanıda belirli bir tetkikin gerekli olmadığı ortaya çıkarsa bunu yapmayız; böylece hem maliyeti hem de hastaya verebileceğimiz rahatsızlığı ortadan kaldırırız.

Üçüncüsü, basit modeller küçük veri kümelerinde daha gürbüzdür [*robust*]; yani daha az veriyle eğitilebilirler veya aynı miktarda veriyle eğitim aldıklarında daha küçük değişinti (belirsizlik) gösterirler.

Dördüncüsü, veriler daha az özellikle açıklanabildiği zaman yorumlanması daha kolay olan daha basit bir modele sahip oluruz.

Beşincisi, veriler birkaç (örneğin iki) boyutta gösterilebildiğinde, görsel olarak çizilebilir ve böylece yapısı ve aykırılıklar gözle analiz edilebilir; bu da veriden saklı bilgi elde etmeyi kolaylaştırır. Bir çizim bin nokta değerindedir ve verileri görselleştirmek için iyi bir yol bulabilirsek görme korteksimiz herhangi bir ek model uydurma hesabına gerek kalmadan gerisini halledebilir.

Boyut indirmenin temelde iki yolu vardır: özellik seçimi ve özellik çıkarımı. *Özellik seçiminde* önemli özellikleri korur ve önemsiz olanları atarız. Temel olarak girdi nitelikleri kümesinin en yüksek başarı getiren en küçük altkümesini seçmeyi hedefleriz. Özellik seçimi için en yaygın kullanılan yöntem, daha fazla iyileştirme yapılmayana kadar arka arkaya özellik eklediğimiz sarmalayıcı [*wrapper*] yaklaşımıdır. Özellik seçici, her bir altkümede eğitilen temel sınıflandırıcı ya da regresyon modelin "etrafına sarılır".

Özellik çıkarımında ise asıl özelliklerden hesaplanan yeni özellikler tanımlarız. Bu yeni hesaplanan özellikler sayıca daha azdır, ama asıl özelliklerde bulunan bilgiyi kaybetmez. Sentezlenmiş bu birkaç yeni özellik, verileri asıl özelliklerin herhangi birinden daha iyi açıklar ve bazen saklı veya soyut kavramlar olarak da yorumlanabilir.

İzdüşüm yöntemlerinde her yeni özellik, asıl özelliklerin doğrusal bir bileşimidir. Böyle yöntemlerden biri, verideki değişintiyi en iyi koruyan yeni özellikler bulduğumuz *temel bileşen analizidir.* Değişinti büyükse örnekler arasında farklar en belirgindir, küçük olması durumunda veriler arasındaki farkları kaybederiz. Bir başka yöntem olan doğrusal *ayırtaç analizi*, amacı sınıflar arasındaki ayrımı en üst düzeye çıkaran yeni özellikler bulmak olan gözetimli bir özellik çıkarma biçimidir.

Bir uygulamada, özellik seçme mi yoksa çıkarma mı kullanılması gerektiği uygulamaya ve özelliklerin ayrıntı derecesine bağlıdır. Kredi puanlaması yapıyorsak ve müşterinin yaşı, geliri, mesleği vb. özelliklere sahipsek, özellik seçimi yapmak anlamlıdır. Her özelliğin kendi başına bilgilendirici olup olmadığını söyleyebiliriz. Ancak özellik izdüşümü burada anlamsızdır: Yaş, gelir ve mesleğin doğrusal bir bileşimi (ağırlıklı toplamı) anlamlı değildir. Öte yandan, yüz tanıma yapıyorsak ve girdiler piksel ise, özellik seçimi bir anlam ifade etmez – tek başına bir piksel ayırt edici bilgiler içermez. Özellik çıkarmada olduğu gibi, bir yüzü tanımlarken belirli piksel *bileşimlerine* bakmak daha mantıklıdır.

Doğrusal olmayan boyut indirme yöntemleri, doğrusal bir bileşimin ötesine geçer ve daha iyi özellikler bulabilir. Bu, yapay öğrenmede en öne çıkan araştırma alanlarından biridir. İdeal özellik kümesi en az sayıda özellik kullanarak veri kümesindeki (sınıflandırma veya regresyon için) bilgiyi en iyi şekilde temsil eder ve bu bağlamda bir kodlama işlemi olarak görülebilir. Bu aynı zamanda bir soyutlama işlemi olarak da düşünülebilir, çünkü bu yeni özellikler veriyi daha özlü bir şekilde temsil eden üst düzey özelliklere karşılık gelebilirler. Dördüncü bölümde, bu tür doğrusal olmayan özellik çıkarmanın yapay sinir ağlarında öğrenildiği özkodlayıcı ağlar ve derin öğrenme konusunu ele alacağız.

Karar Ağaçları

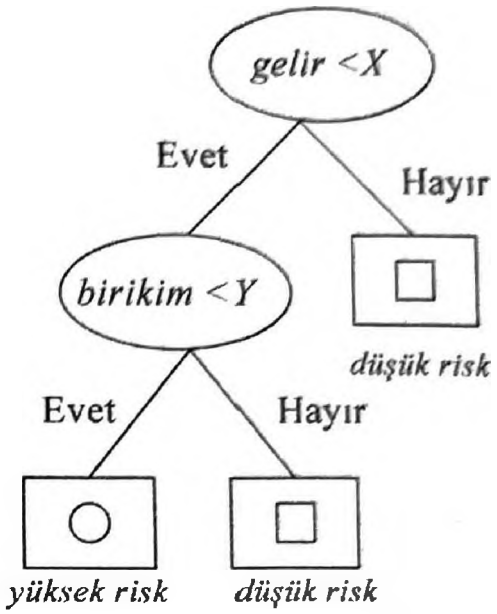
Daha önce ele aldığımız eğer-ise kurallarını öğrenmenin bir yolu da karar ağaçlarıdır. *Karar ağacı*, yapay öğrenmedeki en eski yöntemlerden biridir ve hem eğitim hem de sınamada kullanımı basit olsa da birçok alanda başarıyla kullanılır. Ağaçlar, Caesar'dan beri popüler olan ünlü "böl ve yönet" stratejisini kullanır. Amaç, karmaşık bir görevi –Galya'yı yönetmeyi– daha basit, yerel görevlere bölmektir. Ağaçlar, bilgisayar bilimlerinde genellikle aynı nedenle, çeşitli uygulamalarda karmaşıklığı azaltmak için kullanılır.

Hatırlayacağınız üzere daha önce anafikrin yeni sorguya en benzer komşu öğrenme örneklerinin altkümesini bulmak olduğu parametrik olmayan kestirim hakkında konuşmuştuk. En yakın *k* komşu algoritmalarında bunu, tüm öğrenme verilerini belleğe kaydedip yeni sınama sorgusu ile tüm öğrenme örnekleri arasındaki benzerliği birer birer hesaplayarak ve en benzerleri seçerek yaparız. Öğrenme veri kümesi büyük olduğunda bu oldukça karmaşık bir hesap gerektirir ve eğer veri çoksa pratikte imkânsız olabilir.

Karar ağacı, en benzer öğrenme örneklerini farklı girdi nitelikleri üzerinde yapılan bir dizi sınama yaparak bulur. Ağaç, karar düğümlerinden ve yapraklarından oluşur. Kökten başlayarak her karar düğümü girdiye bir bölme sınaması uygular ve sonuca bağlı olarak dallardan birini seçeriz. Bir yaprağa ulaştığımızda arama durur ve en benzer eğitim örneklerini bulduğumuzu anlayıp onlardan sonucu ara değerleme [*interpolate*] yaparak hesaplarız. (bkz. Şekil 3.2)

Kökten bir yaprağa uzanan her yol, yoldaki karar düğümlerindeki sınama koşullarının VE ile birleşimine karşılık gelir ve böyle bir yol, eğer-ise kuralı biçiminde yazılabilir. Karar ağacının güzel yanlarından biri de budur: Bir ağaç, eğer-ise kurallarından oluşan bir kural tabanına dönüştürülebilir ve bu kuralların yorumlanması kolaydır. Ağaç bir öğrenme kümesi üstünde eğitilir ve bölmeler çıktı bakımından birbirine benzer "saf" bölgeleri ayıracak biçimde veriden öğrenilir.

Karar ağacı eğitimi parametrik değildir – ağaç gerek duyulduğunda büyür ve büyüklüğü verinin altında yatan problemin karma-



Şekil 3.2 Düşük ve yüksek riskli müşterileri ayıran bir karar ağacı. Bu ağaç, Şekil 2.2'de gösterilen ayırtacı gerçekler.

şıklığına bağlıdır; basit bir görev için öğrenilen ağaç küçüktür, zor bir görevde büyük bir ağaç ortaya çıkabilir.

Karar düğümlerinde kullanılan bölme sınamasına ve yapraklarda yapılan ara değerlemeye bağlı olarak farklı karar ağacı modelleri ve öğrenme algoritmaları vardır. Günümüzde son derece yaygın olan yaklaşımlardan biri, öğrenme verisinin rasgele altkümeleriyle birçok karar ağacı eğittiğimiz ve tahminleri üstünden oylama aldığımız *rastsal ormandır*; bu yaklaşım, tahmindeki değişintiyi azaltır ve başarıyı artırır.

Ağaçlar, çeşitli yapay öğrenme uygulamalarında başarıyla kullanılmaktadır ve doğrusal modelle birlikte karar ağacı, daha karmaşık bir öğrenme algoritması denenmeden önce temel kıyaslama yöntemlerinden biri olarak alınmalıdır.

Aktif Öğrenme

Öğrenmede, öğrenenin neyi bildiğini ve neyi bilmediğini bilmesi de çok önemlidir. Eğitilmiş bir model bir tahminde bulunduğunda, tahmindeki kesinliğini de gösterebiliyorsa bu tahmin daha

çok bilgi verir. Daha önce tartıştığımız gibi bu, daha küçük bir aralığın daha az belirsizliği gösterdiği bir güven aralığı şeklinde olabilir.

Genellikle daha fazla veri daha fazla bilgi anlamına gelir ve bu nedenle daha büyük bir veri kümesi kullanmak tahmin belirsizliğini azaltır. Fakat veri noktaları eşit yaratılmamıştır ve eğitilmiş model belirsizliğin nerede yüksek olduğunu bilirse, gözetmenden oradaki örnekleri etiketlemesini aktif olarak isteyebilir. Buna aktif *öğrenme* denir. Model, yeni girdilerin sentezlenmesiyle sorgular üretir ve ders sırasında soru soran bir öğrenci gibi, bunların etiketlenmelerini, yani onlar için çıktının belirtilmesini ister.

Örneğin yapay zekânın epey başlarında, en bilgilendirici örneklerin o an için tahmin edilen ayırtaca en yakın yerde bulunanlar olduğu anlaşıldı: Bir yakın ıskalama [*near miss*], son derece pozitif olarak tahmin ettiğimiz ancak aslında negatif etiketli bir örnektir (Winston 1975).

Yapay öğrenmede aktif öğrenmeyle ilişkili bir araştırma alanına hesaplamalı *öğrenme teorisi* denir. Burada amaç, herhangi bir öğrenme verisinden bağımsız olarak öğrenme algoritmalarının öğrenme yetileri için teorik sınırlar bulmaktır. Örneğin bir model ve öğrenme algoritması için, yeterince yüksek olasılıkla yeterince düşük hatayı garanti etmek için gereken en az öğrenme örneği sayısını bilmek isteyebiliriz. Buna olası yaklaşık doğru öğrenme [*probably approximately correct learning*] denir (Valiant 1984).

Sıralamayı Öğrenme

Sıralama, yapay öğrenmenin regresyon veya sınıflandırmadan farklı ve bir bakıma bunların ikisi arasında duran bir uygulama alanıdır. Sınıflandırma ve regresyonda her örnekle ilgili olarak çıktı için istenen bir mutlak değer vardır. Sıralamada ise örnek çiftleri üzerinde eğitim yaparız ve bunlar için ürettiğimiz çıktıların doğru sırada olması istenir.

Film önerilerinde bulunmak için bir *öneri modeli* öğrenmek istediğimizi varsayalım. Bu görev için girdi, filmin ve müşterinin nite-

liklerinden oluşur. Çıktı, belirli bir müşterinin belirli bir filmden zevk alacağına dair inancımızın ölçüsü olan sayısal bir puandır. Böyle bir modeli eğitmek için geçmiş müşteri değerlendirmelerini kullanırız. Müşterinin geçmişte *A* filmini *B* filminden daha fazla sevdiğini biliyorsak, o müşteri için *A*'nın tahmini puanı *B*'nin puanından daha yüksek bir değerde olacak şekilde eğitimi gerçekleştiririz. Ardından, ilerleyen aşamalarda bu modeli en yüksek puana dayanarak bir öneride bulunması için kullandığımızda, *B*'den çok *A*'ya benzeyen bir film önermesini bekleriz.

Çıktı olan puan için, örneğin kullanılmış bir otomobilin fiyatında olduğu gibi, mutlak bir değer tanımlı değildir. Sıraları doğru olduğu sürece puanlar herhangi bir değerde olabilir. Öğrenme verileri mutlak değerler açısından değil, bu gibi sıralama kısıtları olarak verilir (Liu 2011).

Burada bir sıralayıcının, bir sınıflandırıcı veya bir bağlayıcıya göre üstünlüğünü ve farkını görüyoruz. Eğer kullanıcılar gördükleri filmleri "beğendim ya da beğenmedim" olarak oylarsa, bu iki sınıflı bir sınıflandırma problemi olacaktır ve bir sınıflandırıcı kullanılabilir; ancak beğeni nüanslıdır ve insanları iki değerli bir oylamaya zorlamak kısıtlayıcı olur. Öte yandan, eğer insanlar her bir filmden aldıkları zevki, diyelim ki 1 ile 10 arasında bir değerle gösterirse, bu bir regresyon problemi tanımlar, ancak bu tür değerlerin de atanması zordur. İnsanlar açısından izledikleri iki filmi karşılaştırıp hangisinden daha çok zevk aldıklarını söylemek daha doğal ve kolaydır. Sıralayıcı tüm bu çiftlerle eğitildikten sonra, onun tüm bu sıra kısıtlamalarını karşılayan sayısal puanlar üretmesi beklenir.

Sıralamanın birçok uygulaması vardır. Arama motorlarında bir sorguyla en ilgili belgeleri ekranda görmek isteriz. Arama motoru, ilk on adayı seçip gösterdiğinde, kullanıcı ilk iki öneriyi atlayıp üçüncüyü tıklarsa, üçüncü kaydın bir sonraki seferde birinci ve ikinci kayıtlardan daha üst sıraya konması gerektiğini anlarız. Bu tür tıklama günlükleri sıralayıcıları eğitmek için kullanılır.

Bayes'çi Yöntemler

Bazı uygulamalarda ve bazı modellerde parametrelerin olası değerleri hakkında bazı önsel inançlarımız [*prior belief*] olabilir. Yazı tura attığımızda bunun hilesiz bir atış olmasını ve bu nedenle tura olasılığının ½'ye yakın olmasını bekleriz. Bir otomobilin fiyatını kestirirken kilometrenin fiyat üzerinde negatif bir etki yapmasını bekleriz. Bayes'çi yöntemler parametreleri tahmin ederken bu gibi önsel inançları dikkate almamızı sağlar.*

Bayes'çi kestirimdeki düşünce, bu önsel bilgiyi verilerle birlikte parametrelerin sonsal [*posterior*] dağılımını hesaplamak için kullanmaktır. Bayes'çi yaklaşım, özellikle veri kümesi küçük olduğunda kullanışlıdır. Sonsal dağılım elde etmenin bir güzel yanı, her bir parametre değerinin ne kadar olası olduğunu bize söylemesidir, bu nedenle istersek en olası değeri seçebiliriz; buna en büyük sonsal (EBS) [*maximum a posteriori* (MAP)] kestirim denir. Ya da tek bir değer yerine parametrenin tüm değerleri ya da yüksek olasılığa sahip olanlar üstünden ortalama alırız ve böylece parametrenin kestirilmesindeki belirsizliği de azaltmış oluruz.

Bayes'çi yaklaşım esnek ve ilginç olsa da kısıtlayıcı varsayımlar altındaki basit senaryolar dışında gerekli hesaplama çok karmaşık olabilir. Bir yol, gerekli hesaplaması kolay olmayan bir gerçek sonsal dağılım yerine, kolaylar içinden en benzer olanı kullandığımız yaklaşıklamadır [*approximation*]. Başka bir yol, dağılımın kendisini kullanmak yerine dağılımdan temsilî örnekler ürettiğimiz ve onlara dayanarak çıkarım yaptığımız örneklemedir [*sampling*]. Bu iki yol için yaygın kullanılan yöntemler, yani ilki için değişimsel yaklaşıklama [*variational approximation*] ve ikincisi için *Markov zinciri Monte Carlo* (MZMC) *örneklemesi*, yapay öğrenmede en önemli güncel araştırma yönelimleri arasındadır.

* Bayes kestirim, olasılık teorisindeki (daha evvel gördüğümüz) Bayes kuralını kullanır ve bu ismi Presbiteryen vaizi olan Thomas Bayes'ten (1702-1761) almıştır. Gözlenebilir veriden önce gelen ve altında yatan bir önsel varsayımı, Bayes'in yaptığı işin sanki doğal bir sonucudur.

Bayes'çi yaklaşım, önsel inançlarımızı eğitime dahil etmemizi sağlar. Örneğin, verinin altında yatan problemin düzenli olduğuna olan inancımız daha basit modelleri tercih etmemize yol açar. *Düzenlileştirmede* karmaşıklığı cezalandırırız ve öğrenme sırasında bir yandan veriler için en başarılı çıktı üretmeye çalışırken aynı anda kullandığımız modelin karmaşıklığını da en aza indirmeye çalışırız. Modeli gereksiz yere karmaşık ve çıktıyı değişken yapan parametrelerden kurtulmak isteriz. Bu sadece parametrelerin ayarlanmasını değil, aynı zamanda model yapısındaki değişiklikleri de içeren bir öğrenme yaklaşımı anlamına gelir. Ya da elimizdekinin aşırı basit bir model olduğundan şüphelendiğimizde diğer yöne gidip karmaşıklığı yavaş yavaş arttırabiliriz.

Bu tür *parametrik olmayan* yaklaşımların Bayes'çi kestirimde kullanımı özellikle ilgi çekicidir çünkü artık bazı parametrik model sınıfları tarafından kısıtlanmayız; model karmaşıklığı da verinin altında yatan problemin karmaşıklığına uymak için dinamik olarak değişebilir (Orbanz ve Teh 2010). Bu "sonsuz boyutlu" bir model demektir, çünkü veriye bağlı olarak model istediğimiz kadar karmaşık olabilir ve öğrendikçe büyür.

4

YAPAY SİNİR AĞLARI VE DERİN ÖĞRENME

Yapay Sinir Ağları

Bizi zeki yapan beynimizdir; beynimiz sayesinde görür, duyar, öğrenir, hatırlar, planlar ve eylem yaparız. Bundan dolayı bu yeteneklere sahip makineler gerçeklemeye çalışırken ilk ilham kaynağımız insan beynidir, tıpkı ilk uçma girişimlerimize kuşların ilham kaynağı olması gibi. Yapmak istediğimiz, beynin nasıl işlediğine bakmak ve yaptığı şeyi nasıl yaptığı hakkında bir kavrayış kazanmaya çalışmaktır. Bunu yaparken gerçekleme ayrıntılarından bağımsız bir açıklamaya sahip olmak isteriz – buna birinci bölümde analiz düzeyleri hakkında konuşurken hesaplama teorisi adını vermiştik. Böyle bir soyut, matematiksel ve hesaplamalı açıklama çıkarabilirsek daha sonra mühendisler olarak bunu elimizde olanlarla, örneğin silikon üstünde ve elektrikle çalışacak şekilde gerçekleyebiliriz.

Uçan makineler yapmaya yönelik erken girişimler, biz aerodinamik teorisini anlayana kadar başarısız oldu; ancak bu teoriyi kavradıktan sonra uçak yapabildik. Günümüzde kuşları ve uçakları iki

farklı uçma biçimi olarak görüyoruz. Onlara artık yapay kuş değil, uçak diyoruz ve uçaklar kuşlardan daha fazlasını yapabiliyor. Daha uzun mesafeler kat ediyor, yolcu ya da kargo taşıyabiliyorlar. Buradaki düşünce, aynı şeyi zekâ için başarmaktır ve bunu yapmaya beyinden ilham alarak başlıyoruz.

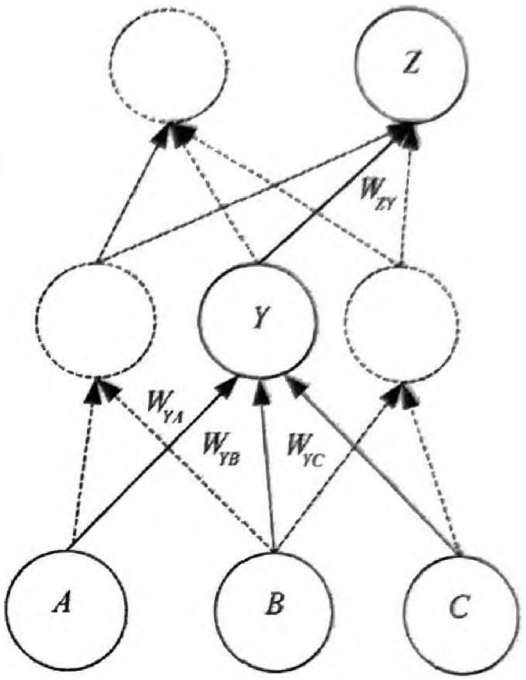
İnsan beyni *nöron* adı verilen çok sayıda işlem biriminden oluşur ve her nöron *sinaps* adı verilen bağlantılarla çok sayıdaki başka nörona bağlanır. Nöronlar aynı anda, paralel olarak çalışır ve sinapslar üzerinden birbirlerine bilgi aktarırlar. Hesaplamanın nöronlarca yapıldığına ve belleğin sinapslarda, yani nöronların birbirine bağlanma ve birbirini etkileme biçiminde tutulduğuna inanılıyor.

Analog hesaplama için –nöron çıktıları 0 ya da 1 değildir– bir model olarak *sinir ağları* üzerine yapılan araştırmalar, sayısal hesaplama araştırmaları kadar erken başladı (McCulloch ve Pitts 1943) ancak sayısal bilgisayarın hızlı başarısının ve yaygın kullanımının ardından uzun süre dikkatlerden uzak kaldı.

1960'larda algılayıcı [*perceptron*] ağ modeli örüntü tanıma için bir model olarak öne sürüldü (Rosenblatt 1962). Bu ağ, her nöronun bir etkilenim [*activation*] değerine sahip olduğu yapay nöronlardan ve aralarındaki sinaptik bağlantılardan oluşur; nöron *A*'dan nöron *B*'ye giden bağlantı, *A*'nın *B* üzerindeki etkisini belirleyen bir ağırlığa sahiptir. Eğer sinaps uyarıcı ise, *A* etkin olduğunda *B*'yi de etkinleştirmeye çalışır; eğer sinaps önleyici ise, *A* etkin olduğunda *B*'yi baskılamaya çalışır.

Ağın çalışması sırasında her nöron, kendisine bir sinaps oluşturan tüm nöronların etkilenimlerini sinaptik ağırlıklarıyla ağırlıklandırarak toplar ve eğer toplam belli bir eşik değerinden büyükse nöron "ateşlenir" ve çıktısı bu nöronun etkilenim değerine karşılık gelir; aksi durumda nöron sessiz kalır. Nöron ateşlenirse etkilenim değerini üstünde sinaps yaptığı bütün nöronlara gönderir (bkz. Şekil 4.1).

Algılayıcı, karar vermeden önce ağırlıklı bir toplam hesaplar ve bu daha önce ele aldığımız doğrusal modeli gerçeklemenin bir yolu olarak görülebilir. *Çok katmanlı algılayıcıda her katmandaki nöronlar* önceki katmandaki tüm nöronlardan girdi alıp kendi değerlerini pa-



Şekil 4.1 Nöron ve aralarındaki sinapslardan oluşan örnek bir sinir ağı. Nöron Y, girdisini A, B ve C nöronlarından alır. A'nın Y'ye etkisini bağlantı ağırlığı W_{YA} belirler. Y kendisine bağlı nöronların değerlerini bağlantı ağırlıklarıyla çarpıp toplar ve toplam yeterince büyükse "ateşlenir" ve kendi değerini başka nöronlara, örneğin Z'ye gönderir.

ralel olarak hesaplarlar ve bu değerler hep birlikte sonraki katmandaki nöronlara aktarılır.

Nöronların bazıları duyusal girdi nöronlarıdır ve değerlerini retinadaki reseptörlere benzer şekilde çevreden (örneğin algılanan görüntüden) alırlar. Bu değerler, nöron etkilenimleri ağ üzerinde yayıldıkça ardışık katmanlarda, üstlerinde biraz daha işlem yapan başka nöronlara aktarılır. Ağın en son katmanında, eyleyiciler [*actuator*] aracılığıyla bir kolu hareket ettirmek, bir sözcük söylemek vb. gibi eylemleri gerçekleştiren çıktı nöronları vardır.

Sinir Ağı Öğrenme Algoritmaları

Bir sinir ağında öğrenme algoritmaları nöronlar arasındaki bağlantı ağırlıklarını ayarlar. İlk algoritmalardan biri Hebb (1949) tarafından ileri sürülmüştür ve *Hebb'ci öğrenme* kuralı olarak bilinir: Eğer iki nöron aynı anda etkin ise ikisi arasındaki ağırlık artar – sinaptik ağırlık iki nöron arasındaki ilintiyi öğrenir.

Diyelim ki görsel alanda bir daire olup olmadığını kontrol eden bir nöronumuz ve görsel alanda "6" rakamının olup olmadığını kontrol eden bir başka nöronumuz var. Ne zaman bir 6 görsek –ya da okumayı öğrenirken bize onun altı olduğu söylense– bir daire de görürüz, bu yüzden bu iki nöron arasındaki bağlantı gittikçe güçlenir; ancak daire nöronu ile diyelim ki "7" rakamı nöronu arasındaki bağlantı güçlendirilmez, çünkü birini gördüğümüzde ötekini görmeyiz. Böylece bir dahaki sefere görme alanında bir daire gördüğümüzde, bu 6 rakamı nöronunun etkilenimini artıracak ve 6'yı 7'den daha olası bir hipotez yaparak 7 rakamı nöronunun etkilenimini azaltacaktır.

Bazı uygulamalarda ağdaki belli bazı nöronlar girdi birimleri, bazıları da çıktı birimleri olarak tanımlanmıştır. Böyle bir uygulamada, bir girdi ile buna karşılık gelen (bir gözetmen tarafından belirlenmiş) doğru çıktı çiftlerini içeren bir öğrenme kümemiz vardır – örneğin kullanılmış bir otomobilin fiyatını tahmin etmede girdi olarak otomobil niteliklerine ve çıktı olarak fiyatlarına sahibizdir. Bu gözetimli öğrenme uygulamasında, girdi birimlerine öğrenme kümesindeki girdi değerlerini verir, ağırlıklara ve ağ yapısına bağlı olarak etkinliğin ağ üzerinden yayılmasına izin veririz ve en sondaki çıktı birimlerinde hesaplanan değerlere bakarız.

Ağın bir girdi için tahmin ettiği gerçek çıktılar ile onların gözetmen tarafından öğrenme kümesinde belirlenmiş istenen değerleri arasındaki farkların toplamı bir *hata fonksiyonu* tanımlar. Sinir ağı eğitiminde her öğrenme örneği için bağlantı ağırlıklarını bu örnek için yapılan çıktı hatasını azaltacak yönde hafifçe güncelleriz. Hatayı azaltmak, aynı veya benzer girdiyi bir daha gördüğümüzde, ağın hesapladığı çıktının doğru değere biraz daha yakın olacağı anlamına gelir. *Teorik olarak* bu, ikinci bölümde ele aldığımız regresyondan başka bir şey değildir; tek fark, burada modelin nöronlar ve bağlantılardan oluşan bir sinir ağı olarak *gerçeklenmesidir.*

Bu, sinir ağı öğrenme algoritmalarının önemli bir özelliğidir; yani bu algoritmalar, öğrenme örneklerini birer birer gördükçe bağlantı ağırlıklarında küçük güncellemeler yaparak *çevrimiçi* öğrenebilirler. Toptan [*batch*] öğrenmede tüm öğrenme verisine baktıktan

sonra bir kerede güncelleme yaparız. Günümüzde yaygın olan bir yaklaşım, her güncellemede küçük örnek kümeleri kullandığımız *mini yığınlar* kullanmaktır.

Bugünlerde veri kümelerinin büyümesiyle çevrimiçi öğrenme çekici hale gelmiştir, çünkü bu yöntem tüm verilerin toplanmasını ve saklanmasını gerektirmez. Bir akan veri senaryosunda, bir defada yalnızca bir veya birkaç örnek kullanarak öğrenebiliriz. Dahası, verinin temel özellikleri yavaşça değişirse, ki bu çok sık rastlanan bir durumdur, çevrimiçi öğrenme durup yeni veri toplamaya gerek duymadan sorunsuzca yeni ortama uyum sağlayabilir.

Bir Algılayıcı Neyi Yapıp Neyi Yapamaz

Algılayıcı birçok işte başarılı olsa da –doğrusal modelin birçok alanda oldukça iyi çalıştığını unutmayalım– bir algılayıcının hiçbir zaman gerçekleyemeyeceği işler de vardır (Minsky ve Papert, 1969). Bunlardan en ünlüsü, dışlayıcı VEYA (XOR) problemidir.

Mantıkta iki tür VEYA vardır: kapsayıcı VEYA ve dışlayıcı VEYA. Gündelik konuşmada, "Havaalanına gitmek için otobüse veya trene bineceğim," dediğimizde, kastettiğimiz dışlayıcı VEYA'dır. Burada iki seçenek vardır ve bir seferde bunlardan yalnızca biri doğrudur. Kapsayıcı VEYA'yı belirtmek için "ve/veya" yapısını kullanırız, örneğin, "Bu güz döneminde Matematik 101 ve/veya Fizik 101 dersini alacağım," yani, Matematik 101, Fizik 101 veya her ikisini birden alacağım.

Kapsayıcı VEYA bir algılayıcı tarafından gerçeklenebilse de dışlayıcı VEYA gerçeklenemez. Bunun nedenini görmek zor değil. Örneğin otobüs ve tren gibi iki girdiniz varsa ve bunlardan birinin yeterli olmasını istiyorsanız, nöronun ateşlenmesi için her birine eşikten daha büyük bir ağırlık vermeniz gerekir ki bunlardan herhangi biri doğru olduğunda nöron ateşlenebilsin. Fakat her ikisi de doğru olduğunda, toplam etkilenim iki kat daha yüksek olacaktır ve bu da eşikten daha az olamaz.

İlk başlardan beri dışlayıcı VEYA'nın birden fazla algılayıcı kat-

manı kullanılarak gerçeklenebileceği bilinmesine rağmen, bu tür ağların nasıl eğitileceği bilinmiyordu ve algılayıcının birkaç (sayısal) mantık kapısı tarafından kolayca uygulanabilen dışlayıcı VEYA kadar basit bir görevi bile yerine getirememesi, dünyadaki birkaç yer dışında hayal kırıklığına ve sinir ağı araştırmalarının uzun süre terk edilmesine neden oldu. Çok katmanlı algılayıcıların eğitimi için hata geri yayma [*backpropagation*] algoritması ancak 1980'lerin ortasında öne sürüldü –fikir 1960'lardan ve 1970'lerden bu yana biliniyordu ancak yaygın kabul görmemişti– ve böylece algılayıcıya olan ilgi tekrar canlandı (Rumelhart, Hinton ve Williams 1986).

Yapay sinir ağlarının tümü ileriye doğru bağlı [*feedforward*] değildir; katmanlar arasındaki bağlantılara ek olarak, nöronların (kendileri dahil) aynı katmandaki nöronlarla ve hatta önceki katmanlarda bulunan nöronlarla da bağlantı kurduğu özyineli [*recurrent*] ağlar da vardır. Her bir sinaps belirli bir gecikmeye neden olur, böylece tekrarlayan bağlantılar yoluyla nöron etkilenimleri bağlamsal bilgi için *kısa süreli bellek* görevi görür ve ağın geçmişi hatırlamasını sağlar.

Diyelim ki girdi nöronu *A* nöron *X*'e bağlı ve ayrıca *X*'ten kendisine özyineli bir bağlantı var. Bu nedenle *X*'in *t* zamanındaki değeri, *A'nın t* zamanındaki değerine ve *X*'ten kendisine tekrarlayan (ve bir birim zaman gecikmesine neden olan) bağlantı nedeniyle, *X*'in *t*-1 zamanındaki değerine bağlı olacaktır. Bir sonraki zaman adımında, *X*'in *t*+1 zamandaki *değeri*, *A'nın t*+1 zamanındaki değerine ve *X*'in *t* zamanındaki *değerine* bağlı olacaktır (ki bu daha önce *t* zamanındaki *A* ve *t*-1 zamanındaki *X* kullanılarak hesaplanmıştı) vs. Bu şekilde, *X*'in herhangi bir zamandaki değeri, o zamana kadar görülen tüm girdilere bağlı hale gelmiş olur.

Bir ağın durumunu belirli bir zamanda tüm nöronların değerlerinin birleşimi olarak tanımlarsak özyineli bağlantılar o anki durumun yalnızca o anki girdiye değil, ağda önceki zaman adımlarındaki girdilerden hesaplanan nöronlara da bağlı olmasına izin verir. Örneğin, bir cümleyi her defasında bir sözcüğü girdi alır şekilde görüyorsak, özyineleme cümle içindeki önceki sözcüklerin sıkıştırılmış ve soyut bir gösterimde tutulmasını ve böylece o anki sözcüğü işlerken dikkate alınmasını sağlar. Ağın mimarisi ve özyineli bağlantıların

ayarlanma şekli, geçmişin o anki çıktıyı ne kadar geriden ve nasıl etkilediğini tanımlar.

Özyineli sinir ağları, zaman boyutunun önemli olduğu konuşma veya dil işlemede kullanılır ve bu uygulamalarda tanımak istediğimiz şey dizilerdir. Bir dilden diğerine yapılan bir metin çevirisinde sadece girdi değil, çıktı da bir dizidir.

Bilişsel Bilimde Bağlantısal Modeller

Yapay sinir ağı modelleri bilişsel psikoloji ve bilişsel bilimde bağlantısal [*connectionist*] veya *paralel dağıtık işleme* (PDİ) modelleri olarak bilinmektedir (Feldman ve Ballard 1982; Rumelhart ve McClelland ve PDI Araştırma Grubu 1986). Buradaki düşünce, nöronların kavramlara karşılık geldiği ve bir nöronun etkilenim değerinin o kavramın doğruluğu hakkındaki inancımıza karşılık geldiğidir. Bağlantılar, kavramlar arasındaki kısıtlamalara veya bağımlılıklara karşılık gelir. Bir bağlantı, iki kavram eşzamanlı ortaya çıkıyorsa –örneğin daire ve "6" nöronları arasında– pozitif bir ağırlığa sahiptir ve uyarıcıdır; kavramlar birbirini dışlıyorsa –örneğin daire ve "7" nöronları arasında– negatif bir ağırlığa sahiptir ve önleyicidir.

Değerleri gözlemlenen –çevreyi algılayan– nöronlar bağlı oldukları nöronları etkiler ve bu etkilenim ağ üstünde yayılırken sıradaki nöronlar, bağlantıların tanımladığı kısıtlamaları sağlayacak biçimde etkilenim değeri alırlar.

Bağlantısal modellerdeki temel düşünce, zekânın beliren [*emergent*] bir özellik olduğu ve tanıma ya da örüntüler arasında ilişkilendirme gibi üst düzey görevlerin, etkilenimin birbirine bağlı basit işlem birimlerinin temel işlemleri sonucu yayılmasıyla otomatik olarak ortaya çıktığıdır. Benzer şekilde öğrenme, bağlantı düzeyinde, örneğin Hebb'ci kurala göre basit işlemlerle, üst düzey bir programcıya gerek duyulmadan gerçeklenir.

Bağlantısal ağların biyolojik açıdan akla yakın olması önemlidir ama yine de bunların beynin soyut modelleri olduğu akıldan çıkarılmamalıdır; örneğin beyindeki her kavram için bir nöron olması pek

olası değildir; bu, beynimde sadece büyükannemi gördüğümde veya düşündüğümde etkin hale gelen özel bir nöron olduğunu savunan *büyükanne hücre* teorisidir. Bu, *yerel gösterim* demektir. Beyinde nöronların öldüğü ve yeni nöronların doğduğu bilinmektedir; bu nedenle kavramların, temel nöron yapısındaki fiziksel değişikliklere rağmen hayatta kalabilmelerine yetecek fazlalıkla, birlikte çalışan bir nöron kümesi üzerinde *dağıtık bir gösterime* sahip olduklarına inanmak daha mantıklıdır.

Bir Paralel İşleme Paradigması Olarak Sinir Ağları

1980'lerden bu yana binlerce işlemciye sahip bilgisayar sistemleri üretilebilmektedir. Ne var ki böyle paralel mimarilerin yazılımları, donanımları kadar hızlı gelişmemiştir. Bunun nedeni, o zamana dek hemen hemen tüm hesaplama teorimizin seri çalışan, tek işlemcili bilgisayarlara dayanmasıydı. Paralel bilgisayarları tam kapasitelerinde kullanamıyoruz çünkü verimli bir şekilde programlayamıyoruz.

Paralel hesaplama için temelde iki paradigma vardır. Tek komut, çok veri (TKÇV) [*single instruction, multiple data* (SIMD)] bilgisayarlarda tüm işlemcilerin programı aynıdır ancak bu programı farklı veri parçaları üzerinde aynı anda çalıştırırlar. Çok komut, çok veri (ÇKÇV) [*multiple instruction, multiple data* (MIMD)] bilgisayarlarda, farklı işlemciler farklı veriler üzerinde farklı programlar çalıştırabilirler. TKÇV bilgisayarların programlanması daha kolaydır çünkü yazılacak tek bir program vardır. Bununla birlikte, problemler bir TKÇV bilgisayarda paralelleştirilebilmeye uygun düzenli bir yapıya nadiren sahiptir. ÇKÇV bilgisayarları daha geneldir ancak tüm işlemciler için ayrı program yazmak kolay bir iş değildir; senkronizasyon, işlemciler arasında veri transferi vb. ile ilgili ek problemler ortaya çıkabilir. TKÇV bilgisayarlarının yapımı da daha kolaydır ve TKÇV yapısıyla daha fazla işlemciye sahip bilgisayarlar yapılabilir. ÇKÇV yapısında işlemciler daha karmaşıktır ve işlemcilerin istedikleri gibi veri alışverişi yapabilmesi daha karmaşık bir iletişim yapısı gerektirir.

Şimdi, işlemcileri TKÇV işlemcilerinden biraz daha karmaşık ama ÇKÇV işlemcileri kadar da karmaşık olmayan bir bilgisayar düşünelim. Bazı parametrelerin saklanabileceği az miktarda yerel belleğe sahip basit işlemcilerimiz olsun. Her işlemci, TKÇV yapısında olduğu gibi sabit bir fonksiyon uygular; ancak yerel belleğine farklı değerler yüklendiğinde her işlemci farklı şeyler yapabilir ve bu sayede asıl iş bu işlemcilere dağıtılabilir. Böylece nöral komut, çok veri (NKÇV) [*neural instruction, multiple data* (NIMD)] bilgisayarları diyebileceğimiz yapıya sahip oluruz ki burada her işlemci bir nörona, yerel parametrelerse onun sinaptik ağırlıklarına karşılık gelir ve tüm yapı bir sinir ağı oluşturur. Her bir işlemcide uygulanan fonksiyon basit ve yerel bellek küçükse, bu tür çok sayıda işlemci tek bir yonga üzerine yerleştirilebilir.

Şimdi problem, bu tür işlemcilerden oluşan bir ağa asıl işi dağıtmak ve buna uygun yerel parametre değerlerini belirlemektir. Öğrenmenin devreye girdiği yer de burasıdır: Bu bilgisayarlar eğer örneklerden öğrenebilirse, onları programlamamız ve parametre değerlerini bizim belirlememiz gerekmez.

Bu sayede yapay sinir ağları, günümüz teknolojisiyle üretebildiğimiz paralel donanımı verimli bir biçimde kullanmanın bir yoludur ve –öğrenme sayesinde– programlanmaları da gerekmez.

Çoklu Katmanlarda Hiyerarşik Gösterimler

Daha önce tek bir algılayıcı katmanının dışlayıcı VEYA gibi bazı işleri gerçekleyemeyeceğinden ve bu sınırlamaların birden fazla katman olduğunda geçerli olmadığından bahsetmiştik. Aslında çok katmanlı algılayıcının evrensel *bir* yaklaşıklayıcı [*approximator*] olduğu kanıtlanmıştır, yani yeterince nöron kullandığımızda, herhangi bir fonksiyonu istediğimiz doğrulukla yaklaşıklayabilir – her ne kadar bunu yapmak her zaman çok kolay olmasa da.

Algılayıcı algoritması yalnızca tek katmanlı ağları eğitebilir, ancak 1980'lerde çok katmanlı algılayıcıları eğitmek için hata *geri yayma algoritması* icat edilince bu, çok farklı alanlardaki uygulamaların

birdenbire çoğalmasını sağlayarak sinir ağları üstüne araştırmaları, bilişsel bilimden bilgisayar bilimleri ve mühendisliğine kadar birçok alanda önemli ölçüde hızlandırdı.

Çok katmanlı ağ akla yakındır; çünkü ham girdiden başladığımız ve soyut bir çıktı gösterimi elde edene kadar aşamalı olarak her düzeyde daha karmaşık bir dönüşüm gerçekleştirdiğimiz işlem katmanlarına karşılık gelir.

Örneğin görüntü tanımada, ilk katmanın girdisi görüntü piksellerimizdir. Bir sonraki katmandaki nöronlar bunları birleştirerek farklı açılarda çizgi parçaları ve kenarlar gibi temel görüntü öğelerini bulur. Daha sonraki bir katman bunları daha uzun çizgiler, yaylar ve köşeler oluşturmak için birleştirir. İzleyen katmanlar bunları birleştirerek daireler, kareler vb. daha karmaşık şekilleri tanır. Bunlar da öğrenmek istediğimiz nesneleri, örneğin yüzler veya elle yazılmış karakterleri tanımak için ek işleme katmanlarıyla birleştirilir.

Bir katmandaki her bir nöron, altındaki katmanda bulunan daha basit özellikleri kullanarak daha karmaşık bir özelliği tanımlar. Bu ara özellik tanıyan birimlere *saklı birimler* denir, çünkü bunlar doğrudan gözlenmeyen ancak gözlemlenen unsurlarca tanımlanmış saklı niteliklere karşılık gelir. Bu ardışık saklı birim katmanları, artan soyutlama katmanları olarak yorumlanabilir. İlk katmanda pikseller gibi ham verilerden başlarız ve son katmanda rakam veya yüz gibi soyut kavramlara ulaşırız.

İlginçtir ki benzer bir mekanizmanın görme korteksinde de olduğu düşünülmektedir. Görsel nörofizyoloji alanındaki çalışmalarıyla daha sonra 1981 Nobel Ödülü'nü alan Hubel ve Wiesel, kediler üzerinde yaptıkları deneylerde, görsel alanda belirli konumlardaki belirli açılardaki çizgilere tepki veren *basit hücreler* olduğunu ve bunların daha karmaşık şekilleri tanımak üzere *karmaşık* ve *hiper karmaşık* hücrelere beslendiğini göstermiştir (Hubel 1995) – henüz daha sonraki katmanlarda neyin olup bittiği çok iyi bilinmemektedir.

Ağı böyle yapılandırmak, girdiyle ilgili bağımlılık varsayımlarında bulunmak anlamına gelir. Örneğin, görüntüde birbirine yakın

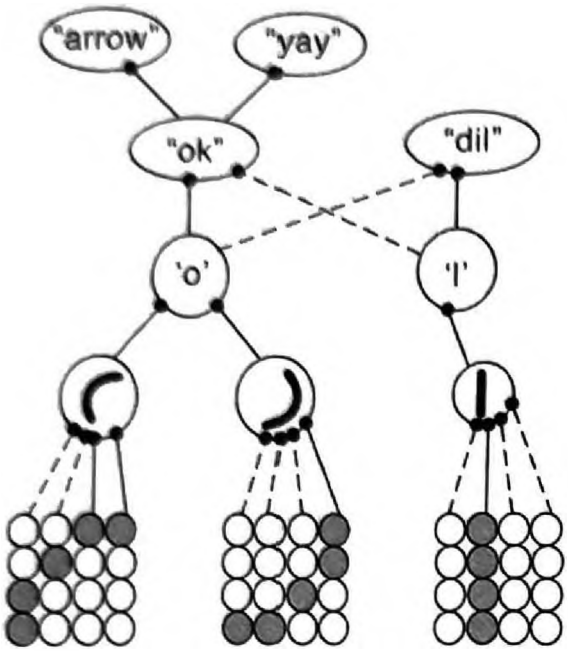
piksellerin ilintili olduğunu ve kenarlar ile köşeler gibi yerel özellikler bulunduğunu biliyoruz. Elle yazılmış bir rakam gibi herhangi bir nesne, bu temel öğelerin bir bileşimi olarak tanımlanabilir. Görsel sahne düzgün bir şekilde değiştiğinden, yakındaki piksellerin aynı nesneye ait olma eğiliminde olduğunu ve piksel değerlerinde ani değişim olan bir noktanın –bir kenarın– nadir rastlandığı için bilgilendirici olduğunu biliyoruz.

Benzer şekilde, konuşmada yerellik zaman boyutundadır ve zamanda birbirine yakın girdiler konuşmanın temel öğeleri olarak gruplandırılabilir. Bu temel öğeler birleştirilerek daha uzun ifadeler, yani ses fonemleri tanımlanabilir. Bunlar da sözcükleri tanımlamak ve ardından cümleleri oluşturmak için birleştirilebilir.

Bu gibi durumlarda katmanlar arasındaki bağlantıları tasarlarken, birimler tüm girdi birimlerine bağlanmaz, çünkü girdilerin hepsi ilintili değildir. Bunun yerine, girdi alanında bir pencere tanımlayan ve girdilerin yalnızca küçük bir *yerel* altkümesine bağlı olan birimler belirleriz. Bu, bağlantı sayısını ve dolayısıyla öğrenilecek parametre sayısını azaltır. *Evrişimli sinir ağı* denen böyle bir yapıda, her bir birim, girdiyle ağırlıkların evrişimini [*convolution*] –yani ne kadar eşleştiğini– hesaplar (Le Cun ve diğerleri 1989). Erken tarihli benzer bir mimari *neocognitron*'dur (Fukushima 1980).

Buradaki düşünce bunu, her bir katmanın aşağıdaki az sayıda yerel birime bağlı olduğu ardışık katmanlarda tekrarlamaktır. Her bir *özellik çıkarıcı* katman, girdi uzamının biraz daha büyük bir parçasındaki özellikleri bir araya getirerek biraz daha karmaşık özellikleri denetler, ta ki tüm girdiye baktığımız çıktı katmanına ulaşana kadar. Özellik çıkarma, *boyut sayısının azaltılmasını* da sağlar, çünkü gözlemlediğimiz ham nitelikler sayıca fazla olsa da, verilerden çıkardığımız ve çıktıyı hesaplamak için kullandığımız saklı özellikler genellikle çok daha azdır.

Bu çok katmanlı ağ bir *hiyerarşik koni* örneğidir. Bu yapıda özellikler, yukarı çıktıkça daha karmaşık, soyut ve sayıca daha az hale gelir (bkz. Şekil 4.2).



Şekil 4.2 Hiyerarşik işlemenin oldukça basitleştirilmiş bir örneği. En alt düzeydeki nöronlar piksellere karşılık gelir ve yay veya çizgi parçası gibi temel öğeleri tanımak için birleştirilirler. Bir sonraki katman bunları birleştirerek harfleri, sonraki katman bunları birleştirip sözcükleri tanır. Yukarı çıktıkça nöronlar daha soyut kavramlara karşılık gelir. Sürekli çizgiyle gösterilen bağlantılar pozitif (uyarıcı), kesik olanlar da negatiftir (engelleyici): "o" harfi "ok" sözcüğünde vardır ama "dil" sözcüğünde yoktur. Daha yüksek düzeylerde ağ üstünde etkilenim daha soyut bağlantılar üstünden yayılır; örneğin "ok" ve "yay" sözcükleri anlamsal olarak ilintilidir; "arrow" sözcüğü de "ok"un İngilizce karşılığıdır.

Özel bir tür çok katmanlı ağ yapısı, istenen çıktının girdiye eşit olacak şekilde tanımlandığı özkodlayıcıdır [*autoencoder*]. Bu ağın ara katmanında girdide olduğundan çok daha az saklı birim vardır ve böylece girdiden saklı katmana kadar olan ilk bölüm, yüksek boyutlu girdinin sıkıştırılıp daha az saklı birimin etkilenim değerleriyle temsil edildiği bir kodlama aşamasıdır. Saklı katmandan çıktıya uzanan ikinci bölüm ise, saklı katmandaki düşük boyutlu gösterimden yüksek boyutlu girdiyi çıktıda yeniden oluşturan bir kod çözüm aşamasına karşılık gelir.

Ağın girdiyi çıktı birimlerinde yeniden yapılandırabilmesinde az sayıdaki saklı birim bir darboğaz işlevi görür ve çıktıda girdiyi yeniden oluşturabilmek için bilgiyi en üst düzeyde koruyan en iyi özellikleri çıkarabilmelidir. Özkodlayıcı gözetimsizdir; saklı birimler girdinin iyi bir kodlamasını ve kısa bir sıkıştırılmış gösterimini

bulmayı, en önemli özellikleri çıkarırken ilgisiz, yani gürültü yüzünden eklenmiş unsurları dikkate almamayı öğrenirler.

Derin Öğrenme

Son yarım yüzyılda bilgisayarla görüntü tanıma alanında doğru sınıflandırma için en iyi özellikleri bulmak amacıyla önemli araştırmalar yapılmış ve elle tanımlanmış özellik çıkarıcıları olarak birçok farklı görüntü filtresi, dönüşümü ve evrişimi önerilmiştir.

Bu yaklaşımlar bir miktar başarı kaydetmiş olsa da öğrenme algoritmaları son zamanlarda büyük veri ve güçlü bilgisayarlarla daha yüksek doğruluk elde etmektedir. Az sayıda varsayım ve az insan katkısıyla, hiyerarşik koniye benzer yapılar büyük miktarlarda veriden otomatik olarak öğrenilir. Bu öğrenme yaklaşımları özellikle ilginçtir, çünkü öğrendikleri için belirli bir göreve yönelik olarak tanımlanmazlar ve farklı uygulamalarda kullanılabilirler. Bu ağlar, hem saklı özellik çıkarıcılarını hem de bunların çıktıyı tanımlamak için en iyi nasıl birleştirildiğini öğrenirler.

Ham girdiden başlayarak her saklı katmanın önceki katmanda bulunan değerleri birleştirerek girdinin daha karmaşık işlevlerini öğrendiği *derin sinir ağlarının* arkasındaki fikir budur. Gizli birim değerlerinin 0 veya 1 değil de sürekli olması, benzer girdilerin daha hassas ve dereceli bir şekilde gösterilmesine izin verir. Ardışık katmanlar, çıktıların en soyut kavramlar cinsinden öğrenildiği son katmana ulaşana kadar gittikçe daha soyut gösterimlere karşılık gelir.

Bunun bir örneğini, piksellerden başlayarak bir rakam elde edene kadar önce kenarlar, sonra köşeler ve diğer elde ettiğimiz evrişimli sinir ağında gördük. Böyle bir ağda, bağlantıyı ve genel mimariyi tanımlamak için biraz kullanıcı bilgisi gereklidir. Girdilerin görüntü pikselleri olduğu bir yüz tanıyıcı ağ düşünün. Her saklı birim *tüm* piksellere bağlı olsa, ağ girdisinin yüz görüntüsü ve hatta girdinin iki boyutlu olduğu hakkında bile bilgimiz olmaz; girdi yalnızca bir değerler kümesidir. Saklı birimlerin yerelleştirilmiş iki boyutlu yerel alanlarla beslendiği evrişimli bir ağın kullanılması, bu konum bilgi-

sini doğru soyutlamaların öğrenilebilmesini sağlayacak biçimde beslemenin bir yoludur.

Derin öğrenmedeki temel anafikir, artan soyutlama özelliği düzeylerinin en az insan katkısıyla öğrenilmesidir (Schmidhuber 2015; LeCun, Bengio ve Hinton 2015). Bunun nedeni çoğu uygulamada yukarı çıktıkça ve karşılık gelen kavramlar "saklı" hale geldikçe, girdide nasıl bir yapının olduğunu bilememizdir. Bu nedenle, herhangi bir bağımlılık, büyük bir örneklem üstünde yapılan eğitimle otomatik olarak keşfedilmelidir. Soyutlamayı ve genel tanımları öğrenmeyi sağlayan şey işte bu, yani verilerden bu şekilde saklı bağımlılıkların veya örüntülerin veya düzenliliklerin çıkarılmasıdır.

Birden fazla gizli katmana sahip bir ağı eğitmek zor ve yavaştır, çünkü çıktıdaki hatanın önceki tüm katmanlardaki ağırlıkları güncellemek için geriye iletilmesi gerekir ve çok fazla parametre olduğunda karışma [*interference*] olur. Evrişimli bir ağda, her birim önceki birimlerin sadece küçük bir altkümesine beslenir ve sonraki birimlerin yalnızca küçük bir altkümesini besler, böylece daha az karışma olur ve eğitim daha hızlı yapılabilir.

Derin bir sinir ağı, bir seferde bir katman ele alınacak şekilde eğitilebilir. Her katmanın amacı, girdisindeki dikkat çekici özellikleri çıkarmaktır ve bu amaçla özkodlayıcı gibi bir yöntem kullanılabilir. Burada etiketlenmemiş verileri kullanabilmemiz gibi ek bir üstünlük vardır – özkodlayıcı gözetimsizdir ve bu nedenle etiketli veriye ihtiyaç duymaz. Böylece ham girdiden başlayarak bir özkodlayıcıyı eğitiriz ve onun saklı katmanında öğrenilen kodlanmış gösterim, bir sonraki özkodlayıcıyı eğitmek için girdi olarak kullanılır; bunu gözetimli eğitilecek son katmana dek tekrar ederiz. Tüm katmanlar bu şekilde birer birer eğitildikten sonra hepsi birbiri ardına eklenir ve böyle istiflenmiş tüm özkodlayıcıların ağırlıklarına etiketli verilerle istenirse ek bir gözetimli ayarlama yapılabilir.

Çok sayıda etiketli veri ve büyük miktarda bilgisayar gücü olduğunda tüm derin ağ gözetimli bir şekilde eğitilebilir, ancak ağırlıkları başlatmak için gözetimsiz bir yöntem kullanmak, rasgele başlatma

işlemlerinden çok daha iyi çalışır. Ayrıca öğrenme çok daha hızlı ve daha az etiketli veri ile yapılabilir.

Derin öğrenme yöntemleri çekicidir çünkü elle müdahaleye daha az gerek duyarlar. Doğru özellikleri veya uygun dönüşümleri bizim oluşturmamız gerekmez. Yeterince veri –ki günümüzde "büyük" verimiz var– ve yeterli bilgisayar gücü bulunduğunda –ki günümüzde binlerce işlemcisiyle veri merkezlerimiz var– sadece bekleyip öğrenme algoritmanın gerekli her şeyi kendi başına keşfetmesine izin veririz.

Derin öğrenmenin temelini oluşturan soyutlama katmanları fikri akla yakındır. Sadece görüntüde –elle yazılmış rakamlarda veya yüz görüntülerinde– değil, birçok uygulamada böyle soyutlama katmanlarını hayal edebiliriz. Bu soyut gösterimleri keşfetmek, doğru tahminin yanı sıra soyutlamaya ve böylece verinin altında yatan sürecin daha iyi tanımlanmasına ve anlaşılmasına olanak sağlar.

Bir başka çarpıcı örnek, iyi özellik çıkarıcılara, yani iyi saklı gösterimlere duyulan ihtiyacın en belirgin şekilde görüldüğü doğal dil işlemedir. Araştırmacılar bir dilde sözcükler arasındaki ilişkileri temsil etmek için *ontoloji* adı verilen önceden tanımlanmış veritabanları önermiştir ve bu tür veritabanları kısmen başarı kaydetmiştir. Ancak bu tür ilişkileri çok büyük miktarda veriden otomatik öğrenmek en iyisidir. Farklı soyutlama düzeylerindeki hiyerarşileri öğrenen derin ağlar bunu yapmanın bir yolu olabilir. Özkodlayıcılar ve özçağrılı [*recursive*] özkodlayıcılar, bu tür derin mimarilerin yapıtaşları olmak için uygun adaylardır. Özyineli bir özkodlayıcı, öğrenilen saklı gösterimin yalnızca geçerli girdiye değil, önceki girdilere de bağlı olması için eğitilebilir.

Yapay çeviriyi düşünün. İngilizce bir cümleden başlayarak, İngilizcenin sözcüksel, sözdizimsel ve anlamsal kurallarını kodlamak için çok büyük bir İngilizce derlemden otomatik olarak öğrenilen çoklu işlem ve soyutlama düzeyleriyle en soyut gösterime ulaşırız. Şimdi aynı cümlenin Fransızcasını düşünelim. Bu kez Fransızca bir derlemden öğrenilen işlem düzeyleri farklı olacaktır, ancak iki cümle

aynı anlama geliyorsa, en soyut, dilden bağımsız düzeyde oluşturulan gösterimlerin oldukça benzer olmasını bekleriz.

Dili anlama, verilen bir cümleye göre bu üst düzey soyut gösterimi çıkardığımız bir kodlama işlemidir; dil üretimi de böyle bir üst düzey gösterimden doğal bir dil cümlesini sentezlediğimiz bir kod çözme işlemidir. Çeviri sırasında kaynak dili kodlayıp hedef dilde bu kodu çözeriz. Bir diyalog sisteminde ilk önce soruyu soyut bir düzeyde kodlar ve soyut düzeyde bir cevap oluşturmak üzere işleriz, sonra da bu kodu çözerek yanıtı oluştururuz.

Tek bir derin ağla beyin olmaz. Derin ağlar günümüzde hâlâ görece sınırlı alanlarda çalışmaktadır; yine de ağlar büyüyüp daha fazla veriyle eğitildikçe her gün daha etkileyici sonuçların ortaya çıktığını da görüyoruz.

5

ÖBEKLERİ VE ÖNERİLERİ ÖĞRENME

Verideki Grupları Bulma

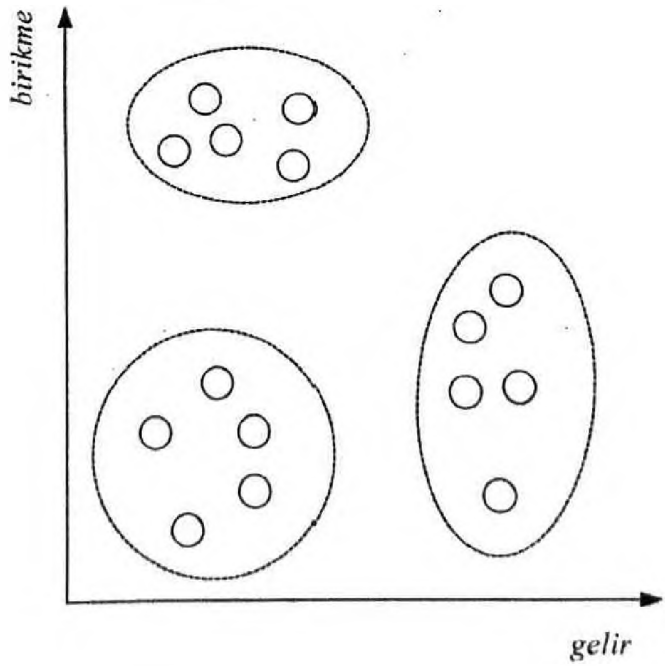
Daha önce verinin girdi ve çıktı çiftlerinden (örneğin otomobil nitelikleri ve fiyatı) oluştuğu ve amacın girdiyi çıktıyla eşleştirmeyi öğrenmek olduğu gözetimli öğrenmeyi ele aldık. Bir gözetmen örnek girdiler için doğru çıktı değerlerini sağlar ve bir modelin parametreleri çıktıyı bu istenen çıktılara mümkün olduğunca yaklaştıracak biçimde güncellenir.

Şimdi, önceden tanımlanmış bir çıktının ve dolayısıyla böyle bir gözetmenin olmadığı *gözetimsiz öğrenmeyi* tartışacağız; burada elimizde sadece girdi verisi vardır. Gözetimsiz öğrenmedeki amaç, girdideki düzenlilikleri bulmak, normalde ne olduğunu görmektir. Girdi alanında belirli örüntülerin diğerlerinden daha sık ortaya çıktığı bir yapı vardır ve biz de genel olarak neyin olup neyin olmadığını görmek isteriz.

Gözetimsiz öğrenme yöntemlerinden biri, amacın girdi öbeklerini ya da gruplarını bulmak olduğu öbeklemedir [*clustering*]. İstatistikte bu yaklaşıma *karışım modelleri* denir.

Bir şirket düşünelim; müşteri verileri, yaş, cinsiyet, posta kodu vb. demografik bilgileri ve ayrıca şirketle yapılan geçmiş işlemleri içerir. Şirket, hangi tip müşterilerin sıklıkla ortaya çıktığını görmek için müşterilerinin profil dağılımını görmek isteyebilir. Böyle bir durumda öbekleme modeli, özellikleri bakımından benzer müşterileri aynı gruba yerleştirir ve şirkete müşterilerinin doğal bir gruplamasını sunar; buna *müşteri segmentasyonu* denir. Bu tür gruplar bulunduktan sonra şirket farklı gruplara özgü stratejilere, örneğin hizmetlere ve ürünlere karar verebilir. Buna *müşteri ilişkileri yönetimi* (MİY) denir.

Bu tür bir gruplandırma aynı zamanda şirketin aykırı örnekleri, yani bilinen müşteri gruplarından farklı olan kişileri saptamasına izin verir. Bunlar daha fazla araştırma gerektiren müşterilerdir ve pazarda şirket tarafından daha fazla yararlanılabilecek yeni bir müşteri tipi ya da kaybedilmesi olası müşteriler olabilir. (bkz. Şekil 5.1.)



Şekil 5.1 Müşteri öbekleme. Burada girdi olarak müşterilerin gelir ve birikim bilgileri kullanılmıştır. Bu örnekte, üç müşteri segmenti görünüyor. Böyle bir öbekleme, farklı segmentteki müşterilerin özelliklerini anlamamızı sağlar; böylece her segmentle farklı etkileşimler tanımlayabiliriz. Buna müşteri ilişkileri yönetimi denir.

Birçok farklı uygulamada, küçük değişikliklerle tekrarlanan düzenlilikleri ve örüntüleri görmeyi bekleriz. Bunları temel öğeler olarak algılamak ve ilgisiz kısımları (yani gürültüyü) göz ardı etmek sıkıştırma yapmanın bir yoludur. Örneğin bir görüntü, temelde piksellerden oluşur, ancak doku ve nesneler gibi tekrarlanan örüntüleri analiz ederek düzenlilikleri tanımlayabiliriz. Bu, görüntünün daha yüksek düzeyde, daha basit ve kullanışlı bir tanımını olanaklı kılar ve piksel düzeyinde sıkıştırmadan daha iyi bir sıkıştırma sağlar. Taranan bir belge sayfasında rasgele 0/1 pikseller değil, karakter görüntüleri bulunur. Veride bir düzen vardır ve bu düzeni, farklı açıda çizgiler kullanarak verinin daha kısa bir tanımını oluşturmada kullanırız. Daha da ileri gidip bu çizgilerin karakterleri oluşturmak için nasıl birleştiğini keşfedebilirsek bir karakter görüntüsü yerine yalnızca o karakterin kodunu kullanabiliriz ki bu, karakterin görüntüsünden çok daha kısadır.

Belge öbeklemede amaç, benzer belgeleri gruplandırmaktır. Örneğin haberler, siyaset, spor, moda, sanat ve benzeri konularla ilgili alt türlere ayrılabilir. Belgeyi, böylesi belge türlerini yansıtan bir sözlüğe dayanan bir sözcük torbası kullanarak temsil edebiliriz ve ardından belgeler, kullanılan sözcüklere bağlı olarak gruplandırılır. Sözlüğün nasıl seçildiği elbette çok önemlidir.

Gözetimsiz öğrenme yöntemleri biyoenformatikte de kullanılmaktadır. Genomumuzdaki DNA, "yaşamın modeli"dir ve A, G, C ve T gibi bazların dizisidir. RNA, DNA'dan çeviriyazılır [*transcribe*] ve proteinler RNA'dan çevrilir [*translate*]. Proteinler, canlı bedenin olduğu ve yaptığı şeydir. DNA'nın bir baz dizisi olması gibi, bir protein de (bazlarla tanımlanan) bir aminoasit dizisidir. Bilgisayar biliminin moleküler biyolojideki uygulama alanlarından biri, bir diziyi diğeriyle eşleştiren hizalamadır [*alignment*]. Bu zor bir dizi eşleştirme problemidir, çünkü diziler oldukça uzun olabilir, eşlenecek çok sayıda şablon dizisi bulunabilir ve gen dizilerinde silme, ekleme veya değiştirme yüzünden bozulmalar olabilir.

Öbekleme, proteinlerde tekrar tekrar ortaya çıkan aminoasit dizileri olan *motifleri* öğrenmede kullanılır. Motifler ilgi çekicidir, çünkü yapısal veya işlevsel unsurlara karşılık gelebilirler. Aminoasit-

leri harf, proteinleri cümle olarak düşünürsek motifler sözcüklere benzer; farklı cümlelerde sıklıkla ortaya çıkan ve belirli bir anlamı olan harf dizileridir.

Öbekleme, veride doğal olarak ortaya çıkan grupları tanımladığımız bir *keşifçi veri analizi* tekniği olarak kullanılabilir. Örneğin daha sonra bu grupları sınıf olarak etiketleyebilir ve ardından onları sınıflandırmayı deneyebiliriz. Bir şirket, müşterilerini öbekleyip segmentleri bulabilir ve daha sonra belirli bir hedef için (örneğin, zaman içinde kaybedilen müşteriler) bazı segmentleri etiketleyebilir ve yeni müşterilerin davranışını tahmin etmek için bir sınıflandırıcı eğitebilir. Ancak önemli olan nokta şudur ki, varlığını hiçbir uzmanın öngöremeyeceği öbekler olabilir ve gözetimsiz, veri güdümlü analizin gücü burada ortaya çıkar.

Bazen bir sınıf birden fazla gruptan oluşur. Optik karakter tanıma örneğini düşünün. Yedi rakamını yazmanın iki yolu vardır; Amerikan versiyonu "7" iken, Avrupa versiyonu ortada yatay bir çubuğa sahiptir (Avrupa'da kullanılan ve üstte küçük bir çizgi taşıyan "1"den ayırmak için). Böyle bir durumda, örneklem her iki kıtadan da örnekler içerdiğinde, yedi sınıfının iki grubun birleşimi/karışımı olarak gösterilmesi gerekir.

Benzer bir durum, telaffuz, aksan, cinsiyet, yaş, vb. farklılıklar nedeniyle aynı sözcüğün farklı şekillerde söylenebildiği konuşma tanımada da ortaya çıkar. İstatistiksel olarak doğru olmak için, geçerli tek bir yol olmadığında tüm farklı yolların eşit derecede geçerli seçenekler olarak gösterilmeleri gerekir.

Öbekleme algoritmaları, örnekleri girdi gösterimleri açısından hesaplanan benzerliklerine göre gruplandırır ve örnekler arasındaki benzerlik, bu gösterimi oluşturan özelliklerdeki benzerlikler toplanarak ölçülür. Bazı uygulamalarda örnekler arasında asıl veri yapısına bağlı olarak doğrudan, böyle bir özellik gösterimi tanımlamayıp benzerlik hesaplamaları yapmaksızın hazır bir benzerlik ölçüsü tanımlayabiliriz.

Web sayfalarını öbeklemeyi düşünelim. Benzerlik hesabında, metin alanına ek olarak, başlıklar veya anahtar sözcükler gibi meta

(veya üstbaşlık türü) bilgilerdeki benzerliği veya bu ikisine bağlantı veren veya bunlara bağlanan ortak web sayfalarının sayısını kullanabiliriz. Bu bize, web sayfalarının metnindeki sözcük torbası gösterimini kullanarak hesaplanandan daha iyi bir benzerlik ölçüsü verir. Uygulamaya daha fazla uyan bir benzerlik ölçüsü kullanmak (eğer tanımlanabilirse) daha iyi öbekleme sonuçlarına yol açar. İzgesel [*spectral*] öbeklemede temel fikir budur.

Bu tür uygulamaya özgü benzerlik gösterimleri, çekirdek fonksiyonu [*kernel function*] adıyla gözetimli öğrenme uygulamalarında da kullanılır. Destek vektör makinesi [*support vector machine*] (Vapnik 1998), hem sınıflandırma hem de regresyon için kullanılan böyle bir öğrenme algoritmasıdır.

Hiyerarşik öbekleme yapmak da mümkündür; burada düz bir öbek listesi yerine farklı çözünürlük düzeylerinde kümeler içeren ve daha yukarıdaki öbeklerin daha küçük öbeklere bölündüğü bir ağaç yapısı oluştururuz. Bu tür ağaçlara, biyolojiden (örneğin Linnaeus'un taksonomisinden) veya insan dillerinden aşinayız. Öbeklerin daha küçük öbeklere ayrılmasının bir açıklaması türoluş [*phylogeny*] yani evrimsel değişimlerle ilgilidir –küçük mutasyonlar, bir tür ikiye ayrılıncaya kadar zaman içinde yavaş yavaş birikir; başka uygulamalarda benzerliğin nedeni farklı olabilir.

Özellikle öbeklemede veya genel olarak gözetimsiz öğrenmede amaç, veride yapı bulmaktır. Gözetimli öğrenmede (örneğin sınıflandırmada) bu yapı, farklı sınıfları tanımlayan ve öğrenme verisindeki örnekleri bu sınıflara göre etiketleyen bir gözetmen tarafından dışarıdan belirlenir. Gözetmen tarafından sağlanan bu ek bilgi elbette yararlıdır, ancak bunun bir yanlılık kaynağına dönüşmediğinden veya yapay sınırlar getirmediğine her zaman emin olmalıyız. Ayrıca etiketlemede hata olma riski vardır; buna "öğretmen gürültüsü" denir.

Gözetimsiz öğrenme önemli bir araştırma alanıdır, çünkü etiketlenmemiş veriler bulmak çok daha kolay ve ucuzdur. Konuşma tanımada radyo etiketlenmemiş konuşma verisi kaynağıdır. Buradaki düşünce, etiketlenmemiş veriden temel yapıyı ve neyin tipik olduğunu çıkarmak ve bunları daha sonra farklı amaçlar için gözetimli olarak

kullanmaktır. Bir bebek ilk birkaç yılını etrafına bakınarak geçirir; bu zaman diliminde farklı koşullar altında aynı şeyleri, nesneleri, yüzleri tekrar tekrar görür ve yavaş yavaş kendi temel özellik çıkarıcılarını ve onların nesneler oluşturmak üzere nasıl bir araya geldiklerini öğrenir. Ardından konuşmayı öğrendiğinde bu nesnelerin etiketlerini öğrenir.

Öneri Sistemleri

Birinci bölümde, bir yapay öğrenme uygulaması olarak müşteri davranışını tahmin eden *öneri sistemlerini* ele aldık. Geniş bir müşteri işlem veri kümesi söz konusu olduğunda, "X'i alan insanların Y'yi de satın alması olasıdır" şeklindeki ilişkilendirme [*association*] kurallarını bulabiliriz. Böyle bir kural, X'i satın alan müşterilerin büyük bir yüzdesinin Y'yi de aldığını belirtir. X'i alan ancak Y'yi almayan müşteriyi potansiyel bir Y müşterisi olarak hedefleyebiliriz. X ve Y, ürünler, yazarlar, gidilecek şehirler, izlenecek videolar ve benzerleri olabilir. Her gün, özellikle de internette gezinirken bu tür öğrenilmiş öneri örneklerine sık sık rastlıyoruz.

Bu hedefleme yaklaşımı sıkça kullanılsa ve bu kuralları çok büyük veri kümelerinden öğrenmek için etkili algoritmalar ileri sürülmüş olsa da bugünlerde üretici modelleri kullanan ilginç algoritmalar daha öne çıkmaktadır.

Hatırlanacağı üzere üretici bir model tasarlarken verinin nasıl üretildiğini modellemeye çalışırız. Dolayısıyla müşteri davranışında, bu davranışı etkileyen etkenleri göz önünde bulundururuz. İnsanların rasgele alışveriş yapmadıklarını biliyoruz. Satın aldıkları mallar, hane halkı bileşimine, yani kaç kişiyle birlikte yaşadıklarına, onların cinsiyetlerine, yaşlarına, gelirlerine, beğenilerine (ki bu da yaşadıkları yer gibi başka etkenlerin sonucudur) ve benzeri bir dizi etkene bağlıdır. Her ne kadar bazı şirketler müşteri kartlarıyla bu bilgilerin bir kısmını toplasa da gerçek dünyada bu etkenlerin çoğu bilinmez, saklıdır ve gözlemlenen verilerden çıkarılması gerekir.

Bununla birlikte, böylesi etkenlere aşırı güvenin yanıltıcı olabileceğini de unutmayalım, çünkü bunlar yanlış veya eksik olabilir. İlk

başta aklımıza gelmeyen veya düşündüğümüz kadar önemli olmayan etkenler de olabilir, bu yüzden bunları veriden öğrenmek (keşfetmek) her zaman en iyisidir.

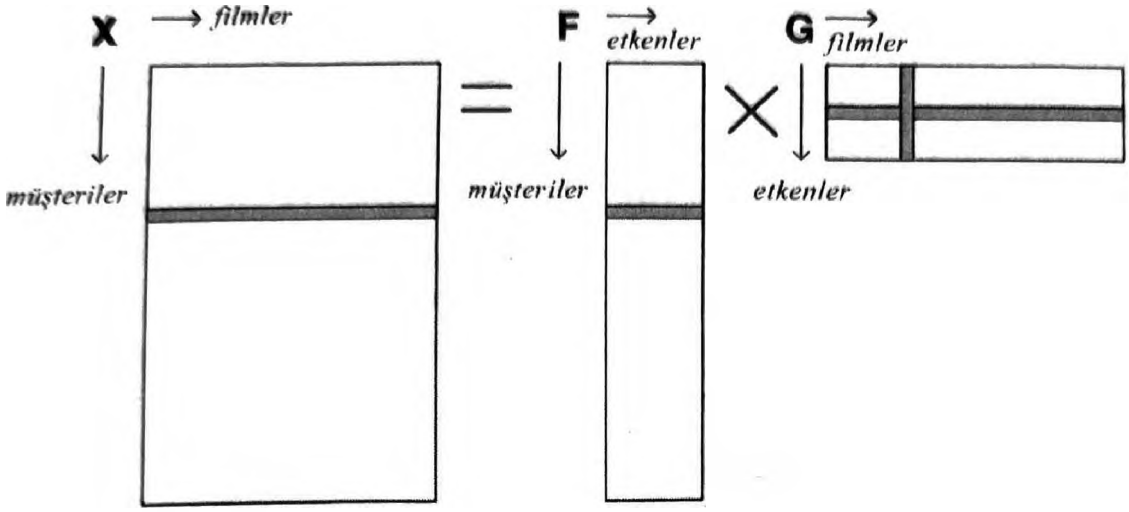
Değerleri –örneğin satın alınan miktarları ya da beğeni değerleri– gözlemlenebilir çok sayıda ürün olabilir, ancak bunların satın alınıp alınmadığını belirleyen az sayıda saklı etken vardır. Bir müşteri için bu etkenleri veriden çıkarabilirsek o müşterinin daha sonraki satın alma davranışı hakkında daha doğru tahminlerde bulunabileceğimizi düşünürüz.

Bu tür saklı nedenleri ortaya çıkarmak, ürünler arasındaki doğrudan ilişkileri öğrenmeye çalışmaktan daha iyi bir model oluşturacaktır. Örneğin çocuk bezi, süt, bebek maması, ıslak mendil vb. farklı öğelerin satın alınmasına yol açacak olan saklı bir etken, "evdeki bebek" olabilir. Bu nedenle bu ürünlerin çiftleri veya üçlüleri arasındaki ilişkilendirme kurallarını öğrenmek yerine, geçmiş satın alma hareketlerine dayanarak saklı bebek etkenini tahmin edebilirsek, henüz satın alınmamış ve bir bebek için gereken başka ürünlerin tahmin edilmesi tetiklenecektir.

Bu tür birçok etken olabilir. Her müşteri bunların bir kısmı tarafından etkilenir (veya bunlarca tanımlanır) ve her etken, ürünlerin bir altkümesini tetikler. Etken değerleri, 0 veya 1 değil de sürekli olursa bu dağıtık gösterim, veriyi temsil etme konusunda bir zenginlik sağlar.

Bu yaklaşım, yapıyı verileri iki parçaya ayırarak bulmayı amaçlar. Birincisi, yani müşteriler ve etkenler arasındaki eşleştirme, bir müşteriyi (farklı ağırlıktaki) etkenler açısından tanımlar. İkincisi, yani etkenlerle ürünler arasındaki eşleştirme ise bir etkeni (farklı ağırlıktaki) ürünler açısından tanımlar. Matematikte verileri matrisler kullanarak gösteririz, bu yüzden bu yaklaşıma matris ayrışımı [*matrix decomposition*] denir.

Gizli etkenlere sahip böyle bir üretici yaklaşım, başka birçok uygulamada da yarar sağlar. Film önerileri örneğini ele alalım (bkz. Şekil 5.2). Belli sayıda müşterimiz bazı filmleri izlemiş olsun; amacımız, müşterilere henüz izlememiş oldukları filmler arasından bir öneride bulunmaktır.



Şekil 5.2 Film önerileri için matris ayrışımı. X veri matrisindeki her satır bir müşterinin hangi filmleri seyrettiğini gösterir; bu değerlerin çoğu sıfırdır (çünkü o müşteri o filmi seyretmemiştir). X matrisi, F ve G matrislerinin çarpımı olarak ayrışır, öyle ki F matrisinin her satırı bir müşteriyi etkenler cinsinden tanımlarken G matrisinin her satırı bir etkenin filmler üstündeki etkisini tanımlar. G matrisinin her sütunu bir filmi etkenler cinsinden tanımlar. Etken sayısı, hem müşteri hem de film sayısından çok daha azdır ve verinin karmaşıklığını belirleyen odur; buna X matrisinin kertesi denir.

Bu problemin ilk özelliği, çok sayıda müşterimiz ve filmimiz varken, verilerin seyrek olmasıdır. Her müşteri filmlerin yalnızca küçük bir yüzdesini izlemiştir ve çoğu film, müşterilerin yalnızca küçük bir yüzdesi tarafından izlenmiştir. Bu veriye dayanarak öğrenme algoritmasının yeni filmler veya yeni müşteriler eklendiğinde bile başarılı bir şekilde genelleştirme ve doğru tahmin yapabilmesi gerekir.

Bu durumda da aksiyon, komedi ve benzeri bazı türlerin seçilme olasılığını artıran, müşterinin yaşı ve cinsiyeti gibi saklı etkenleri düşünebiliriz. Ayrışım kullanarak her müşteriyi bu etkenler bakımından (farklı oranlarda) tanımlayabiliriz ve bu etkenlerin her biri belli bazı filmleri (farklı olasılıklarla) tetikler. Bu, film çiftleri arasında kurallar bulmaya çalışmaktan daha iyi, daha kolay ve daha ucuzdur. Bu etkenlerin önceden tanımlanmadığını, ancak öğrenme sırasında otomatik olarak keşfedildiğini tekrar söyleyelim. Ama bir veri kümesinde böyle etkenler her zaman bulunabilse de bunları yorumlamak veya anlamlandırmak her zaman kolay olmayabilir.

Diğer bir olası uygulama alanı *belge modellemedir* (Blei 2012). Diyelim ki çok sayıda belgemiz var ve bunlar belli bir sözcük torbası kullanılarak yazılmış. Veri matrisi yine seyrektir; her belge sözcüklerin küçük bir altkümesini kullanır. Burada saklı etkenlerini farklı konular olarak yorumlayabiliriz. Bir muhabir haber yazdığında belli birkaç konu hakkında yazar; bu nedenle her belge bazı konuların bir bileşimidir ve her konu, olası tüm sözcüklerin küçük bir altkümesini kullanarak yazılır. Buna örtük anlamsal analiz [*latent semantic analysis*] denir. Bunun, "*X* sözcüğünü kullanan kişiler *Y* sözcüğünü de kullanır," gibi kurallar ortaya çıkarmaya çalışmaktan daha akla yatkın olduğu açıktır.

Verilerin saklı etkenlerle nasıl üretildiği ve bu saklı etkenlerin gözlemlenebilir verileri üretmek için nasıl bir araya geldikleri hakkında akıl yormak önemlidir ve tahmini çok daha kolay yapabilir. Burada tartıştığımız, saklı etkenlerin etkilerinin toplamını aldığımız doğrusal modeldir. Modeller her zaman doğrusal değildir –örneğin bir etken başka bir etkeni ketleyebilir– ve bu tür doğrusal olmayan üretici modelleri veriden öğrenmek, yapay öğrenmede önemli güncel araştırma yönelimlerinden biridir.

6

EYLEM YAPMAYI ÖĞRENME

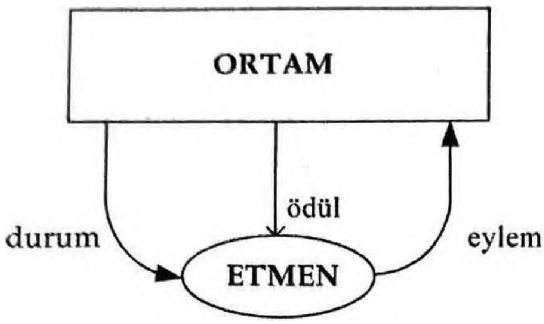
Pekiştirmeli Öğrenme

Satranç oynamayı öğrenen bir makine yapmak istediğimizi düşünelim. Hem kendimize hem de rakibimize ait taşların tahtadaki konumlarını görmek için bir kameramız olduğunu varsayalım ve amaç, oyunu kazanacak şekilde hamlelerimize karar vermek olsun.

Bu durumda gözetimli öğrenmeyi iki nedenden dolayı kullanamayız. Birincisi, taşların tahtadaki her durumu için en iyi hamleyi göstererek bizi birçok oyun üzerinden eğitecek bir öğretmene sahip olmak çok maliyetlidir. İkincisi, çoğu durumda en iyi hamle diye bir şey yoktur. Bir hamlenin ne kadar iyi olduğu, onu takip eden hamlelere bağlıdır. Tek bir hamle sayılmaz; bir hamleler dizisi, eğer onlardan sonra oyunu kazanıyorsak iyidir. Tek gerçek geribildirim oyunun sonunda, oyunu kazandığımız veya kaybettiğimiz zaman gelir.

Bir başka örnek, bir labirent içinde hedef bir konumu bulması gereken robottur. Robot herhangi bir anda dört pusula yönünden birinde hareket edebilir ve hedefe ulaşmak için bir dizi eylemde bulunmak zorundadır. Robot hedef konuma ulaşana kadar birçok eyle-

mi dener ve bu ara eylemlerde bir geribildirim yoktur. Ancak hedefe ulaştığında bir ödül alır. Bu durumda rakip yoktur fakat hedefe daha kısa yoldan ulaşmak gibi bir tercihimiz olabilir –robot bir bataryayla çalışıyor olabilir– bu da zamana karşı oynadığımız anlamına gelir.



Şekil 6.1 Pekiştirmeli öğrenmede etmen bir ortam içindedir. Ortamın her durumunda etmen bir eylem seçer ve bu eylem ortamın durumunu değiştirir; ayrıca bazen bir ödül getirir.

Bu iki uygulamanın bir dizi ortak noktası bulunur. Bir *ortam içinde*, *etmen* olarak adlandırılan bir karar verici vardır (bkz. Şekil 6.1). İlk örneğimizde, satranç tahtası oyun oynayan etmenin ortamıdır; ikinci örnekte, labirent robotun ortamıdır. Herhangi bir zamanda ortam, belirli bir *durumdadır*, bu da sırasıyla tahtadaki taşların konumu veya robotun labirentin içindeki konumuna karşılık gelir. Karar vericinin olası bir *eylemler* kümesi vardır: satranç tahtasında taşların nizami hareketi ve robotun herhangi bir engele çarpmadan çeşitli yönlerdeki hareketi gibi. Bir eylem seçilip gerçekleştirildiğinde ortam ve durum değişir.

Görevin çözümü bir dizi eylem gerektirir ve bir *ödül* biçiminde geribildirim alırız. Öğrenmeyi zorlaştıran şey, ödülün nadiren ve genellikle sadece tüm eylem dizisi tamamlandıktan sonra gelmesidir – oyunu uzun bir hamle dizisinden sonra kazanır veya kaybederiz. Ödül, görevin amacını tanımlar ve öğrenme için gereklidir. Etmenin öğrenmesi gereken; en iyi eylem dizisini üretebilmektir ve burada "en iyi"; en fazla ödülü en erken döndüren eylem dizisidir. Buna *pekiştirmeli öğrenme* diyoruz (Sutton ve Barto 1998).

Pekiştirmeli öğrenme, önceki bölümlerde gördüğümüz öğrenme yöntemlerinden bazı açılardan farklıdır. Gözetimli öğrenmede sahip olduğumuz gözetmenle öğrenmenin aksine, "eleştirmenle öğrenme" olarak adlandırılır. Bir eleştirmeni öğretmenden ayıran şey, ne yapmamız gerektiğini söylememesi, yalnızca geçmişte ne kadar iyi yaptığımızı söylemesidir. Eleştirmen asla önceden bilgilendirmez! Eleştirmenden gelen geribildirim seyrektir ve geldiğinde geç gelir. Bu sorumluluk atama [*credit assignment*] sorununa yol açar: Birkaç deneme gerçekleştirip ödül kazandıktan sonra bu denemeler sırasında yaptığımız eylemleri değerlendirmek ve ödülü kazanmamızı sağlayan hamleleri bulup öğrenmek isteriz.

Aslında bir pekiştirmeli öğrenme programının yaptığı şey, ara durumların veya eylemlerin bizi hedefe yönlendirmede ve gerçek bir ödül kazandırmada ne kadar iyi olduklarını gösteren bir *içsel değer* oluşturmaktır. Böyle bir içsel ödüllendirme mekanizması öğrenildikten sonra, etmen en fazla ödülü getirecek eylemi seçebilir. Görevin çözümü, toplamda en yüksek gerçek ödülü verecek şekilde seçilen bir *dizi* eylem gerektirir.

Önceden ele aldığımız senaryoların aksine, burada eğitim verilerini sağlayan dışsal bir süreç yoktur. Etmenin kendisi ortam içinde farklı eylemleri deneyip bir ödül şeklinde geribildirim alarak (veya almayarak) kendisi veri üretir. Daha sonra bilgisini güncellemek için bu geribildirimi kullanır, böylece zaman içerisinde en yüksek ödülü döndüren eylemleri seçmeyi öğrenir.

K Kollu Kumar Makinesi

Basit bir örnekle başlayalım. *K* kollu kumar makinesi [*K-armed bandit*], *K* kollu varsayımsal bir kumar makinesidir. Burada eylem, kollardan birini seçip çekmektir ve seçilen kolla (eylemle) ilişkili ödüle karşılık gelen belli bir miktar para kazanırız. Görev, en çok ödülü almak için hangi kolun çekileceğine karar vermektir.

Bu, *K seçenekten* birini seçtiğimiz bir sınıflandırma problemidir. Gözetimli öğrenme olsaydı, öğretmen bize doğru sınıfı, yani en

yüksek kazancı getiren kolu söylerdi. Bu pekiştirmeli öğrenme durumunda ise yalnızca farklı kolları deneyebilir ve en iyisinin kaydını tutabiliriz.

Tüm kollar için başta tahmin edilen değer sıfırdır. Ortamı keşfetmek için, kollardan birini rasgele seçip bize getirdiği ödülü gözlemleyebiliriz. Eğer bu ödül sıfırdan yüksekse, onu bu eylemin içsel ödül tahmini olarak saklayabiliriz. Sonra tekrar bir kol seçmemiz gerektiğinde, sürekli o kolu çekip pozitif ödüller almaya devam edebiliriz. Ancak başka bir kol daha yüksek bir ödül getiriyor olabilir, bu yüzden pozitif ödülü olan bir kol bulduktan sonra bile öteki kolları denemek isteriz. Kendimizi bir yola sabitlemeden önce tüm seçenekleri yeterince araştırdığımıza emin olmalıyız. Tüm kolları denedikten ve bilinmesi gereken her şeyi öğrendikten sonra en yüksek ödülü getiren eylemi belirleyebiliriz.

Buradaki senaryo, ödüllerin belirlenimci olduğunu, yani bir kol için her zaman aynı ödülü aldığımızı varsaymaktadır. Gerçek bir kollu kumar makinesinde ödül bir şans meselesidir ve aynı kol farklı denemelerde farklı ödül değerlerine yol açabilir. Böyle bir durumda, en üst düzeye çıkarmak istediğimiz *beklenen ödüldür* ve bunu yaparken her eylem için içsel ödül tahminimiz, aynı eylemde dönen tüm ödül değerlerinin ortalamasıdır. Bu, bir kez bir eylemde bulunmanın, o eylemin ne kadar iyi olduğunu öğrenmek için yeterli olmadığı anlamına gelir. Ortalama hakkında iyi bir tahmin hesaplamak için birçok deneme yapmamız, aynı kolu birçok kez çekip birçok gözlem (ödül) almamız gerekir.

K kollu kumar makinesi, basitleştirilmiş bir pekiştirmeli öğrenme problemidir, çünkü sadece bir durum –bir kumar makinesi– vardır. Genel durumda, etmen bir eylem seçtiğinde bir ödül alır ya da almaz ve bunun yanında durumu da değişir. Etmenin bu sonraki durumu, ortamdaki saklı etkenler nedeniyle rastsal da olabilir ve bu, aynı işlem için farklı ödüllere yol açabilir.

Örneğin şans oyunlarında rastsallık vardır: Bazı oyunlarda zar atılır ya da kart oyunlarında bir desteden rasgele kart çekeriz. Satranç gibi bir oyunda zar veya kart destesi bulunmaz ancak davranış-

ları öngörülemeyen bir rakip vardır ki bu da başka bir belirsizlik kaynağıdır. Robot bir çevrede dolaşırken, engeller yer değiştirebilir veya algıyı bozan veya hareketi sınırlayabilen başka hareketli etmenler olabilir. Sensörler gürültülü olabilir ve hareket sağlayıcılarını [*actuator*] kontrol eden motorlarda bazı kusurlar bulunabilir. Bir robot ileri gitmek isterken aşınma ve yıpranma nedeniyle sağa veya sola kayabilir. Bütün bunlar, belirsizliği ortaya çıkaran saklı etkenlerdir ve her zaman olduğu gibi, belirsizliğin etkisini ortalama alarak azaltmak için beklenen değerleri kestiririz.

K kollu kumar makinesinin sadeleştirilmiş olmasının bir başka nedeni de ödülü tek bir eylemden sonra almamızdır. Ödül ertelenmez ve eylemimizin değerini hemen görürüz. Bir satranç oyununda veya görevi bir odadaki hedef konumu bulmak olan bir robot durumundaysa ödül ancak en sonda gelir; ara eylemler için ödül veya başka bir tür geribildirim almayız.

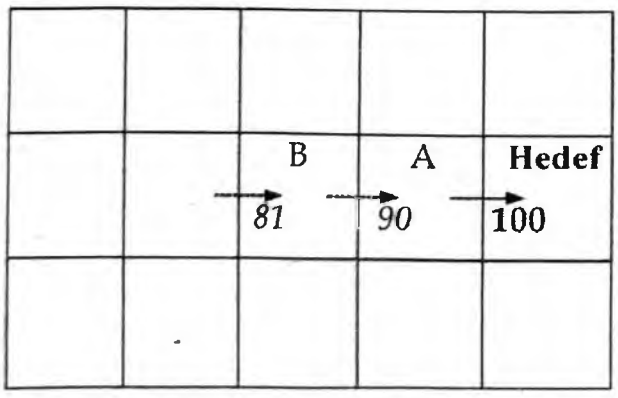
Pekiştirmeli öğrenmede istediğimiz şey, herhangi bir ara eylemin bizi gerçek ödüle götürmede ne kadar iyi olduğunu tahmin edebilmektir. Bu, o eylem için içsel ödül tahminimizdir. Başlangıçta tüm eylemler için bu ödül tahmini sıfırdır çünkü henüz herhangi bir şey bilmiyoruzdur. Öğrenmek için verilere ihtiyacımız vardır, bu yüzden bazı eylemleri rasgele denediğimiz ve sonucunda herhangi bir ödül alıp almadığımızı gözlemlediğimiz bir ***keşif*** [*exploration*] süreciyle başlarız. Daha sonra bu bilgiyi kullanarak içsel tahminlerimizi güncelleriz.

Daha fazlasını keşfettikçe daha fazla veri toplarız ve ortam ile eylemlerimizin ne kadar iyi olduğu hakkında daha fazla şey öğreniriz. Eğer eylemlerimiz hakkındaki ödül tahminlerimizin yeterince iyi olduğu bir düzeye ulaştığımıza inanırsak, bunları kendi çıkarımıza kullanmaya [*exploitation*] başlayabiliriz. Bunu, içsel ödül tahminlerimize göre en yüksek ödülü üreten eylemleri gerçekleştirerek yaparız. Başlangıçta fazla bir şey bilmediğimiz için eylemleri rasgele deneriz. Daha fazla öğrendikçe, rasgele seçimlerden içsel ödül tahminlerimize göre yaptığımız eylemlere geçer ve böylece kazandığımız ödülü sürekli artırırız.

Zamansal Fark Öğrenme

Herhangi bir durum ve eylem çifti için, o durumdan o eylemle başlayarak amaca dek yapılacak tüm eylemler sonucunda beklenen toplam ödülü kestirmek isteriz. Bu bir *beklenen* değerdir, çünkü ödüllerdeki ve gelecek durumlardaki olası tüm rastsallık kaynakları üzerinden ortalama bir değerdir. Ardışık iki durum-eylem çiftinin beklenen toplam ödülleri, *Bellman denklemi* yoluyla ilişkilendirilir ve bunu aşağıda anlattığımız gibi, ödülleri sonraki eylemlerden önceki eylemlere doğru geri taşımak [*back up*] için kullanırız.

Robotu hedefe götüren son hareketi düşünelim. Hedefe ulaştığımızda 100 birim ödül aldığımızı varsayalım (bkz. Şekil 6.2). Şimdi bundan bir önceki durumu ve eylemi düşünelim. Bu eylem bize hemen bir ödül vermese de (çünkü hedeften halen bir adım ötedeyiz) bizi bir hamle sonra 100 birimlik gerçek ödülü aldığımız duruma götürdüğünü biliyoruz. Bu yüzden bu ödülü, diyelim ki 0,9'luk bir etkenle iskonto eder ve hedeften bir adım önceki durum-eylem çiftinin *içsel ödülünün* 90 birim olduğunu söyleyebiliriz. İskonto ederiz çünkü ödül gelecektedir ve gelecek hiçbir zaman kesin değildir – "Bugünkü tavuk yarınki kazdan iyidir."



Şekil 6.2 Ödülü geriye taşıyarak zamansal fark öğrenme. A durumundayken sağa gidersek hedefe ulaşır ve 100 birimlik gerçek ödülü alırız. Ondan bir önceki B durumunda doğru eylemi seçersek (yani sağa gidersek) bir hamle daha yaparak gerçek ödülü alabileceğimiz A'ya ulaşmış olacağız; bu yüzden B'den sağa gitmenin de ödülü varmış gibi olur. Ama bu ödül iskonto edilir (burada 0,9 çarpanıyla) çünkü bir adım öncedir ve bu hayalî bir iç ödüldür, gerçek ödül değildir. B'den A'ya gitmenin gerçek ödülü hâlâ sıfırdır; 90 birimlik iç ödül bize gerçek ödülü almaya ne kadar yaklaştığımızı gösterir.

Bu bir içsel ödüldür; dışsal, gerçek ödül bu eylem için hâlâ sıfırdır çünkü henüz hedefe ulaşmadık, ama kendimizi hedeften yalnızca bir adım uzaklıktaki bir duruma geldiğimiz için içsel olarak ödüllendiriyoruz. Benzer şekilde, bu eylemden önceki eylem, 81'lik bir içsel ödül alır ve bu eylem dizisinde önceki tüm eylemlere aynı şekilde içsel değerler atamaya devam edebiliriz. Tabii ki bu sadece bir deneme içindir. Öğrenme, ilk durumdan başlayıp amaç duruma dek arka arkaya eylemler yaparak gittiğimiz birçok denemeden oluşur. Her birinde belirsizlikler nedeniyle farklı ödüller gözlemlediğimiz ve sonrasında farklı durumlara geçtiğimiz birçok deneme yapmamız gerekir ve tüm bu iç ödüllendirme tahminleri üzerinden ortalama alırız. Buna zamansal fark (ZF) [*temporal difference* (TD)] *öğrenme* denir. Her bir durum-eylem çifti için içsel ödül tahmini *Q* ile gösterilir ve bunları güncelleyen algoritmaya *Q öğrenme* denir.

Sadece en son eylemin bize gerçek ödülü kazandırdığını unutmayalım. Tüm ara eylemler içsel ve hayalî ödüllerdir. Amaç değildirler; sadece sonunda bizi gerçek ödüle götüren eylemleri bulmamıza yardım ederler. Okulda öğrenci farklı derslerde başarısına bağlı olarak notlar alır, ama bunlar hayalî ödüllerdir; öğrencinin mezun olup toplumun üretken bir üyesi olduğunda yapacağı katkıyı, ki gerçek ödül budur, en fazla yapmak yönünde öğrenciyi yönlendirmeyi sağlarlar.

Bazı uygulamalarda ortam *kısmen gözlemlenebilir* ve bu yüzden etmen durumu tam olarak bilmez. Çevrenin durumunu kısmen kestirmek için kullandığı sensörler olabilir. Diyelim ki odada dolaşan bir robotumuz var. Robot odadaki yerini veya odada başka neyin bulunduğunu bilmiyor olabilir. Robotun bir kamerası olabilir, ancak bir görüntü ona ortamın durumunu tam olarak anlatmaz, sadece olası durum hakkında kısıtlı bilgi verir. Örneğin robot yalnızca solunda bir engelin durduğunu biliyor olabilir.

Böyle bir durumda etmen, kısmi bilgi veren gözlemlere dayanarak durumunu tahmin eder veya daha kesin bir ifadeyle söylersek, etmen gözlemlerine dayanarak her bir durumda olma olasılığını tahmin eder ve tüm olası durumları olasılıklarla ağırlıklandırarak karar verir. Bu ek belirsizlik, görevi çok daha karmaşık ve problemi daha zor öğrenilir yapar (Thrun, Burgard ve Fox 2005).

Pekiştirmeli Öğrenme Uygulamaları

Pekiştirmeli öğrenmenin ilk uygulamalarından biri, kendi kopyasına karşı oynayarak tavla oynamayı öğrenen *TD-Gammon* programıdır (Tesauro 1995). Bu program, yine Tesauro tarafından geliştirilen ve uzmanlar tarafından oynanan oyunlara dayanarak gözetimli eğitilmiş önceki *NeuroGammon* programından üstündür. İçerdiği yaklaşık 10^{20} sayıdaki durum yüzünden iyi tavla oynamak zordur. Hem rakip hem de zar işe rastsallık katar. Zamansal fark öğrenmenin bir türevini kullanan program, kendisinin bir kopyasına karşı 1.500.000 oyun oynayarak usta düzeyine ulaşmıştır.

Gözetimli öğrenme algoritmalarından daha yavaş da olsalar, pekiştirmeli öğrenme algoritmalarının daha geniş bir uygulama çeşitliliğine ve daha iyi öğrenme programları oluşturma potansiyeline sahip oldukları açıktır (Ballard 1997). Herhangi bir gözetime ihtiyaçları yoktur ve öğretmen yanlılığı olmayacağından bu aslında daha iyi olabilir. Örneğin Tesauro'nun TD-Gammon programı, belli durumlar için en iyi oyunculardan daha üstün hamleler keşfetmiştir.

Etkileyici yeni bir çalışma, video oyunları oynamakta pekiştirmeli öğrenmeyi derin öğrenmeyle birleştirdi (Mnih ve diğerleri 2015). Derin Q Ağı [*Deep Q Network*], ekranın doğrudan 84×84 görüntüsünü alan (1980'lere ait bu oyunlarda görüntü çözünürlüğü düşüktü) ve yalnızca görüntü ile puan bilgilerini kullanarak oyunu oynamayı öğrenen evrişimli bir sinir ağından oluşur. Eğitim, piksellerden eylemlere kadar uçtan ucadır. Aynı algoritma, ağ mimarisi ve üst parametrelere sahip aynı ağ, çeşitli video oyunlarından herhangi birini öğrenebilir. Bir insan kadar iyi oynayabilen program, programcıları tarafından öngörülmeyen veya beklenmeyen ilginç stratejiler öğrenmiştir.

Yakın zamanda aynı araştırma grubu, Go oynamak için yine derin evrişimli ağları pekiştirmeli öğrenmeyle birleştiren *AlphaGo* programını geliştirdi (Silver ve diğerleri 2016). Girdi 19×19 Go tahtasıdır ve en iyi hamleyi seçmek üzere eğitilmiş bir politika ağı [*policy network*] ve programın oyunu kazanmada ne kadar iyi olduğunu

değerlendirmek üzere eğitilmiş bir değer ağı [*value network*] vardır. Politika ağı ilk önce çok büyük bir uzman oyunları veritabanıyla eğitilmiş ve daha sonra kendi kopyasına karşı oynayarak pekiştirmeli öğrenme yoluyla daha da gelişmiştir. *AlphaGo*, 2015'te Avrupa Go şampiyonunu 5-0 ve 2016 yılının Mart ayında dünyanın en büyük Go oyuncularından birini 4-1 mağlup etti.

Bu yaklaşımları daha karmaşık girdi ve daha büyük eylem kümeleriyle daha karmaşık senaryolara ölçeklendirmek, pekiştirmeli öğrenmedeki şu anki zorluklardan biridir. Bu sistemlerin en önemli ve etkileyici özelliği, eğitimin algı ve eylemden oluşan iki parça arasında bir ara gösterim varsayılmadan, ham girdiden eyleme kadar uçtan uca yapılmasıdır. Örneğin aynı yaklaşımı satrançta uygulamanın daha zor olması beklenebilir, çünkü tahta daha küçük olsa da Go oyununda olduğu gibi bütün taşlar aynı değildir.

Bu tür bir uçtan uca eğitimin herhangi bir ara işlemin veya gösterimin otomatik olarak veriden öğrenildiği oyun oynama dışındaki görevlerde de kullanılıp kullanılamayacağını görmek ilginç olacaktır. Sözgelimi, Türkçe konuştuğunuz ve öteki uçtaki kişinin söylediğiniz cümlenin İngilizce karşılığını duyduğu bir çeviri uygulamasına sahip bir akıllı telefon düşünün.

7

GELECEKTE BİZİ NE BEKLİYOR?

Onları Akıllı Yapın, Kendileri Öğrensin

Yapay öğrenme, günümüzde uygulanabilir bir teknoloji olduğunu kanıtlamıştır ve farklı alandaki uygulamaları her geçen gün artmaktadır. Bu veri toplama ve veriden öğrenme eğiliminin yakın gelecekte daha da güçlü bir şekilde devam etmesi beklenmektedir (Jordan ve Mitchell 2015). Verinin analizi, hem geçmiş verinin altında yatan süreci anlamayı –tıpkı biliminsanlarının yüzlerce yıldır farklı bilim alanlarında yaptığı gibi– hem de sürecin gelecekteki davranışını öngörmeyi sağlar.

Birkaç on yıl önce, bilgi işlem donanımı yeni mikroişlemcilerle gelişmekteydi. Her yeni mikroişlemci –ilk olarak 8, ardından 16, ardından 32 bit– birim zamanda biraz daha fazla hesaplama yapabiliyor ve biraz daha fazla bellek kullanabiliyordu, bu da biraz daha iyi bilgisayarlar demekti. Benzer şekilde, bilgisayar yazılımı da her adımda bir programlama dili düzeyinde gelişmekteydi. Her yeni dil, yeni hesaplama türlerinin programlanmasını kolaylaştırdı. Bilgisayarlar matematiksel hesaplama için kullanıldığında Fortran, ticari uygula-

malar için Cobol kullandık; daha sonra, bilgisayarlar her türden ve daha karmaşık bilgi türlerini işlemeye başladığında, daha karmaşık veri yapıları ve onları işleyen uzmanlaşmış algoritmalar tanımlamamızı sağlayan nesne yönelimli diller geliştirdik.

Daha sonra bilgi işlem, yeni işletim sistemleriyle gelişmeye başladı. Her yeni sürüm, bilgisayarların kullanımını biraz daha kolaylaştırdı ve yeni bir uygulama grubunu destekledi. Günümüzdeyse bilgi işlem, akıllı cihaz ya da akıllı uygulamalarla gelişiyor. Bir zamanlar bilgi işlemin gelişimini tanımlayan anahtar kişi donanım tasarımcısıydı, ardından yazılım mühendisi oldu, daha sonra bilgisayar başında oturan kullanıcıya dönüştü ve artık herhangi bir şey yapan herhangi bir kişidir.

Artık kimse yeni bir mikroişlemciyi hevesle beklemiyor ve ne yeni bir programlama dili ne de yeni bir işletim sistemi sürümü haber niteliği taşıyor. Artık ya tasarımcısı nedeniyle ya da akıllı olmayı öğrendiği için akıllı olan bir sonraki yeni cihazı veya uygulamayı bekliyoruz.

Yaşamlarımızın gittikçe daha fazla bölümü sayısal alana giriyor ve sonuç olarak giderek daha fazla veri yaratıyoruz. Kişisel bilgisayarlar için yapılmış en eski sabit diskler beş megabayt kapasiteye sahipti. Günümüzde sıradan bir bilgisayar, kutusundan beş yüz gigabaytla çıkıyor – bu yaklaşık otuz yıl içinde yüz bin kat daha fazla veri saklama kapasitesi demektir. Makul ölçülerde büyük bir veritabanı artık yüzlerce terabayt depoluyor ve şimdiden ölçü olarak petabaytı kullanmaya başlamış durumdayız. Çok yakında ölçü olarak, bin petabayt ya da bir milyon terabayta karşılık gelen eksabaytı kullanmaya başlayacağız. Veri saklama kapasitesindeki artışın yanı sıra, teknolojik gelişmeler sayesinde, daha hızlı bilgisayar yongaları ve aynı anda çalışırken her biri büyük bir sorunun bir parçasını çözen binlerce işlemciye sahip paralel mimarilerle veri işleme daha ucuz ve daha hızlı hale geldi.

Çok amaçlı kişisel bilgisayarlardan belli işler için özelleşmiş akıllı cihazlara geçme eğiliminin de hızlanması bekleniyor. Birinci bölümde, geçmişte kuruluşların bir bilgisayar merkezinden birbirine

bağlı birçok bilgisayar ve saklama aygıtına dağıtılmış veri işlem yapısına geçişinden bahsettik. Bugün, tekil kullanıcı açısından da benzer bir dönüşüm yaşanıyor. Bir kişinin artık tüm verilerini tutan ve tüm işlemlerini yapan tek bir kişisel bilgisayarı yok; bunun yerine kullanıcı verileri "bulut"ta, yani uzaktaki bir veri merkezinde, akıllı cihazlarından herhangi birinin ihtiyaç duyduğu kısma istediği anda erişebileceği şekilde saklanıyor.

Bulut dediğimiz şey, tüm hesaplama ihtiyaçlarımızı karşılayan sanal bir bilgisayar merkezidir. İşlemin nerede ve nasıl yapıldığına veya istediğimizde erişebildiğimiz sürece verilerin nerede ve nasıl saklandığına dair endişelenmemize gerek yoktur, tıpkı elektriğin nereden geldiğini düşünmemize gerek kalmadan istediğimiz anda televizyonumuzun fişini en yakın prize taktığımız gibi. Bu yüzden buna elektrik şebekesi [*grid*] benzetmesi yapılarak hesaplama şebekesi [*grid computing*] de denir.

Bu aynı zamanda daha büyük bant genişlikleriyle bağlanırlık anlamına gelir. Müzik ve video yayını günümüzde uygulanabilir bir teknolojidir. Bir zamanların plak ve video kasetlerinin yerini alan CD ve DVD'ler bugün işe yaramaz hale geldi ve yerlerini, *tüm* şarkılarla filmleri saklayan görünmez bir kaynağa bıraktı. E-kitap ve sayısal abonelik hizmetleri hızla basılı kitap ve kitapçının yerini alıyor; arama motorları yüzünden uzun zamandır basılı ansiklopediler kullanılmıyor.

Akıllı cihazlarla birlikte milyonlarca insanın aynı şarkının/filmin/kitabın ayrı kopyalarını saklamasına gerek kalmadı. Şimdilerde slogan şudur: "Satın almayın, kiralayın!" Akıllı cihazı, uygulamayı veya hizmet aboneliğini ve ihtiyacınız olduğunda buna erişmenize olanak sağlayan bant genişliğini satın alın.

Erişebilirlikteki bu değişim ve kolaylık, aynı zamanda ürünleri "paketlemek" ve satmak için de yeni yollar sunmaktadır. Örneğin geleneksel olarak plak ve CD, birkaç şarkıdan oluşan bir "albüme" karşılık gelirdi; artık şarkıları ayrı ayrı kiralamak da mümkün. Öykü kitabını almadan tek bir kısa öykü satın alınabilir.

Beşinci bölümde yapay öğrenmenin öneri sistemlerinde kulla-

nımını ele almıştık. Paylaşılan daha fazla veri ve yayınla, analiz edilecek daha fazla veri olacak ve ayrıca veriler daha ayrıntılı hale gelecektir. Örneğin, artık bir insanın bir şarkıyı kaç kez dinlediğini veya bir romanı kaçıncı sayfaya kadar okuduğunu biliyoruz; bu bilgiler, üründen ne kadar zevk alındığının bir ölçüsü olarak kullanılabilir.

Mobil teknolojideki gelişmeler sayesinde giyilebilir cihazlara ilgi de artmaktadır. Giyilebilir bir cihaz olan akıllı telefon artık bir telefondan çok daha fazlasıdır. Aynı zamanda saat veya gözlük gibi daha küçük akıllı "şeyleri" çevrimiçi yapmakta aracı işlevi de görür. Telefon yakın gelecekte, örneğin gerçek zamanlı çeviri yapan bir uygulamayla daha da akıllı hale gelebilir: Bir uçta kendi dilinizde konuşuyorsunuz ve diğer uçtaki kişi bunu otomatik olarak kendi diline çevrilmiş şekliyle duyuyor. Yalnızca içerik sözdizimi ve anlambilimi açısından doğru olmakla kalmıyor, aynı zamanda sizin kendi sesinizde, doğru vurgu ve tonlamayla da söyleniyor.

Yapay öğrenme, giderek daha karmaşıklaşan bir dünyayı anlamamıza yardımcı olacaktır. Günümüzde duyularımızın başa çıkabileceğinden veya beyinlerimizin işleyebileceğinden daha fazla veriye maruz kalıyoruz. Çevrimiçi olarak erişilebilen bilgi kaynakları çok büyük miktarda sayısal metin içerir ve artık elle işlenemeyecek kadar büyüktür. Yapay öğrenmeyi bu amaçla kullanmaya yapay *okuma* denir.

Sadece anahtar sözcükleri kullananlardan daha akıllı arama motorlarına ihtiyacımız var. Günümüzde farklı kaynaklarda veya ortamlarda dağıtılmış bilgilere sahibiz, bu yüzden bunların hepsinin sorgulanması ve yanıtların akıllıca birleştirilmesi gerekmektedir. Bu farklı kaynaklar farklı dillerde olabilir – örneğin İngilizce bir kaynak, sorgunuz Türkçe olsa bile konuyla ilgili daha fazla bilgi içerebilir. Bir sorgu ayrıca görüntü veya video veritabanında bir aramayı da tetikleyebilir. Yine de genel sonuç bir kullanıcı tarafından sindirilebilecek kadar özetlenmeli ve yoğunlaştırılmalıdır.

Web kazıma [*web scraping*], programların web'de otomatik olarak gezinip web sayfalarından bilgi derlemesidir. Bu web sayfaları sosyal medya olabilir ve toplanan bilgiler, örneğin seçim zamanla-

rında en fazla konuşulan konuları takip etmek veya ürünler veya insanlar hakkındaki hisleri, fikirleri ve inançları saptamak için öğrenme algoritmalarıyla analiz edilebilir. Yapay öğrenmenin sosyal medyada kullanıldığı bir diğer önemli araştırma alanı da insanların "sosyal ağ"larını saptamaktır. Bu tür ağları analiz etmek, aynı düşünceye sahip insan grupları bulmamıza veya bilgilerin sosyal medya üzerinde nasıl yayıldığını izlememize olanak tanır.

Şu anki araştırma yönelimlerinden biri de gözlük ve saat gibi geleneksel giyilebilir eşyalar dahil olmak üzere her türlü geleneksel araç ve cihaza akıl –yani veri toplama, işleme ve diğer çevrimiçi cihazlarla iletişim yeteneği– katmaktır. Daha fazla cihaz akıllı olduğunda analiz edilecek ve anlamlı çıkarımlar yapılacak daha fazla veri olacaktır. Farklı cihazlar ve sensörler o işin farklı yönlerini kaydeder ve bu durumda bu çoklu kipleri nasıl birleştireceğimiz ve bütünleştireceğimiz çok önemli olur. Bu da, öğrenme algoritmalarının kullanılabildiği farklı türlerde, yepyeni ilgi çekici senaryo ve uygulamalar demektir.

Akıllı cihazlar bize hem işte hem evde yardımcı olabilir. Yapay öğrenme, en az denetim ve en yüksek kullanıcı mutluluğu için ortamlarını öğrenebilen ve kullanıcılarına uyum sağlayabilen sistemler üretmekte bize yardımcı oluyor.

Akıllı otomobiller konusunda önemli çalışmalar yapılıyor. Çevrimiçi otomobiller (veya otobüsler, kamyonlar vd.) yolcularının çevrimiçi olmalarına izin verir ve video yayını gibi her tür hizmeti sayısal bilgi-eğlence [*infotainment*] sistemleri üzerinden sunabilir. Çevrimiçi olan otomobiller ayrıca bakım amacıyla veri alışverişinde bulunabilir ve yol ile hava koşulları hakkında gerçek zamanlı bilgilere erişebilir. Zorlu şartlar altında araç kullanıyorsanız, bir kilometre önünüzdeki başka bir otomobil, bir kilometre önünüzdeki bir sensör olur.

Ancak çevrimiçi olmaktan daha önemlisi, otomobillerin bizzat sürüşe yardım edebilecek ölçüde akıllı olmasıdır. Otomobillerde zaten hız denetimi, kendi kendine park etme ve şerit izlemeye yönelik yardım sistemleri var ve bunlar yakında daha da yetkin hale gelecek-

ler. Asıl amaç, bu sistemlerin araç sürme görevini tümüyle devralmasını sağlamaktır ve günümüzde bu amaca uygun otonom araç prototipleri üretilmektedir.

Bir sürücünün görsel sistemi çok yüksek bir çözünürlüğe sahip değildir ve sadece ileriye bakabilir. Görme alanları, yan aynalar ve dikiz aynası kullanılarak hafifçe genişletilse de kör noktalar kalır. Öte yandan, *kendi kendini süren bir otomobil* her yöne bakan daha yüksek çözünürlüklü kameralara sahip olabilir ve GPS, ultrason veya gece görüşü gibi, bir insanın sahip olmadığı sensörleri kullanabilir ya da LIDAR adı verilen ve mesafeyi ölçmek için lazer kullanan bir radarla donatılabilir. Akıllı bir otomobil, hava durumu gibi her tür bilgiye daha hızla erişebilir. Elektronik bir sürücünün tepki süresi çok daha kısadır.

Yapay öğrenme, daha yumuşak sürüş, daha hızlı kontrol ve daha yüksek yakıt verimliliği yanında yayalar, bisikletliler, trafik işaretleri vb. unsurların otomatik tanınması yoluyla akıllı algılamayı gerçekleştirerek kendi kendini süren otomobillerde önemli bir rol oynuyor. Kendi kendini süren otomobiller zaman içinde daha güvenli ve hızlı olacaktır. Yine de bazı problemler vardır: Lazerler ve kameralar zorlu hava koşullarında (yağmur, sis veya kar olduğunda) çok etkili değildir; bu yüzden teknoloji, her tür havada çalışabilen akıllı otomobiller elde edilebilmesi için gelişmelidir.

Kendi kendini süren otomobillerin ve robot taksilerin önümüzdeki on yılda şehirlerde ve otoyollarda, belki ilk başta önceden belirlenmiş şeritlerde ve daha sonra normal trafiğin bir parçası olarak sürüşü devralması bekleniyor. Ayrıca önümüzdeki on yıl içinde otomobil ve hava araçlarının kaynaşıp kendi kendini kullanan uçan otomobillerin ortaya çıkması oldukça olası görünüyor ve bunlarda da yapay öğrenmenin önemli katkı yapması bekleniyor.

Yapay öğrenme, bir işin programlanmaya gerek duymadan öğrenilebilmesi gibi temel bir üstünlüğe sahiptir. Uzay, yapay öğrenme için de yeni sınır olacaktır. Gelecekteki uzay misyonları büyük olasılıkla insansız gerçekleştirilecek. Önceleri uzaya insanları göndermemiz gerekiyordu çünkü çok akıllı ve becerikli makinelerimiz yoktu,

ancak günümüzde yetenekli robotlarımız var. Uzay aracında hiç insan yoksa yük daha hafif ve basit olacaktır ve yükü geri getirmeye de gerek kalmayacaktır. Eğer bir robot daha önce kimsenin gitmediği yere cesaretle gidecekse, bu ancak öğrenen bir robot olabilir.

Yüksek Performanslı Hesaplama

Büyük ve daha büyük veriyle birlikte, daha yüksek kapasiteli *ve* daha hızlı erişilebilen saklama sistemlerine ihtiyacımız artıyor. Daha fazla verinin makul bir sürede işlenebilmesi için işleme gücünün de birlikte artması gerekecektir. Bu da çok fazla veri saklayabilen ve çok hızlı hesaplama yapabilen yüksek performanslı bilgisayar sistemleri demektir.

Işık hızı ve atomun büyüklüğü gibi fiziksel sınırlar vardır ve bu, iletişim hızında en yüksek* ve temel elektronik aletlerin boyutunda en düşük sınırlar demektir. Bunun en açık çözümü paralel işlemedir – paralel sekiz hattınız varsa, sekiz veri öğesini aynı anda gönderebilirsiniz. Sekiz işlemciniz de varsa, bu sekiz öğeyi aynı anda, tek bir işlemi yapmak için gereken sürede işleyebilirsiniz.

Günümüzde paralel işleme, bilgisayar sistemlerinde yaygın olarak kullanılmaktadır. Eşzamanlı çalışan binlerce işlemci içeren güçlü bilgisayarlarımız var. Tek bir işlemci içinde aynı anda basit hesaplamalar yapabilen ve tek bir fiziksel yongada paralel işlem uygulayan birden fazla "çekirdekli" bilgisayarlar da var.

Fakat yüksek performanslı hesaplama sadece bir donanım sorunu değildir. Farklı uygulamaların hesaplamalarını ve verilerini çok sayıda işlemci ve saklama aygıtına verimli bir şekilde dağıtmak için iyi yazılım arayüzlerine de ihtiyaç duyuyoruz. Gerçekten de büyük

* Işık hızı yaklaşık 300.000 km/sn'dir, bu nedenle yaklaşık 1.000 km uzaklıktaki bir veri merkezine iletişim yaklaşık 3,33 milisaniye sürer. Bu aslında elektronik cihazlarda o kadar da küçük bir sayı değildir. Bağlantı hiçbir zaman doğrudan değildir ve ara yönlendirme cihazlarından dolayı her zaman gecikmeler vardır. Bir cevap almak için önce bir sorgu göndermemiz gerektiğini, bu yüzden zamanı ikiye katlamamız gerektiğini de unutmayın.

veriler için paralel ve dağıtık hesaplama yazılımları ve donanımları, günümüzde bilgisayar bilimi ve mühendisliğinde önemli araştırma alanları haline gelmiş durumda.

Öğrenme algoritmalarının paralelleştirilmesi, yapay öğrenmede giderek daha önemli hale geliyor. Farklı bilgisayarlarda ayrı modeller, verinin farklı parçaları üstünde aynı anda eğitilebilir ve daha sonra bu modeller tahmin yaparken birleştirilebilir. Diğer bir seçenek, tek bir modelin işlenmesini çoklu işlemcilere dağıtmaktır. Örneğin, çoklu katmanlarında binlerce birimden oluşan bir derin sinir ağında farklı işlemciler, farklı katmanlar veya katman altkümelerinden sorumlu olabilir ve arka arkaya gelen veriler üstünde işlem boru hattı biçiminde çok daha hızla gerçeklenebilir.

Grafik işlem birimi (GİB) [*graphical processing unit* (GPU)], başlangıçta görüntülerin hızlı işlenmesi ve grafik arayüzlerde –örneğin video oyun konsollarında– aktarılması için tasarlandı, ancak sonrasında grafik için kullanılan paralel hesaplama ve veri aktarım mimarisinin yapay öğrenme için de uygun olduğu anlaşıldı. Günümüzde bu amaç için özel yazılım kütüphaneleri geliştirilmekte ve GİB'ler, araştırmacılar ve uygulayıcılar tarafından çeşitli yapay öğrenme uygulamalarında etkin olarak kullanılmaktadır.

Yapay öğrenme uygulamalarında da *bulut hesaplamaya* yönelik bir eğilim görüyoruz; gerekli donanımı satın almak ve sürekli bakımını yapmak yerine, kullanıcılar şirket dışındaki *veri merkezlerinin* kullanımını kiralıyor. Veri merkezi, çok sayıda işlemciye ve yeterli saklama alanına sahip çok sayıda bilgisayar sunucusunu barındıran fiziksel bir yerdir. Genellikle coğrafi olarak farklı yerlerde birden fazla veri merkezi bulunur; bunların hepsi bir ağ üzerinden bağlıdır, görevler de otomatik olarak bunlara dağıtılır veya gerektiğinde birinden diğerine taşınır, böylece farklı müşterilerden farklı zamanlarda ve farklı boyutlarda gelen hesaplama yükleri dengelenir. Bu gereksinimlerin tümü, bugün dikkate değer araştırmaların yapılmasını teşvik etmektedir.

Bulutun önemli kullanımlarından biri, akıllı cihazların, özellikle de mobil cihazların yeteneklerinin artırılmasıdır. Bu düşük kapasiteli

ama çevrimiçi cihazlar veri alışverişi yapmak veya yerel olarak yapılamayacak kadar büyük veya karmaşık hesaplamaları talep etmek için buluta erişebilir. Bir akıllı telefonda konuşma tanımayı düşünelim. Telefon işitsel verileri toplar, temel özellikleri çıkarır ve buluta gönderir. Gerçek tanıma bulutta yapılır ve sonuç telefona geri gönderilir.

Hesaplama alanında iki paralel eğilim vardır. Bunlardan biri, veri merkezlerindeki sunucularda kullanılanlar gibi farklı görevler ve amaçlar için programlanabilen genel amaçlı bilgisayarlar oluşturmaktır. Diğeri de bazı uygulamalar için uygun girdi ve çıktı birimleriyle birlikte özel hesaplama aygıtları oluşturmaktır. İkinci eğilim, eskiden *gömülü sistemler* olarak adlandırılıyordu ancak günümüzde, etkileşim içinde oldukları fiziksel dünyada çalıştıkları gerçeğini vurgulamak için *siber-fiziksel sistemler* olarak adlandırılıyorlar. Bir sistem birden fazla birimden oluşabilir (bunların bazıları mobil olabilir) ve bir ağ üzerinden birbirine bağlanırlar – örneğin bir otomobil, bir uçak veya bir ev, farklı işlere yönelik olarak bu tür cihazlardan çok sayıda içerebilir. Bu sistemleri akıllı hale getirmek –diğer bir deyişle, kullanıcıyı da içeren ortamlarına uyum sağlama yeteneğini vermek– önemli bir araştırma alanıdır.

Veri Madenciliği

Yapay öğrenme, bir *veri madenciliği* çalışmasındaki en önemli adım olsa da adımlardan yalnızca biridir (Han ve Kamber 2011). Önceden verilerin hazırlanması ve daha sonra sonuçların yorumlanması gerekir.

Verinin madencilik için hazır hale getirilmesi çeşitli aşamaları içerir. İlk olarak, birçok alana sahip büyük bir veritabanından ilgilendiğimiz kısımları seçer ve çalışmak için daha küçük bir veritabanı oluştururuz. Analiz edeceğimiz veri, farklı veritabanlarından geliyor da olabilir, bu yüzden onları birleştirmemiz gerekir. Ayrıntı düzeyi de farklı olabilir. Örneğin, sürekli güncellenen bir veritabanından günlük toplamları çıkarıp tekil hareketler yerine bunları kullanabiliriz. Ham veri, hata ve tutarsızlıklar içerebilir veya eksik olabilir; bunlar da bir ön işleme aşamasında ele alınmalıdır.

Veri toplandıktan sonra, üstünde analiz yapacağımız bir veri ambarında saklanır. Veri analizi türlerinden biri, verilmiş bir hipotez için –"bira alanlar cips de alır"– verinin bu hipotezi destekleyip desteklemediğini kontrol ettiğimiz elle yapılan veri analizidir. Verimiz, satırların veri örnekleri –sepetler– ve sütunların nitelikler –ürünler– olduğu bir tablo şeklindedir. Veriyi, boyutları nitelikler olan *çok boyutlu bir veri küpü* olarak da düşünebiliriz ve veri analizi, bu küp üzerinde dilimleme, doğrama vb. operasyonları olarak tanımlanabilir. Verilerin böyle elle analizi yanında sonuçların görselleştirilmesi de çevrimiçi analitik işleme (ÇAİ) [*online analytical processing* (OLAP)] araçlarıyla kolaylaştırılmıştır.

ÇAİ, insan güdümlü olması açısından kısıtlayıcıdır ve yalnızca aklımıza gelen hipotezleri sınayabiliriz. Örneğin, sepet analizi bağlamında, çok uzak ürün çiftleri arasında bir ilişki bulamayız. Bu tür keşifler, yapay öğrenme algoritmalarının yaptığı gibi veriye dayalı bir analiz gerektirir.

Veriden bir model oluşturmak için önceki bölümlerde ele aldığımız yöntemlerden (sınıflandırma, regresyon, öbekleme vb.) herhangi birini kullanabiliriz. Genelde verimizi bir öğrenme kümesi ve bir geçerleme kümesi olarak ikiye böleriz. Modelimizi eğitmek için ilk kısmı kullanır, ardından modelin tahmin kalitesini geçerleme kümesinde ölçeriz. Eğitilmiş modeli eğitim için kullanılmayan örnekler üstünde sınayarak, daha sonra gerçek dünyada yepyeni örnekler üstünde kullanıldığında ne kadar iyi çalışacağını tahmin etmek isteriz. Geçerleme kümesinde elde edilen başarı, eğitilen modeli kabul etme veya reddetme konusundaki ana kriterlerimizden biridir.

Bazı yapay öğrenme algoritmaları bir *kara kutu modeli* öğrenir. Örneğin bir sinir ağı söz konusuysa, bir girdi verildiğinde ağ bir çıktı hesaplar, ancak ağın ara katmanlarında ne yapıldığını anlamak zordur. Öte yandan, karar ağaçlarının bulduğu eğer-ise kuralları yorumlanabilir niteliktedir ve bu kurallar, (yapay öğrenme yöntemlerinin nasıl çalıştığını bilmeseler bile) uygulamayı bilen kişiler tarafından kontrol edilip değerlendirilebilir. Çoğu veri madenciliği senaryosunda –örneğin, kredi puanlamada– bilgi çıkarma ve modelin uz-

manlarca değerlendirilme süreci önem taşıyabilir ve hatta modelin doğrulanması için şart olabilir.

Burada görselleştirme araçları da işe yarar. Görselleştirme, veri analizi için en iyi araçlardan biridir ve bazen veriyi akıllıca görselleştirmek, maliyetli istatistiksel işlemeye gerek kalmadan karmaşık bir veri kümesinin altında yatan süreci anlamak için yeterlidir; örnekler için bkz. Börner 2015.

Daha fazla veri ve bilgisayar gücüne sahip oldukça daha karmaşık senaryolardaki saklı ilişkileri keşfetmeye çalışacak daha karmaşık veri madenciliği uygulamaları üstünde çalışabiliriz. Günümüzde çoğu veri madenciliği uygulaması, tek bir veri kaynağı kullanılarak, tek bir alanda gerçekleşmektedir. Farklı kiplerdeki farklı kaynaklardan gelen verimizin olduğu durum özellikle ilginçtir. Böylesi bir veride madencilik yapmak ve kaynaklarla kipler arasında bağımlılıklar bulmak, gelecek vaat eden bir araştırma alanıdır.

Veri Gizliliği ve Güvenliği

Çok fazla veriye sahip olduğumuzda, bu verilerin analizi faydalı sonuçlara yol açabilir. Tarihsel olarak veri toplama ve analizi, tıptan astronomiye kadar birçok alanda insanlık için önemli bulgulara yol açmıştır. Sayısal teknolojinin günümüzde yaygın kullanımı, çok farklı yeni alanlarda verileri hızlı ve doğru toplayıp analiz etmemizi mümkün kılıyor.

Daha fazla ve daha ayrıntılı verinin ortaya çıkışıyla birlikte *veri gizliliği ve güvenliği* günümüzde kritik bir konu haline gelmiştir (Horvitz ve Mulligan 2015). İnsanların gizlilik kaygılarını ihlal etmeden veri topladığımızdan ve işlediğimizden, ayrıca verilerin asıl amaçları dışında kullanılmadığından nasıl emin olabiliriz?

Bir toplumdaki bireylerin sağlık ve güvenlik gibi alanlarda veri toplama ve analiz etmenin sunduğu faydaların farkında olmasını bekleriz. İnsanlar perakende alışveriş gibi daha az kritik alanlarda bile beğenilerine ve tercihlerine uyarlanmış hizmetler ve ürünler görmek ister. Yine de hiç kimse, birilerinin özel hayatına burnunu

sokmasını istemez. Örneğin akıllı cihazlarımız hayatımızla ilgili ayrıntıları kaydedip bilgimiz veya iznimiz olmadan bunları kullanılabilir kılan sayısal paparazzilere dönüşmemelidir.

Veriyi üreten kullanıcının hangi verisinin ne kadar toplandığını, hangi kısmının saklandığını, verilerin herhangi bir amaç için analiz edilip edilmeyeceğini ve varsa bu amacın ne olduğunu her zaman bilmesi temel zorunluluklardır. Şirketler, topladıkları ve analiz ettikleri veriler konusunda tümüyle şeffaf olmalıdır.

Verilerin sahipleri, hem veri toplama hem de kullanımı sırasında daima bilgilendirilmelidir. Herhangi bir analizden önce bu veriler anonimleştirilmeli, yani verideki tüm kişisel bilgiler ayıklanmalıdır. Veriyi anonimleştirmek basit bir işlem değildir. İnsan kayıtlarında, örneğin isim veya kimlik numarası gibi tekil bilgileri kaldırmak yeterli olmaz. Doğum tarihi, posta kodu vb. alanlar kısmi ipuçları sağlar ve bu bilgiler birleştirilerek bireyler saptanabilir (Sweeney 2002).

Veri o kadar değerli bir hammadde haline geliyor ki, veri toplayıcısının verinin güvenli saklanması için gerekli tüm adımları atması ve onları, veri sahibinin açık onayı olmadan başkasıyla paylaşmaması gerekiyor.

İnsanlar kendilerine ait veriler üzerinde tam kontrol sahibi olmalıdır. Kendileriyle ilgili hangi verilerin toplandığını her zaman görebilecekleri araçlara sahip olmalıdırlar. Bu verilerin düzeltilmesini veya tümüyle kaldırılmasını isteyebilmelidirler.

Yakın dönemde ortaya çıkan araştırma alanlarından biri de *gizliliği koruyan öğrenme* algoritmaları üstünedir. Diyelim ki farklı kaynaklardan veri parçalarına sahibiz (örneğin farklı ülkelerde aynı hastalığa sahip kişiler olabilir) ve bu kaynaklar, kendi verilerini (vatandaşları hakkındaki ayrıntılı bilgiyi) tüm verileri birleştiren bir modeli eğitmek üzere merkezî bir kullanıcıya vermek istemiyorlar. Böyle bir durumda olasılıklardan biri, verileri yeterince anonimleştirilmiş bir biçimde paylaşmaktır. Başka bir yol da farklı parçalarla farklı modelleri eğitip eğitilmiş modelleri paylaşmak veya ayrı olarak eğitilmiş modellerin kararlarını tahminde birleştirmektir.

Veri gizliliği ve güvenliğiyle ilgili böylesi bir endişe, herhangi bir veri analizi senaryosunun ayrılmaz bir parçası olmalı ve herhangi bir öğrenme yapılmadan önce çözülmelidir. Veri madenciliği altın madenciliği gibidir. Kazıya başlamadan önce gerekli tüm izinleri aldığınızdan emin olmanız gerekir. Gelecekte, her veri kümesinin bu sahiplik ve izin bilgileri hakkında bazı meta veriler içerdiği veri işleme standartlarımız olabilir; böylece herhangi bir yapay öğrenme veya veri analizi yazılımının bunları kontrol etmesi ve yalnızca gerekli onay damgaları varsa çalışması zorunlu kılınabilir.

Veri Bilimi

Yapay öğrenme yöntemlerinin büyük veri alanında kaydettiği ilerleme ve başarılar yanında daha fazlasının vaat edilmesi, sektördeki araştırmacı ve uygulayıcıları bu çabayı yeni bir bilim ve mühendislik dalı olarak adlandırmaya itmiştir. Bu yeni *veri bilimi* alanının neleri kapsadığı konusunda hâlâ tartışma olsa da ana konular yapay öğrenme, yüksek performanslı hesaplama ve konunun sosyal, etik ve yasal yönleri –örneğin veri gizliliği ve güvenliği– olacak gibi görünüyor.

Tabii ki tüm öğrenme uygulamalarının bir buluta, bir veri merkezine veya pahalı bilgisayar sistemlerine ihtiyacı yoktur. Şirketlerin aynı eski ürünleri pazarlamak için yeni ve süslü isimler icat etmeye dayanan satış stratejileri ve yutturmacaları karşısında kişi daima temkinli olmalıdır. Ne var ki çok fazla veri olduğunda ve süreç çok fazla hesaplama gerektirdiğinde, yapay öğrenme çözümlerinin etkin bir şekilde gerçeklenmesi önemli bir konudur.* Bununla ilgili bir diğer konu da veri analizi ve işlemenin etik ve yasal yönleridir. Örneğin daha fazla veri toplayıp analiz ettikçe çeşitli alanlardaki kararlarımız giderek daha otomatik ve veri odaklı hale gelecektir. Bu yüzden otonom işlemlerin sonuçları ve verdikleri kararlar konusunda dikkatli olmamız gerekir.

* Daha fazlası için bkz. *Frontiers in Massive Data Analysis*, National Academies Press, Washington, DC, 2013.

Gelecekte pek çok "veri bilimcisi"ne ve "veri mühendisi"ne ihtiyacımız olacak gibi görünüyor; bugün bile birçok alanda verinin ve veriden bilgi çıkarmanın öneminin gittikçe daha çok fark edildiğini görüyoruz. Bu tür senaryolar geleneksel istatistik uygulamalarından çok farklı özelliklere sahiptir.

Öncelikle, veriler artık çok daha büyüktür –bir süpermarket zincirinde yapılan tüm işlemleri düşünün. Her örnek için binlerce niteliğe sahibiz– bir gen dizisini düşünün. Veriler artık sadece rakam değil; metin, görüntü, ses, video, sıralamalar, frekanslar, gen dizileri, sensör kayıtları, tıklama günlükleri, öneri listeleri ve benzerlerinden oluşuyor. Çoğu zaman eldeki veri, istatistikte tahmin yapmayı kolaylaştırmak için kullandığımız çan şeklindeki Gauss eğrisi gibi parametrik varsayımlara uymaz. Bunun yerine, verinin altında yatan sürece göre karmaşıklığın otomatik olarak ayarlanabilen daha esnek, parametrik olmayan modellere başvurmamız gerekir. Tüm bu gereksinimler, yapay öğrenmeyi eskiden bildiğimiz ve uyguladığımız istatistikten daha zorlu kılıyor.

Eğitimde bunun anlamı şudur: İstatistik derslerini bu ek ihtiyaçları kapsayacak şekilde genişletmemiz ve iyi bilinen ancak artık yetersiz, çoğu zaman tek değişkenli (yani tek bir girdi kullanan) parametrik tahmin, hipotez sınama ve regresyon modellerinden fazlasını öğretmemiz gerekiyor. Günümüzde yüksek performanslı hesaplamanın temellerini hem donanım hem de yazılım boyutlarıyla öğretmek de gerekli hale gelmiştir, çünkü gerçek dünyadaki uygulamalarda verilerin ne kadar verimli saklandığı ve işlendiği, tahmin doğruluğu kadar kritik olabilir. Veri bilimi öğrencisinin, veri gizliliği ve güvenliğinin esaslarını da bilmesi ve veri toplama ile analizinin olası etik ve yasal sonuçlarının farkında olması gerekir.

Yapay Öğrenme, Yapay Zekâ ve Gelecek

Yapay öğrenme, yapay zekâyı elde etmenin bir yoludur. Bir veri kümesi üzerinde eğiterek veya tekrarlanan denemelerle pekiştirmeli öğrenme kullanarak bir bilgisayar programı bir başarı kriterini en

üst düzeye çıkaracak şekilde davranabilir ve bu da onun belirli bir bağlamda zeki görünmesini sağlayabilir.

Önemli noktalardan biri de zekânın muğlak bir terim olması ve bilgisayar sistemlerinin performansını değerlendirme açısından yanıltıcı olabilmesidir. Örneğin bilgisayarları satranç oynamak gibi insanlar için zor olan görevlere göre değerlendirmek, zekâlarını ölçmek için iyi bir yöntem değildir. Satranç insanlar için zor bir iştir, çünkü düşünüp taşınma ve planlama gerektirir; oysaki insanlar da diğer hayvanlar gibi sınırlı hesaplama ile sınırlı duyusal verileri kullanarak çok hızlı karar almak için evrilmiştir. Bir bilgisayar için rakibinin yüzünü tanımak, ona karşı satranç oynamaktan çok daha zordur. Bir bilgisayarın en iyi insan oyuncudan daha iyi satranç oynayıp oynayamaması, bilgisayarların daha zeki olduklarının göstergesi değildir, çünkü insan zekâsı satranç gibi işler için evrilmemiştir.

Araştırmacıların oyun oynamayı yapay zekânın deney alanı olarak kullanmasının ana nedeni, oyunların kesin kuralları olması ve kazanma ve kaybetme kriterlerinin açıkça tanımlanmış olmasıdır. Belli sayıda taş veya kart vardır ve rastsallık olsa bile biçimi iyi tanımlanmıştır: Zarlar adil olmalı, desteden kartlar aynı olasılıkla çekilmelidir; aksine girişimler hile olarak kabul edilir. Gerçek hayatta ise her çeşit rastsallık olabilir ve hayatta kalmak için her tür, öteki türlerden daha iyi bir hilekâr olmak için evriliyor.

Önemli bir sorun, bir davranışın zeki olarak kabul edilmesi için gereken başarı kriterinin ne olduğu (yani zekâyı nasıl ölçtüğümüz) ve böyle belirgin bir başarı kriterinin bulunmadığı görevlerin olup olmadığıdır. Etik ve yasal kaygıların belirli karar türlerinde rol oynadığını zaten tartışmıştık.

Verilerden eğitilen ve otonom kararlar alan bilgisayar sistemlerimiz gittikçe çoğaldığı için, bilgisayarlara çok fazla güvenme konusunda dikkatli olmamız gerekiyor. Buradaki önemli zorunluluk, yazılım sistemlerinin geçerlenmesi ve doğrulanması, yani yapmaları gereken şeyi yaptıklarından ve yapmamaları gereken şeyi yapmadıklarından emin olunmasıdır. Bu özellikle veri üstünde eğitilmiş modeller için zor olabilir, çünkü eğitim, verideki ve öğrenme sırasında-

ki her türlü rastsallıktan etkilenir. Bu, eğitilmiş yazılımı programlanan yazılımdan daha az öngörülebilir kılar. Diğer bir endişe kaynağı da verilerdeki genel davranışı öğrenen modellerin yetersiz temsil edilen vakalar veya aykırılıklar için iyi kararlar vermeyebilmesidir.

Örneğin, önerilerin geçmiş kullanım ve tercihlere çok fazla dayandırılması önemli bir risk getirir. Bir kişi yalnızca daha önce dinleyip beğendiği şarkılara benzer şarkıları dinlerse veya daha önce izleyip beğendiği filmlere benzer filmler izlerse, daha önce okuyup zevk aldığı kitaplara benzer kitapları okursa, yeni bir deneyim ortaya çıkmayacaktır ve bu, hem o kişi hem de ona yeni ürünler satmaya istekli olan şirket için sınırlayıcı olacaktır. Bu nedenle, böylesi bir öneri sisteminde her zaman bir miktar çeşitlilik sağlamak da gerekir.

Yakın tarihli bir çalışma (Bakshy, Messing ve Adamic 2015) sosyal medyadaki etkileşimler için de benzer bir riskin bulunduğunu göstermiştir. Bir kişi yalnızca hemfikir olduğu kişileri takip eder ve geçmişte okuduklarına benzer paylaşımları, mesajları ve haberleri okursa, nispeten daha geniş bir yelpazede haberler ve görüşler içeren gazete veya TV gibi geleneksel haber medyasının aksine, başkalarının görüşlerinin farkında olmayacak ve bu da deneyimlerini sınırlandıracaktır.

Zekâ bir bedene gömülüp fiziksel eylemler gerçekleştirdiğinde, davranışın doğruluğu daha kritik bir mesele haline gelir ve insan yaşamını bile tehdit edebilir. Bu endişenin gerçekleşmesi için söz konusu sistemin silahlı bir insansız hava aracı olması gerekmez; kötü bir şekilde sürülürse, otonom bir otomobil bile silah haline gelir. Bu tür kaygılar devreye girdiğinde, beklenen değer veya faydacı yaklaşımlar artık geçerli değildir, tıpkı aşağıdaki "tramvay problemi" sürümünde ele alındığı gibi.

Diyelim ki kendi kendini kullanan bir otomobilin içindesiniz ve bir çocuk aniden yola atlıyor. Varsayalım otomobil o kadar hızlı gidiyor ki, duramayacağını biliyor. Ama yine de direksiyon kırabiliyor ve çocuğa çarpmamak için sağa yönelebiliyor. Ama çocuğun annesi de yolun sağında duruyorsa otomobil nasıl karar vermeli? Devam edip çocuğa mı, sağa kırıp anneye mi çarpmalı? Böyle bir kararı nasıl

programlayabiliriz? Yoksa otomobil, sizin canınızın çocuk veya anneden daha az değerli olduğunu hesapladıktan sonra bu iki seçenek yerine sola kırıp kendisini uçurumdan aşağı mı sürmeli?

Yapay zekânın vaat ettiği güç, birçok araştırmacı için bir endişe kaynağıdır ve hiç şaşırtıcı olmayan bir şekilde bu alanda bir düzenleme çağrısı yapılmaktadır. Yakın zamanda yayımlanan bir röportajda (Bohannon 2015), önde gelen bir araştırmacı ve yapay zekâ alanındaki en önemli ders kitabının yazarlarından biri olan Stuart Russell (Russell ve Norvig 2009), sınırsız zekânın sınırsız enerji kadar tehlikeli olabileceğini ve kontrolsüz yapay zekânın en az nükleer silahlar kadar tahrip edici olabileceğini savunuyor. Buradaki zorluk, bu yeni zekâ kaynağının kötülük değil iyilik amacıyla, bir azınlığın çıkarlarını değil tüm halkın refahını artırmak ve insanlığın ortak yararı için kullanıldığından emin olmaktır.

Bazı insanlarsa fazla kötümser davranıp yapay zekâ araştırmalarının bir gün bize egemen olmak için ayaklanacak, Dr. Frankenstein'ın yaratığının elektronik benzerleri metalik canavarlara yol açabileceğinden korkuyor. Bunun gerçekleşeceğinden şüpheliyim. Ancak bugün bile otonom otomobillerden finansa kadar çeşitli uygulamalarda bizim yerimize kararlar veren, bazıları veriden eğitilmiş olan otomatik sistemlerimiz var. Süper zeki makinelerin olası isyanından dehşete düşmektense, kötü programlanmış veya iyi eğitilmemiş yazılımlardan korkmak için daha fazla nedenimiz olduğuna inanıyorum.

Kapanış Notları

Bugün büyük verimiz var ancak yarının verisi daha da büyük olacak. Sensörlerimiz daha ucuz hale geliyor ve bu nedenle daha yaygın ve daha hassas kullanılıyor. Bilgisayarlar da hesaplama güçleri bakımından büyüyorlar. Araştırmacılar grafen gibi daha fazlasını sunmayı vaat eden yeni teknolojiler ve materyaller buldukları için fiziğin dayattığı sınırlardan hâlâ uzak görünüyoruz. Yeni ürünler 3D baskı teknolojisi kullanılarak daha hızlı tasarlanıp üretilebilecek ve bu ürünlerin daha fazlasının akıllı olması gerekecektir.

Daha fazla veri ve hesaplama sayesinde, eğitilen modellerimiz daha akıllı hale gelebilir. Günümüzdeki derin ağlar yeterince derin değil; elle yazılan sayıları veya nesnelerin bir altkümesini tanımak için sınırlı bir bağlamda yeterince soyutlamayı öğrenebilseler de görme korteksimizin tanıma yeteneğine sahip olmaktan uzaklar. Büyük metin verilerinden bazı dilsel soyutlamaları öğrenebiliyoruz ancak metnin gerçekten anlaşılmasının çok uzağındayız – örneğin verilmiş bir öyküyle ilgili soruları henüz cevaplayamıyoruz. Öğrenme algoritmalarımızın nasıl ölçekleneceği bir başka açık sorudur. Örneğin derin bir ağa daha fazla katman ekleyerek ve daha fazla veriyle eğiterek görme korteksimiz kadar iyi bir model elde edebilir miyiz? Çok fazla veriyle eğitilmiş çok büyük bir ağ kullanarak bir dilden diğerine çeviri yapabilecek bir model oluşturabilir miyiz? Cevap evet olmalı, çünkü beyinlerimiz böyle modellerdir. Ancak bu ölçek yükseltme giderek zorlaşabilir. Bizler de özel bir donanımla doğmuş olsak da ilk cümlemizi söylemeden önce yıllarca çevremizi gözlemlememiz gerekiyor.

Görüntü tanımada, barkoddan optik karakter okuyucularına ve yüz tanımaya geçtikçe, her biri bir ihtiyacı çözen, her biri kendi zamanında pazarlanabilir bir ürün olan ve gittikçe karmaşık hale gelen bir dizi uygulama tanımlıyoruz. Araştırma ve geliştirmeyi teşvik eden şey bilimsel meraktan öte bu ekonomik değere dönüştürme sürecidir. Öğrenen sistemlerimiz daha akıllı oldukça, giderek daha akıllı ürün ve hizmetlerde kullanım bulacaktır.

Son elli yılda bilgisayarların hayatlarımızda yeni uygulama alanları bulduğunu ve ayrıca hesaplamayı kolaylaştırmak için hayatlarımızı değiştirdiğini gördük. Benzer şekilde, cihazlarımız daha akıllı hale geldikçe, içinde yaşadığımız ortam ve bu ortamda sürdürdüğümüz yaşamlar da değişecektir. Her çağın teknolojisi, kısıtlamalarıyla bir ortam tanımlar ve bu kısıtlamalar, sonrasında yeni icatların ve yeni teknolojilerin ortaya çıkışını körükler. İki bin yıl geriye gidip Romalılara cep telefonu teknolojisini verebilseydik hâlâ ata bindikleri için, yani hayatlarının geri kalanıyla uyumlu olmadığı için yaşam kaliteleri artar mıydı, emin değilim. Makinelerde insan düzeyinde zekâya ihtiyaç duyacağımız dünya çok farklı bir dünya olacaktır.

Bu zekâ düzeyine ne zaman varacağımız, bunun için ne kadar hesaplama ve eğitim gerekeceği henüz bilinmiyor. Şu anda yapay öğrenme bunu elde etmenin en umut verici yolu gibi görünüyor, bu yüzden bizi izlemeye devam edin.

SÖZLÜKÇE

Akıllı aygıt [*Smart device*]

Veriyi algılayıp sayısal olarak gösteren ve bu veri üstünde bir dizi hesaplama yapabilen aygıt. Mobil ve/veya çevrimiçi de olabilir, yani başka akıllı aygıtlar, bilgisayarlar veya bulutla veri iletişimi yapabilir.

Algılayıcı [*Perceptron*]

Algılayıcı, katmanlar halinde düzenlenmiş bir *sinir ağı* türüdür. Her katman, bir önceki katmanın birimlerinden bağlantılar alır, hesaplama yapar ve çıktısını bir sonraki katmana aktarır.

Anonimleştirme [*Anonymization*]

Kaydın kaynağının, örneğin ait olduğu kişinin tanınmasını imkânsız kılacak biçimde bilginin ayıklanması ya da saklanması. Düşünüldüğü kadar kolay değildir.

Ayırtaç [*Discriminant*]

Bir örneğin bir *sınıfın* öğesi olmasının koşullarını tanımlayan ve böylece onu başka sınıfların örneklerinden ayırt eden kural.

Aykırılık saptama [*Outlier detection*]

Bir aykırılık, anomali ya da yenilik, örneklemdeki diğer örneklerden oldukça farklı olan bir örnektir. Sahtekârlık saptama gibi bazı uygulamalarda genel kurallara istisna oluşturan böylesi aykırılıkları bulmak isteriz.

Bağlantısalcılık [*Connectionism*]

Bilişsel bilimde, zihnin paralel olarak çalışan çok sayıda temel işlemciden oluşan ağın işleminin sonucu olarak ortaya çıktığını savunan yaklaşım. *Paralel dağıtık işleme* olarak da bilinir.

Bayes ağı [*Bayesian network*]

Bkz. çizgesel model.

Bayes kuralı [*Bayes' rule*]

Olasılık teorisinin temel dayanaklarından biridir ve bağımsız olmayan iki ya da daha fazla rasgele değişkende bir yöndeki koşullu olasılığı öteki yöndeki koşullu olasılık cinsinden yazmamızı sağlar:

$$P(B|A) = P(A|B)\,P(B) / P(A).$$

Örneğin, B, A'nın nedeni ve bize $P(A|B)$ verilmiş olsun. $P(B|A)$'yı hesaplamak bir tanı yapmayı, yani A sonuç semptomları verildiğinde B nedeninin olasılığının hesaplanmasını sağlar.

Bayes'çi kestirim [*Bayesian estimation*]

Yalnız örneklem verisini değil, bilinmeyen parametreler hakkındaki bir *önsel dağılım* şeklinde verilen önsel bilgiyi de kullandığımız bir parametre kestirim yöntemi. Bu önsel dağılım, Bayes kuralını kullanarak verilerdeki bilgiyle birleştirilir ve *sonsal dağılımı* ortaya çıkar.

Belge sınıflandırma [*Document categorization*]

Metin belgelerini, genellikle metinde görülen sözcüklere dayanarak (örneğin *sözcük torbası gösterimi* kullanarak) sınıflandırma. Sözgelimi, haber belgeleri, politika, sanat, spor vb, ve e-postalar, *istenmeyen* ya da *geçerli* olarak sınıflandırılabilir.

Bilgi çıkarma [*Knowledge extraction*]

Bazı uygulamalarda, özellikle veri madenciliğinde, bir modeli eğittikten sonra modelin neyi öğrendiğini anlayabilmek isteriz. Bu, uygulamanın uzmanı olan insanların modeli doğrulaması için kullanılabilir, ayrıca verinin üretildiği süreci anlamamıza yardımcı olur. Bazı modeller, içinde yapılan hesabı anlamanın zor olduğu "kara kutu"lardır; doğrusal modeller ve karar ağaçları gibi bazı modellerse yorumlanabilir ve çalışılmış bir modelden bilgi çıkartılmasını mümkün kılar.

Biyoenformatik [*Bioinformatics*]

Biyolojik veriyi analiz etmek ve işlemek için yapay öğrenme kullanan yöntemleri de içeren hesaplama yöntemleri.

Biyometri [*Biometrics*]

İnsanların fizyolojik özelliklerini (örneğin yüz, parmak izi) ve davranışsal özelliklerini (örneğin imza, yürüyüş biçimi) kullanarak tanınması ya da onaylanması.

Boyut indirme [*Dimensionality reduction*]

Girdi niteliklerinin sayısını azaltma yöntemleri. Bir uygulamada girdilerden bazıları bilgilendirici olmayabilir ve bazıları da aynı bilgiyi vermenin farklı yollarına karşılık gelebilir. Girdi sayısını azaltmak aynı zamanda öğrenilen *modelin* karmaşıklığını azaltır ve yapılacak hesaplamayı basitleştirir. Bkz. *özellik seçimi* ve *özellik çıkartma*.

Bulut hesaplama [*Cloud computing*]

Bilgi işlemde, verilerin ve hesaplamanın yerel yapılmayıp uzak veri merkezlerinde gerçeklendiği yeni bir paradigma. Genellikle çok sayıda veri merkezi vardır ve farklı

kullanıcıların görevleri kullanıcının göremeyeceği bir biçimde bunlara dağıtılır. Buna önceden hesaplama şebekesi deniyordu.

Çevrimiçi analitik işleme (ÇAİ) [*Online analytical processing* (OLAP)]

Bir *veri ambarından* bilgi çıkarmak için kullanılan veri analizi yazılımı. ÇAİ, kullanıcı odaklıdır: Kullanıcı süreç/işlem hakkında bazı hipotezler düşünür ve ÇAİ araçlarını kullanarak verinin bu hipotezleri destekleyip desteklemediğini kontrol eder. Yapay öğrenme ise otomatik veri analizi sayesinde daha önce kullanıcıların düşünmediği bağımlılıkları bulabilmesini sağladığı için daha veri odaklıdır.

Çizgesel model [*Graphical model*]

Olasılıksal kavramlar arasındaki bağımlılıkları temsil eden bir model. Her bir düğüm farklı doğruluk derecesine sahip bir kavrama karşılık gelir ve düğümler arasındaki bağlantılar, koşullu bağımlılıkları temsil eder. Eğer yağmurun çimlerimin ıslanmasına neden olduğunu biliyorsam bir düğümü yağmur, bir düğümü ıslak çimen için tanımlar ve yağmur düğümünden ıslak çimen düğümüne doğrudan bir bağlantı yerleştiririz. Böyle ağlardaki olasılıksal çıkarım, bilinen etkili çizge algoritmaları olarak gerçeklenebilir. Böylesi ağlar bir görsel gösterimdir ve anlamaya yardım eder. Ayrıca *Bayes ağı* olarak bilinirler, çünkü böylesi ağlarda kullanılan bir çıkarım kuralı, *Bayes kuralıdır*.

Derin öğrenme [*Deep learning*]

Modelleri ham girdilerden çıktılara kadar çeşitli soyutlama düzeylerinde eğitmek amacıyla kullanılan yöntemler. Sözgelimi, görsel tanımada en düşük düzey piksellerden oluşan bir görüntüdür. Katmanlarda yukarı çıktıkça bir derin öğrenici bunları farklı çizgileri ve kenarları oluşturmak üzere birleştirir. Ardından bunlar da daha uzun çizgilerin, yayların, köşelerin ve kesişimlerin tespit edilmesi için, bir sonraki katman bunlardan dikdörtgenlerin, çemberlerin vs. oluşturulması için birleştirilebilir. Her bir katmanın birimleri, farklı bir soyutlama düzeyindeki temel öğeler olarak düşünülebilir.

Doğal dil işleme [*Natural language processing*]

İnsan dilini işlemek için kullanılan bilgi işlem yöntemleri. Aynı zamanda hesaplamalı dilbilim olarak da bilinir.

Eğer-ise kuralları [*If-then rules*]

"EĞER [IF] öncül İSE [THEN] sonuç" biçiminde yazılan karar kuralları. Öncül bir mantıksal koşuldur ve verilmiş girdi için doğruysa, sonuçtaki eylem uygulanır. Bir kural tabanı birçok eğer-ise kuralından oluşur. Eğer-ise kuralları kümesi olarak yazılabilen modeli anlamak kolaydır ve bu yüzden kural tabanları *bilgi çıkarmayı sağlar*.

En yakın komşu yöntemleri [*Nearest-neighbor methods*]

Bir örneği, ona en benzer öğrenme örneklerine göre yorumladığımız modeller. Benzer

girdilerin benzer çıktıları vardır, varsayımını kullanırlar. Bunlara bazen örnek, bellek veya vaka temelli yöntemler de denir.

Geçerleme [*Validation*]

Eğitilmiş bir modelin *genelleştirme* başarısını o modelin eğitimi sırasında görülmemiş veriler üstünde sınamak. Genellikle yapay öğrenmede verimizin bir kısmını geçerleme verisi olarak dışarıda tutar, öğrenme kümesi üstünde eğitim yaptıktan sonra modeli bu dışarıda bırakılmış veriyle sınarız. Bu sağlamanın doğruluğu, modeli gerçek hayatta kullandığımızda ne kadar başarılı çalışmasını bekleyebileceğimizi gösterir.

Genelleştirme [*Generalization*]

Bir öğrenme kümesinde eğitilmiş modelin, öğrenme sırasında görülmemiş yeni veriler üstünde gösterdiği başarı. Bu, yapay öğrenmenin merkezinde yer alır. Bir sınavda, öğretmen dersi verirken çözdüğü alıştırmalardan farklı sorular sorar ve öğrencinin başarısı bu yeni sorulardaki başarısıyla ölçülür. Yalnızca öğretmenin sınıfta çözdüğü soruları çözebilen bir öğrenci yeterince iyi değildir.

Gömülü sistemler [*Embedded systems*]

Bkz. *siber-fiziksel sistemler*.

Gözetimli öğrenme [*Supervised learning*]

Modelin herhangi bir girdi için doğru çıktıyı üretmeyi öğrendiği bir yapay öğrenme türü. Model, örnek girdiler için istenen çıktıyı bilen bir gözetmen tarafından hazırlanmış veriyle eğitilir. *Sınıflandırma* ve *regresyon*, gözetimli öğrenme örnekleridir.

Hata geri yayma [*Error Backpropagation*]

Yapay sinir ağlarında *gözetimli öğrenme* amacıyla kullanılan bir öğrenme algoritması. Burada bağlantı ağırlıkları, çıkış birimlerindeki hatayı azaltmak için güncellenir.

İlişkilendirme kuralları [*Association rules*]

Sepet analizinde iki ya da daha fazla ürünü ilişkilendiren *eğer-ise kuralları*. Örneğin, "Bebek bezi alan insanlar sıklıkla bira da alırlar."

Karakter tanıma [*Character recognition*]

Basılı ya da elle yazılmış karakterlerin tanınması. Optik tanımada girdiler görseldir ve bir kamera ya da tarayıcı tarafından algılanır. Kalem tabanlı tanımada yazı dokunmaya hassas bir yüzey üstünde yazılır ve girdi, kalemin ucunun zaman içinde izlediği koordinatların dizisidir.

Karar ağacı [*Decision tree*]

Karar düğümleri ve dallarından oluşan hiyerarşik bir model. Karar ağacı hızlı çalışır ve bir dizi *eğer-ise kuralına* dönüştürülebilir. Bu da *bilgi çıkarmayı* mümkün kılar.

Konuşma tanıma [*Speech recognition*]

Bir mikrofonun algıladığı akustik bilgiden seslendirilen cümlenin tanınması.

Kötü konumlanmış problem [*Ill-posed problem*]

Verinin tek bir çözüm bulmak için yeterli olmadığı bir problem. Veriye model oturtmak kötü tanımlanmıştır, yani tek bir sonuç model elde etmek için ek varsayımlarda bulunmamız gerekir. Böylesi varsayımlara bir öğrenen algoritmanın öğrenme yanlılığı denir.

Model

Girdi ile çıktı arasındaki ilişkiyi biçimselleştiren şablon. Yapısı sabittir ama değiştirilebilir parametreleri de vardır. Model geneldir, aynı model parametreleri uygun biçimde ayarlayarak farklı işlerde kullanılabilir. Öğrenme sırasında modelin parametreleri veriye bağlı olarak verinin tanımladığı iş için ayarlanır.

Occam'ın usturası [*Occam's razor*]

Basit açıklamaları karmaşık olanlara tercih etmemizi tavsiye eden felsefi bir sezgisel yaklaşım.

Öbekleme [*Clustering*]

Benzer örneklerin kümeler şeklinde gruplandırılması. Bu bir *gözetimsiz öğrenme* yöntemidir çünkü bir kümeyi oluşturan örnekler girdi benzerliklerine dayanılarak bulunur. Bu da bir gözetmenin örnekleri açıkça etiketleyerek sınıflara atadığı bir *sınıflandırma* görevinden farklıdır.

Önsel dağılım [*Prior distribution*]

Veriye bakmadan *önce* bilinmeyen bir parametrenin alabileceği olası değerlerin dağılımı. Örneğin lise öğrencilerinin ağırlıklarının ortalaması hakkında tahminde bulunmadan önce bu ortalamanın 45 ile 90 kg arasında olacağı üstüne önsel bir inancımız olabilir. Böyle bir bilgi, özellikle az verimiz olduğunda kullanışlıdır.

Örneklem [*Sample*]

Gözlemlenmiş bir veri seti. İstatistikte ana kitle ile örneklem arasındaki fark önemlidir. Diyelim ki lise öğrencilerinde obezite üzerine çalışmak istiyoruz. Ana kitle lise öğrencilerinin tümüdür ve bunların hepsinin ağırlıklarını gözlemlememiz mümkün değildir. Bunun yerine, rasgele örneğin 1000 öğrenciden oluşan bir altküme seçer ve onların ağırlıklarını gözlemleriz. Bu 1000 değer bizim örneklemimizi oluşturur. Ana kitle hakkında çıkarımlar yapmak için örneklemi analiz ederiz. Örneklemden hesapladığımız herhangi bir değer bir istatistiktir. Sözgelimi, örneklemdeki 1000 öğrencinin ağırlıklarının ortalaması bir istatistiktir ve ana kitlenin ortalaması için bir kestirimdir.

Örtük anlamsal analiz [*Latent semantic analysis*]

Gözlenen büyük miktardaki örnek verideki bağımlılıkları açıklayan az sayıda saklı değişkeni bulmayı amaçlayan öğrenme yöntemi. Böyle saklı değişkenler, soyut (örneğin anlamsal) kavramlara karşılık geliyor olabilir. Sözgelimi, her bir haber makalesinin bir dizi "konu" içerdiği söylenebilir. Bu konu bilgisi, veride gözetimli bir yolla açıkça verilmemesine rağmen bunları veriden öğrenebiliriz, öyle ki her konu belirli bir sözcük kümesiyle, her bir haber makalesi ise belirli bir konu kümesiyle tanımlanır.

Örüntü tanıma [*Pattern recognition*]

Bir örüntü, verideki belli bir düzendir. Örneğin "A" harfi, belli açı ve uzunluktaki üç çizginin özel bir bileşimidir. Örüntü tanıma, böyle örüntülerin tespit edilmesidir.

Özellik çıkartma [*Feature extraction*]

Bir *boyut indirme* yöntemi olarak asıl girdiler birleştirilerek yeni, daha bilgilendirici özellikler hesaplanır.

Özellik seçimi [*Feature selection*]

Bilgilendirici olmayan özellikleri atıp yalnızca bilgilendirici olanları tutan bir yöntem. *Boyut indirmenin* başka bir yöntemidir.

Özkodlayıcı ağ [*Autoencoder network*]

Çıktılarında girdilerini yeniden oluşturmak üzere eğitilen bir tür *sinir ağı*. Girdilerden daha az sayıda saklı birim bulunduğundan, ağ gizli birimlerinde kısa bir gösterim öğrenmeye zorlanır, bu da bir soyutlama işlemi olarak yorumlanabilir.

Paralel dağıtık işleme [*Parallel distributed processing*]

Görevin daha küçük, her biri farklı bir işlemcide çalıştırılan koşut görevlere bölündüğü bir bilgisayar paradigması. Daha fazla işlemci kullanarak toplam bilgi işlem zamanı düşürülür.

Parametrik olmayan yöntemler [*Nonparametric methods*]

Verinin özellikleri hakkında güçlü varsayımlarda bulunmayan istatistiksel yöntemler. Bu sayede daha esnektirler ancak verileri yeterince kısıtlamak için daha fazla veriye gerek duyabilirler.

Parametrik yöntemler [*Parametric methods*]

Veri hakkında güçlü varsayımlarda bulunan istatistiksel yöntemler. Bu yöntemlerin güzel yanı, varsayım uygunsa hesaplama ve veri bakımından oldukça etkili olabilmeleridir. Riski ise bu varsayımların her zaman geçerli olmamasıdır.

Pekiştirmeli öğrenme [*Reinforcement learning*]

Eleştirmenle öğrenme olarak da bilinir. Etmen bir dizi eylemde bulunur ve en sonda bir ödül/ceza alır, ara eylemler için bir geri besleme yoktur. Etmen bu sınırlı bilgiyi kullanarak sonraki denemelerde en yüksek ödülü getirecek eylemleri üretmeyi öğrenmelidir. Örneğin satrançta bir dizi hamle yaparız ve en sonunda oyunu kazanır ya da kaybederiz. Ardından bize bu sonucu getiren eylemlerin hangileri olduğunu anlamamız ve sonuca göre onları tekrarlama olasılığını artırmamız ya da azaltmamız gerekir.

Q öğrenme [*Q learning*]

Durumlardaki eylemlerin iyilik değerlerinin genellikle Q ile gösterilen bir tabloda (ya da işlevde) saklandığı, *zamansal fark öğrenmeye* dayanan pekiştirmeli öğrenme yöntemi.

Regresyon [*Regression*]

Verilmiş bir girdi için sayısal bir değer tahmininde bulunmak. Örneğin özellikleri verildiğinde kullanılmış bir otomobilin fiyatını tahmin etmek bir regresyon problemidir.

Sepet analizi [*Basket analysis*]

Bir sepet (örneğin bir süpermarkette) birlikte satın alınan bir küme üründür. Sepet analizi aynı sepette sık sık birlikte bulunan ürünleri bulmayı amaçlar. Ürünler arasındaki böyle bağımlılıklar *ilişkilendirme kurallarıyla* gösterilir.

Sınıf [*Class*]

Aynı kimliğe sahip örneklerin kümesi. Örneğin, "A" ile "*A*" aynı sınıfa aittir. Yapay öğrenmede her bir sınıf için o sınıfın örnekler kümesini tanımlayan bir *ayırtaç* öğreniriz.

Sınıflandırma [*Classification*]

Verili bir örneği *sınıflardan* birine atamak.

Sıralama [*Ranking*]

Bir açıdan regresyona benzer bir uygulama türüdür ancak burada yalnızca çıktıların doğru sırada olup olmadığına bakarız. Örneğin *A* ve *B* gibi iki film için, eğer kullanıcı *B*'ye göre *A*'dan daha çok keyif almışsa, *A* için kestirilen puanın *B* için kestirilenden daha yüksek olmasını isteriz. Regresyonda olduğu gibi mutlak puan değerleri yoktur, yalnızca göreli değerler üzerinde bir kısıtlama vardır.

Siber-fiziksel sistemler [*Cyber-physical systems*]

Doğrudan fiziksel dünyayla etkileşen akıllı cihazlar. Bazıları mobil de olabilir. Görevi işbirlikçi bir biçimde yerine getiren bir ağ olarak tasarlanabilirler. *Gömülü sistemler* olarak da bilinirler.

Sinir ağı [*Neural network*]

Nöron denilen temel işleme birimleri ve sinaps denilen nöronlar arası bağlantıların bir ağından oluşan model. Her bir sinapsın yönü ve ağırlığı vardır. Ağırlık, önceki nöronun sonraki nöronun etkilenim değeri üzerindeki etkisini tanımlar.

Sonsal dağılım [*Posterior distribution*]

Veriye baktıktan *sonra* bilinmeyen bir parametrenin olası değerlerinin dağılımı. Bayes kuralı, *önsel dağılım* ile veriyi birleştirerek sonsal dağılımı hesaplamamızı sağlar.

Sözcük torbası [*Bag of words*]

N sözcük için önceden bir sözlük seçtiğimiz ve her bir belgeyi, eğer belgede i sözcüğü varsa i. öğesi 1, yoksa 0 olan N uzunluğunda bir listeyle temsil ettiğimiz belge gösterimi yöntemi.

Üretici model [*Generative model*]

Verinin nasıl üretildiği hakkındaki inancımızı gösterecek şekilde tanımlanan model. Veriyi üreten saklı nedenleri ve onlardan da daha yüksek düzeydeki saklı nedenleri düşünürüz. Kaygan yollar kazalara, yağmur da yolların kaygan olmasına neden olabilir.

Veri ambarı [*Data warehouse*]

Belli bir veri analizi uygulaması için seçilmiş, çıkartılmış ve düzenlenmiş bir veri altkümesi. Asıl veri çok ayrıntılı olabilir ve farklı gündelik veritabanlarından geliyor olabilir. Ambar onları birleştirir ve özetler. Ambar salt okunurdur; verinin altında yatan sürecin, çevrimiçi analitik işleme ve görselleştirme araçlarıyla ya da *veri madenciliği* yazılımıyla incelenmesi için kullanılır.

Veri analizi [*Data analysis*]

Büyük miktarda veriden bilgi çıkarmaya yarayan hesaplama yöntemleri. *Veri madenciliği*, yapay öğrenme kullanır ve veri odaklıdır; çevrimiçi analitik işleme ise kullanıcı odaklıdır.

Veri bilimi [*Data science*]

Bilgisayar bilimleri ve mühendisliğinde, yapay öğrenme, yüksek performanslı hesaplama ve veri gizliliği/güvenliğinden oluşan, yeni önerilen bir alan. Veri bilimi bugün çok farklı senaryolarda karşımıza çıkan "büyük veri" sorunlarını sistematik bir biçimde ele almak için öne sürülmüştür.

Veri madenciliği [*Data mining*]

Büyük miktarda veriden bilgi çıkarmaya yarayan yapay öğrenme ve istatistik yöntemleri. Örneğin *sepet analizi*nde çok sayıda işlemi analiz ederek *ilişkilendirme kurallarını* buluruz.

Veritabanı [*Database*]

Sayısal olarak temsil edilen bilgiyi etkili saklamak ve işlemek için geliştirilmiş bir yazılım türü.

Web kazıma [*Web scraping*]

Web üzerinde otomatik olarak gezinip web sayfalarından bilgi derleyen yazılım.

Yapay zekâ [*Artificial intelligence*]

Bilgisayarların, eğer insanlar yapsaydı "zekâ" gerektirdiği söylenebilecek şeyler yapmak üzere programlanması. İnsan merkezci ve muğlak bir terimdir: Bilgisayarların "yapay olarak zeki" olduğunu söylemek, araç sürmenin "yapay olarak koşmak" olduğunu söylemeye benzer.

Yüksek performanslı hesaplama [*High-performance computing*]

Bugün sahip olduğumuz büyük veri sorunlarıyla makul bir sürede baş etmek için hem saklamada hem de hesaplamada güçlü bilgisayar sistemlerine ihtiyaç duyuyoruz. Yüksek performanslı hesaplama alanı bu kapsamda yapılan çalışma ve araştırmaları içerir. Yaklaşımlardan biri *bulut hesaplamadır*.

Yüz tanıma [*Face recognition*]

İnsanların kimliklerinin bir kamera tarafından alınan yüz görüntülerinden tanınması.

Zamansal fark öğrenme [*Temporal difference learning*]

O anki eylemin iyiliğini, hemen ardından gelen eylemdeki ödülden geriye taşıyarak hesaplayan pekiştirmeli öğrenme yöntemleri. Bir örneği *Q öğrenme* algoritmasıdır.

İLERİ OKUMALAR

Duda, R.O., P.E. Hart ve D.G. Stork, *Pattern Classification*, Wiley, New York, 2001.

Feldman, J.A., *From Molecule to Metaphor: A Neural Theory of Language*, MIT Press, Cambridge, MA, 2006.

Hastie, T., R. Tibshirani ve J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York, Springer, 2011.

Kohonen, T., *Self-Organizing Maps*, Springer, Berlin, 1995.

Manning, C.D. ve H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, 1999.

Murphy, K., *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, MA, 2012.

Pearl, J., *Causality: Models, Reasoning, and Inference*, Cambridge University Press, Cambridge, UK, 2000.

Witten, I. H. ve E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, CA, 2005.

KAYNAKÇA

Ballard, D.H., *An Introduction to Natural Computation*, MIT Press, Cambridge, MA, 1997.

Bakshy, E., S. Messing, ve L.A. Adamic, "Exposure to Ideologically Diverse News and Opinion on Facebook", *Science* 348, 2015, s. 1130-1132.

Bohannon, J., "Fears of an AI Pioneer", *Science* 349, 2015, s. 252.

Börner, K., *Atlas of Knowledge: Anyone Can Map*, MIT Press, Cambridge, MA, 2015.

Blei, D., "Probabilistic Topic Models", *Communications of the ACM* 55, 2012, s. 77-84.

Buchanan, B.G. ve E.H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison Wesley, Reading, MA, 1984.

Feldman, J.A. ve D.H. Ballard, "Connectionist Models and Their Properties", *Cognitive Science* 6, 1982, s. 205-254.

Fukushima, K., "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position", *Biological Cybernetics* 36, 1980, s. 93-202.

Han, J. ve M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco, CA, 2011.

Hebb, D.O., *The Organization of Behavior*, Wiley & Sons, New York, 1949.

Hirschberg, J. ve C.D. Manning, "Advances in Natural Language Processing", *Science* 349, 2015, s. 261-266.

Horvitz, E. ve D. Mulligan, "Data, Privacy, and the Greater Good", *Science* 349, 2015, s. 253-255.

Hubel, D. H., *Eye, Brain, and Vision*, W. H. Freeman, New York, 1995. http://hubel.med.harvard.edu/index.html.

Jordan, M. I. ve T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospects", *Science* 349, 2015, s. 255-260.

Koller, D. ve N. Friedman, *Probabilistic Graphical Models*, MIT Press, Cambridge, MA, 2009.

LeCun, Y., Y. Bengio ve G. Hinton, "Deep Learning", *Nature* 521, 2015, s. 436-444.

LeCun, Y., B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard ve L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation* 1, 1989, s. 541-551.

Liu, T.-Y, *Learning to Rank for Information Retrieval*, Springer, Heidelberg, 2011.

Marr, D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, MIT Press, Cambridge, MA, 1982.

McCulloch, W. ve W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics* 5, 1943, s. 115-133.

Minsky, M.L. ve S.A. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

Mitchell, T., *Machine Learning*, McGraw Hill, New York, 1997.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M.G. Bellemare, A. Graves, vd., "Human-Level Control through Deep Reinforcement Learning", *Nature* 518, 2015, s. 529-533.

Nilsson, N.J., *The Quest for Artificial Intelligence: History of Ideas and Achievements*, Cambridge University Press, Cambridge, UK, 2009.

Orbanz, P. ve Y.W. Teh, "Bayesian Nonparametric Models", *Encyclopedia of Machine Learning*, Springer, New York, 2010.

Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, Washington, DC, 1962.

Rumelhart, D.E., G.E. Hinton ve R.J. Williams, "Learning Representations by Back-Propagating Errors", *Nature* 323, 1986, s. 533-536.

Rumelhart, D.E. ve J.L. McClelland ve PDP Araştırma Grubu, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 1986.

Russell, S. ve P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Upper Saddle River, NJ, 2009.

Sandel, M., *What Money Can't Buy: The Moral Limits of Markets*, Straus and Giroux, New York, Farrar, 2012.

Schmidhuber, J., "Deep Learning in Neural Networks: An Overview", *Neural Networks* 61, 2015, s. 85-117.

Silver, D., A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser vd., "Mastering the Game of Go with Deep Neural Networks and Tree Search", *Nature* 529, 2016, s. 484-489.

Sutton, R.S. ve A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

Sweeney, L., "K-Anonymity: A Model for Protecting Privacy", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 2002, s. 557-570.

Tesauro, G., "Temporal Difference Learning and TD-Gammon", *Communications of the ACM* 38, 1995, s. 58-68.

Thrun, S., W. Burgard ve D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, MA, 2005.

Valiant, L., "A Theory of the Learnable", *Communications of the ACM* 27, 1984, s. 1134-1142.

Vapnik, V., *Statistical Learning Theory*, Wiley, New York, 1998.

Winston, P.H., "Learning Structural Descriptions from Examples", *The Psychology of Computer Vision*, P. H. Winston (yay. haz.), McGraw-Hill, New York, 1975, s. 157-209.

Wirth, N., *Algorithms + Data Structures = Programs*, Prentice Hall, Upper Saddle River, NJ, 1976.

Zadeh, L.A., "Fuzzy Sets", *Information and Control* 8, 1965, s. 338-353.

DİZİN

A

akıllı telefon 21, 116
aktif öğrenme 72
algılayıcı 78
algoritma 27, 30
AlphaGo 110
analiz düzeyleri 30
Anna Karenina 66
anonimleştirme 124
arama motoru 115
ayırtaç 35, 48
aykırılık 68, 94
  tespiti 66

B

bağlantısal model 83
barkod 55
Bayes
  ağları 61
  kestirimi 74
  kuralı 61
beklenen değer 53, 107
belge
  modelleme 101
  öbekleme 95
  sınıflandırma 64
belirsizlik 39, 68, 107, 109
Bellman denklemi 108
beyin 77
bilgi
  çıkarma 34, 50, 122, 126
  gösterimi 18
  iletişimi 20
  saklama 19
  tabanı 51
bilgisayar ağları 20
bilişsel bilim 29, 83
bilişsel psikoloji 83
bit 18
biyoenformatik 95
biyometri 63
boyut indirme 68, 87
bulanık mantık 51
bulut 115
bulut hesaplama 120
büyükanne hücre teorisi 84
büyük veri 89, 119, 120, 125, 129

C

captcha 57
çekirdek fonksiyonu 97
çeviri 65, 91, 111, 116, 130
çevrimiçi
  bağlanırlık 21
  öğrenme 80
çıkarım 35
çıkarım motoru 51
çizgesel model 60
çoğulortam 26
çok katmanlı algılayıcı 78

D

dağıtık gösterim 84, 99
değişmezlik 59
derin

öğrenme 90, 110
sinir ağı 89, 120
Derin Q Ağı 110
destek vektör makinesi 97
dışlayıcı VEYA (XOR) 81
dilbilim 64
dil modeli 64
doğal dil işleme 64, 91
doğrusal
ayırtaç analizi 69
model 42, 78
dolandırıcılık 63, 67
donanım 30
duygulanımsal bilgi işlem 62
düzenlileştirme 75
düzgünlük 44

E
eğer-ise kuralı 49, 51, 70, 122
en yakın k komşu kestirimi 58
e-posta 20
filtreleme 65
istenmeyen 27
etkilenim 78
etmen 104
evrim 28, 127
evrişimli sinir ağı 87, 110
ezberleme 44

F
Fibonacci dizisi 46
fonem (ses parçası) 63
Frisch, M. 53

G
genelleme 44
girdi gösterimi 38
gizli model 59
Go oyunu 110
görüntü tanıma 55
gösterim 30
dağıtık 84
yerel 84
gözetimli öğrenme 43, 48
gözetimsiz öğrenme 93
gözetmen 43
grafik işlem birimi 120
grafik kullanıcı arayüzleri 19

H
hata
fonksiyonu 80
geri yayma 82
Hebb'ci öğrenme 79
hesaplamalı öğrenme teorisi 72
hesaplama teorisi 30, 77
hiyerarşik
koni 87
öbekleme 97
hizalama 95
Homo Faber 53

İ
ilişkilendirme kuralları 98
internet 20
istatistik 35
izgesel öbekleme 97

K
kalem tabanlı karakter tanıma 59
Kanizsa üçgeni 45
kara kutu modeli 122
karar ağacı 70, 122
kestirim 35
en yakın k komşu 58
parametrik 57
parametrik olmayan 58
yarı parametrik 57
keşifçi veri analizi 96
kısa süreli bellek 82

kişisel bilgisayar 19, 114
K kollu kumar makinesi 105
kodlama 69, 92
konuşma tanıma 63, 96, 97, 121
kredi puanlama 47, 69, 122
kumar 40
kural tabanı 50

L
Lisp 51

M
Marr, D. 30
matris ayrışımı 99
mikroişlemci 19, 113
mobil bilgi işlem 21
model 33
  birleştirme 66
  çizgesel 60
  doğrusal 42
  gizli 59
  seçimi 43
  uydurma 46
  üretici 59
motif 95
müşteri
  davranışı 25
  segmentasyonu 94

N
nedensellik 60
neocognitron 87
NeuroGammon 110

O-Ö
Occam'ın usturası 46
olasılık 39
olası yaklaşık doğru öğrenme 72
optik karakter tanıma 56, 59, 96
otonom araçlar 118
öbekleme 93
  belge 95
  hiyerarşik 97
  izgesel 97
öğrenme
  gözetimli 43
  gözetimsiz 93
  pekiştirmeli 104
  uçtan uca 110
  verisi 43
  yanlılığı 45
öneri modeli 72, 98
örneklem 40
örüntü tanıma 26, 32
özellik
  çıkarma 69, 87
  seçimi 68
özkodlayıcı 88, 90
özyineli ağlar 82

P
paralel
  bilgi işlem 84, 119
  dağıtık işleme 83
parametre 33
parametrik kestirim 57
parametrik olmayan kestirim 58
pekiştirmeli öğrenme 104
Prolog 51

R
rastsallık 39, 106, 127
rastsal orman 71
regresyon 43

S
sayısal
  gösterim 18
  iletişim 21
  teknoloji 17

segment 41
sepet analizi 122
sıkıştırma 47, 95
sınıf 48
sınıflandırma 35, 48
sıralama 72
siber-fiziksel sistemler 121
sigorta 40
sinir ağları 78
sinirbilim 29
sinyal işleme 35
sosyal medya 24, 65, 116
sözcük torbası 65, 95, 97, 101
şablon eşleştirme 56
şans 40, 106

T
tavla 110
TD-Gammon 110
temel bileşen analizi 69
ters mühendislik 31
tıbbi tanı 52
Tolstoy, L. 66
tramvay problemi 128

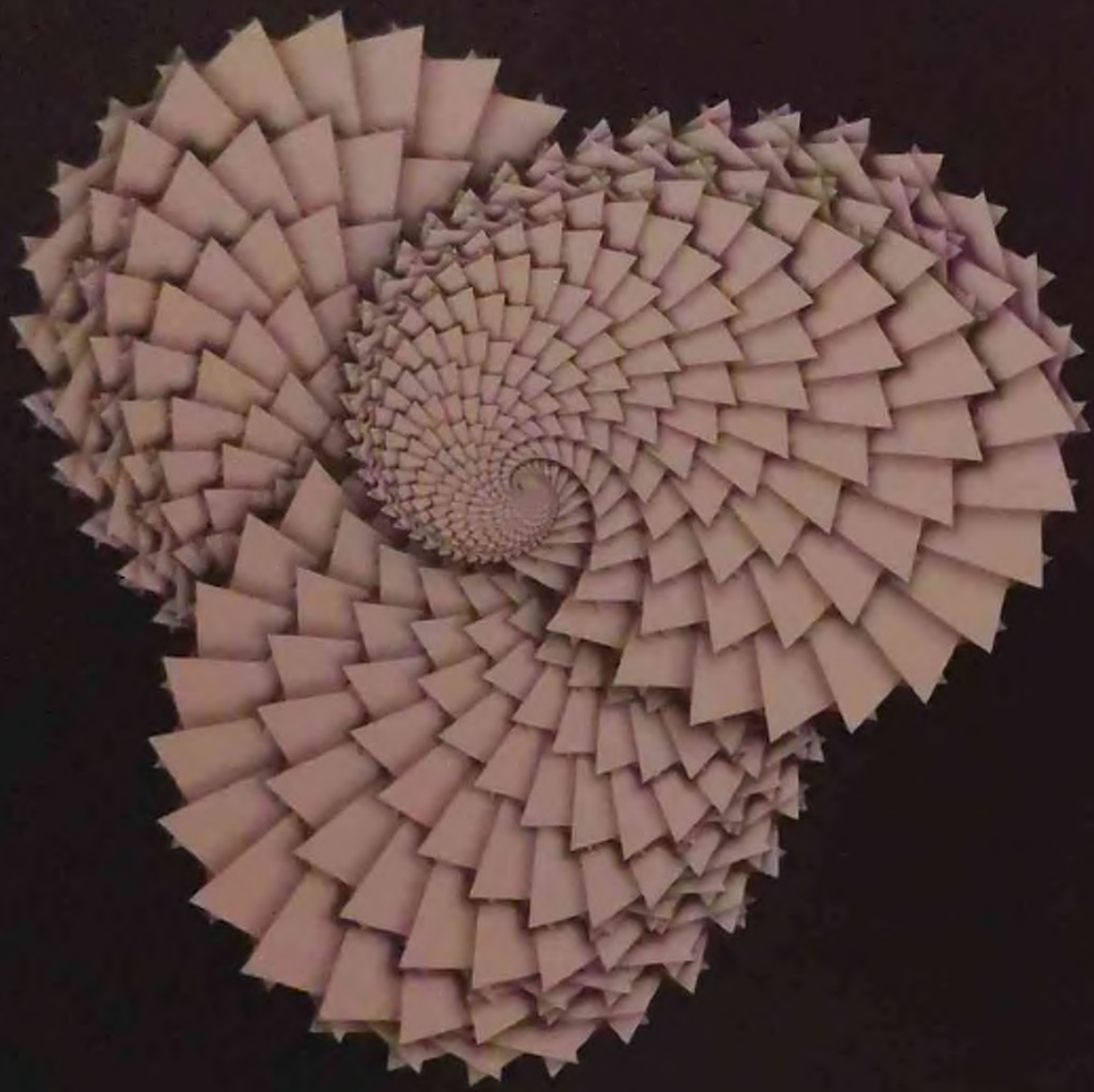
U-Ü
uçtan uca öğrenme 110
uzman sistemler 31, 50
üretici model 59, 98
üst parametre 46

V
veri
ambarı 122
bilimi 125
gizliliği 123
görselleştirme 123
güvenliği 123
iletişimi 20
küpü 122
madenciliği 25, 121
merkezleri 120
tabanları 19
VLSI 35

W
www 21

Y
yapay zekâ 13, 28, 29, 32, 35, 51, 72, 126, 129, 152
yarı parametrik kestirim 57
yenilik tespiti 67
yerel gösterim 84
yüksek performanslı bilgisayar sistemleri 119
yüz tanıma 32, 62, 69

Z
zamansal fark öğrenme 109



Ethem Alpaydın, *Yapay Öğrenme* adlı kitabında günümüzün en heyecan verici alanlarından biri olan yapay öğrenmenin temellerini, veriden öğrenen sistemlerin gelişen teknolojilerle birlikte, yüz tanımadan şoförsüz arabalara, konuşma tanımadan otomatik çeviriye dek yaşamlarımızın her alanına nasıl girdiğini tartışıyor. Bilgisayar biliminin evriminden derin sinir ağlarına, yapay zekâdan veri mahremiyeti ve güvenliğinin etik boyutlarına uzanan geniş çerçevede yapay öğrenmenin sınırlarına ve kullanımlarına ilişkin çok boyutlu ve kapsamlı bir araştırma sunuyor.

Tellekt

www.tellekt.com

ISBN 978-625-7118-09-5

Bilim | Teknoloji

20,5 TL