buzdağı

Yayın Numarası: 1806
1. Baskı: Ekim 2018
ISBN 978-605-82132-9-6

Sertifika Numarası: 41825
Buzdağı Yayınevi
Altay Mahallesi 2668. Cadde
Fırat Life Style Rezidans 1F/61
Eryaman/ANKARA
t: +90 312 219 77 98
f: +90 312 219 55 43
info@buzdagiyayinevi.com

Baskı
Ankamat Matbaacılık
13256

Copyright © 2016 Massachusetts Institute of Technology
© Türkçe yayın hakları Buzdağı Yayınevi'ne aittir.
Bu kitabın hiçbir bölümü, yazarın ve yayınevinin izni alınmadan
basılı ve dijital olarak çoğaltılamaz, yayınlanamaz.

Derin Öğrenme

Ian Goodfellow
Yoshua Bengio
Aaron Courville

Genel Yayın Yönetmeni
Fatih ÖZDEMİR

Yazarlar
Ian Goodfellow
Yoshua Bengio
Aaron Courville

Çevirmenler
Prof. Dr. Fatoş YARMAN VURAL
Dr. Öğr. Üyesi Ramazan Gökberk CİNBİŞ
Doç. Dr. Sinan KALKAN

Çevirmen Yardımcıları
Hüseyin AYDIN
Yarkın Deniz ÇETİN
Berkan DEMİREL
Hazal MOĞULTAY
Mert Bülent SARIYILDIZ
Gencer SÜMBÜL

Editörler
Bilgin AKSOY
Ferhat KURT
Doç. Dr. Vedat ÇELİK

Editör Yardımcıları
Furkan KALINSAZ
Doruk SÖNMEZ

Son Okuma
Ahmet Okan ŞEKER

Dizgi
Veysel TOPRAK

Kapak Fotoğrafı
Daniel Ambrosi
(Joseph Smarr ve Chris Lamb tarafından düzenlenmiştir.)

İçindekiler

Web sitesi **xiii**

Teşekkür **xv**

Çevirmenlerin Ön Sözü **xix**

Derin Öğrenme Sözlüğü **xxi**

Notasyon **xxxix**

1 Giriş **1**

1.1 Bu Kitabı Kimler Okumalı? 9

1.2 Derin Öğrenmede Tarihsel Eğilimler 11

I Uygulamalı Matematik ve Makine Öğrenmesinin Temelleri **27**

2 Doğrusal Cebir **29**

2.1 Skalerler, Vektörler, Matrisler ve Tensörler 29

2.2 Matris ve Vektörleri Çarpmak 32

2.3 Birim ve Ters Matrisler 34

2.4 Doğrusal Bağımlılık ve Uzayı Germe 35

2.5 Normlar 37

2.6 Matrisler ve Vektörlerin Özel Çeşitleri 39

2.7 Özayrışma 40

2.8 Tekil Değer Ayrışımı 43

2.9 Moore-Penrose Tersimsisi 44
2.10 İz Operatörü 45
2.11 Determinant 46
2.12 Örnek: Ana Bileşenler Analizi (PCA) 46

3 Olasılık ve Bilgi Kuramı 51
3.1 Neden Olasılık? 52
3.2 Rastgele Değişkenler 54
3.3 Olasılık Dağılımları 54
3.4 Marjinal Olasılık 56
3.5 Koşullu Olasılık 57
3.6 Koşullu Olasılıkların Zincir Kuralı 57
3.7 Bağımsızlık ve Koşullu Bağımsızlık 58
3.8 Beklenti, Değişirlik ve Eşdeğişirlik 58
3.9 Sık Karşılaşılan Olasılık Dağılımları 60
3.10 Sık Kullanılan Fonksiyonların Kullanışlı Özellikleri 65
3.11 Bayes Kuralı 68
3.12 Sürekli Değişkenlerin Teknik Detayları 68
3.13 Bilgi Kuramı 70
3.14 Yapılandırılmış Olasılık Modelleri 74

4 Sayısal Hesaplama 77
4.1 Taşma ve Küçümenlik 77
4.2 Yetersiz Koşullama 79
4.3 Gradyan-Temelli Eniyileme 79
4.4 Kısıtlı Eniyileme 89
4.5 Örnek: Doğrusal En Küçük Kareler 92

5 Makine Öğrenmesinin Temelleri 95
5.1 Öğrenme Algoritmaları 96
5.2 Kapasite, Aşırı Uydurma ve Yetersiz Uydurma 108
5.3 Hiperparametreler ve Doğrulama Kümeleri 118

5.4 Kestiriciler, Önyargı ve Değişirlik 120
5.5 En Büyük Olabilirlik Kestirimi 129
5.6 Bayes İstatistiği 133
5.7 Denetimli Öğrenme Algoritmaları 137
5.8 Denetimsiz Öğrenme Algoritmaları 144
5.9 Stokastik Gradyan İnişi 149
5.10 Bir Makine Öğrenmesi Algoritması Oluşturmak 151
5.11 Derin Öğrenmeyi Motive Eden Zorluklar 152

II Derin Ağlar: Modern Pratikler 163

6 Derin İleri Besleme Ağları 165
6.1 Örnek: XOR Fonksiyonunu Öğrenmek 168
6.2 Gradyan Tabanlı Öğrenme 174
6.3 Gizli Birimler 188
6.4 Mimari Tasarım 195
6.5 Geri Yayılım ve Diger Türevleme Algoritmaları 201
6.6 Tarihsel Notlar 222

7 Derin Öğrenmede Düzenlileştirme 225
7.1 Parametre Normu Cezaları 227
7.2 Kısıtlanmıs Eniyileme Olarak Norm Cezaları 234
7.3 Düzenlileştirme ve Az Koşullandırılmış Problemler 236
7.4 Veri Kümesi Çeşitleme 237
7.5 Gürültü Direnci 239
7.6 Yarı-denetimli Öğrenme 241
7.7 Çoklu-görev Öğrenme 242
7.8 Erken Durdurma 243
7.9 Parametrelere Eşleme ve Parametre Paylaşımı 250
7.10 Seyrek Gösterimler 252
7.11 İstifleme ve Diğer Topluluk Yöntemleri 254
7.12 İletim Sönümü 256

7.13 Çekişmeli Öğrenme 266
7.14 Tanjant Mesafesi, Tanjant Yayılımı ve Manifold Tanjant Sınıflandırıcısı 268

8 Derin Modellerin Eğitiminde Eniyileme 273
8.1 Öğrenmenin Salt Eniyilemeden Farkı 274
8.2 Sinir Ağı Eniyilemedeki Zorluklar 281
8.3 Temel Algoritmalar 293
8.4 Parametre İlk Değer Atama Stratejileri 299
8.5 Uyarlanır Öğrenme Oranı Kullanan Algoritmalar 306
8.6 İkinci-Dereceden Yaklaşıklık Yöntemleri 310
8.7 Eniyileme Stratejileri ve Meta-Algoritmalar 317

9 Evrişimsel Ağlar 331
9.1 Evrişim İşlemi 332
9.2 Motivasyon 335
9.3 Biriktirme 340
9.4 Son Derece Güçlü Bir Öncül Olarak Evrişim ve Biriktirme 346
9.5 Basit Evrişim Fonksiyonun Başka Biçimleri 347
9.6 Yapılandırılmıs Çıktı 357
9.7 Veri Türleri 359
9.8 Verimli Evrişim Algoritmaları 361
9.9 Rastgele veya Denetimsiz Öznitelikler 361
9.10 Evrişimsel Ağların Sinirbilimsel Temelleri 363
9.11 Evrişimsel Ağlar ve Derin Öğrenmenin Tarihi 370

10 Sıralı Veri Modelleme: Yinelemeli ve Özyinelemeli Ağlar 373
10.1 Hesaplamalı Çizgeleri Açma 375
10.2 Yinelemeli Sinir Ağları 378
10.3 İki Yönlü RNN'ler 393
10.4 Kodlayıcı-Kodçözücü Diziden Dizi Mimarileri 395
10.5 Derin Yinelemeli Sinir Ağları 397
10.6 Özyinelemeli Yapay Sinir Ağları 399
10.7 Uzun Dönem Bağımlılıkların Zorlukları 401

10.8 Yankı Durum Ağları 403
10.9 Sızıntılı Birimler ve Birden Fazla Zaman Ölçeği için Diğer Stratejiler 406
10.10 Uzun Ömürlü Kısa-Dönem Bellek ve Diğer Geçitli RNN'ler 408
10.11 Uzun-Dönem Bağımlılık için Eniyileme 413
10.12 Açık Bellek 416

11 Kullanışlı Metodolojiler 421
11.1 Performans Ölçütleri 422
11.2 Varsayılan Temel Modeller 425
11.3 Daha Fazla Veri Toplayıp Toplamamaya Karar Vermek 426
11.4 Hiperparametreleri Seçme 428
11.5 Hata Ayıklama Stratejileri 437
11.6 Örnek: Çok Rakamlı Sayı Tanıma 441

12 Uygulamalar 445
12.1 Büyük Ölçekli Derin Öğrenme 445
12.2 Bilgisayarlı Görü 454
12.3 Konuşma Tanıma 460
12.4 Doğal Dil İşleme 463
12.5 Diğer Uygulamalar 480

III Derin Öğrenme Araştırmaları 489

13 Doğrusal Faktör Modelleri 493
13.1 Olasılıksal PCA ve Faktör Analizi 494
13.2 Bağımsız Bileşenler Analizi (ICA) 495
13.3 Yavaş Öznitelik Analizi 498
13.4 Seyrek Kodlama 501
13.5 PCA Yönteminin Manifold Olarak Yorumlanması 504

14 Otokodlayıcılar 507
14.1 Tamamlanmamıs Otokodlayıcılar 508

14.2 Düzenlileştirilmiş Otokodlayıcılar 509
14.3 Gösterimsel Güç, Katman Büyüklüğü ve Derinliği 513
14.4 Stokastik Kodlayıcılar ve Kodçözücüler 514
14.5 Arıtan Otokodlayıcılar 516
14.6 Otokodlayıcılar ile Manifold Öğrenme 521
14.7 Daraltan Otokodlayıcılar 525
14.8 Öngörüsel Seyrek Ayrışım 529
14.9 Otokodlayıcılara İlişkin Uygulamalar 530

15 Gösterim Öğrenme 533
15.1 Katmanların Denetimsiz Hırslı Öneğitilmesi 535
15.2 Aktarım Öğrenme ve Alan Uyarlama 543
15.3 Nedensel Faktörlerin Yarı-Denetimli Ayrıştırılması 547
15.4 Dağıtık Gösterim 552
15.5 Derinlikten Üstel Kazanımlar 559
15.6 Altta Yatan Nedenleri Keşfetmek için İpuçları Sağlamak 561

16 Derin Öğrenme için Yapılandırılmış Olasılıksal Modeller 565
16.1 Yapısız Modellemenin Zorluğu 566
16.2 Model Yapısını Açıklamak için Çizge Kullanma 570
16.3 Çizgesel Modellerden Örnekleme 586
16.4 Yapılandırılmış Modellemenin Avantajları 588
16.5 Bağımlılıkları Öğrenme 589
16.6 Çıkarım ve Yaklaşık Çıkarım 590
16.7 Yapılandırılmış Olasılıksal Modellerde Derin Öğrenme Yaklaşımı . 591

17 Monte Carlo Yöntemleri 597
17.1 Örnekleme ve Monte Carlo Yöntemleri 597
17.2 Önem Örnekleme 599
17.3 Markov Zincirli Monte Carlo Yöntemleri 602
17.4 Gibbs Örnekleme 606
17.5 Ayrı Modlar Arasında Karıştırma Zorluğu 607

18 Bölüntü Fonksiyonunu İnceleme **613**
18.1 Log-Benzerlik Gradyanı 614
18.2 Stokastik En Büyük Olabilirlik ve Karşıtsal Iraksama 615
18.3 Yarı Olabilirlik 623
18.4 Skor Eşleştirme ve Oran Eşleştirme 626
18.5 Arıtan Skor Eşleştirme 628
18.6 Gürültüye Karşıt Kestirim 628
18.7 Bölüntü Fonksiyonunu Kestirme 631

19 Yaklaşık Çıkarım **641**
19.1 Eniyileme Olarak Çıkarsama 642
19.2 Beklenti Enbüyütme 644
19.3 En Büyük Ardıl (MAP) Çıkarım ve Seyrek Kodlama 645
19.4 Değişimsel Çıkarım ve Öğrenme 648
19.5 Öğrenilen Yaklaşım Çıkarım 661

20 Derin Üretken Modeller **665**
20.1 Boltzmann Makinesi 665
20.2 Kısıtlandırılmış Boltzmann Makineleri 667
20.3 Derin İnanç Ağları 671
20.4 Derin Boltzmann Makineleri 674
20.5 Gerçek-Değerli Veri için Boltzmann Makinesi 687
20.6 Evrişimsel Boltzmann Makineleri 694
20.7 Yapılandırılmış ve Sıralı Çıktılar için Boltzmann Makineleri 697
20.8 Diğer Boltzmann Makineleri 698
20.9 Rastgele İşlemler Üzerinden Geri Yayılım 699
20.10 Yönlü Üretken Modeller 704
20.11 Otokodlayıcılardan Örnek Almak 723
20.12 Üretken Stokastik Ağlar 726
20.13 Diğer Üretim Yöntemleri 727
20.14 Üretken Modellerin Değerlendirilmesi 728
20.15 Sonuç 731

Kaynakça **733**

İndeks **793**

Web Sitesi

www.deeplearningbook.org

Bu kitap, yukarıda adresi verilen web sitesi ile desteklenmektedir. Bu sitede, okuyuculara ve eğitmenlere faydalı olabilecek alıştırmalar, ders notları ve düzeltilen hatalar gibi pek çok destekleyici türde materyal yer almaktadır.

Teşekkür

Bu kitabın yazımı birçok kişinin katkısı sayesinde gerçekleşmiştir.

Bu kitap ile ilgili önerimize, kitabın içeriği ve organizasyonuna dair geri bildirimlerini esirgemeyenlere teşekkürü borç biliriz: Guillaume Alain, Kyunghyun Cho, Çağlar Gülçehre, David Krueger, Hugo Larochelle, Razvan Pascanu and Thomas Rohée.

Doğrudan kitabın içeriğine dair geri bildirim veren herkese de ayrıca teşekkür etmek istiyoruz. Bazıları birçok bölüm ile ilgili geri bildirimde bulunmuştur: Martín Abadi, Guillaume Alain, Ion Androutsopoulos, Fred Bertsch, Olexa Bilaniuk, Ufuk Can Biçici, Matko Bošnjak, John Boersma, Greg Brockman, Alexandre de Brébisson, Pierre Luc Carrier, Sarath Chandar, Pawel Chilinski, Mark Daoust, Oleg Dashevskii, Laurent Dinh, Stephan Dreseitl, Jim Fan, Miao Fan, Meire Fortunato, Frédéric Francis, Nando de Freitas, Çağlar Gülçehre, Jurgen Van Gael, Javier Alonso García, Jonathan Hunt, Gopi Jeyaram, Chingiz Kabytayev, Lukasz Kaiser, Varun Kanade, Asifullah Khan, Akiel Khan, John King, Diederik P. Kingma, Yann LeCun, Rudolf Mathey, Matías Mattamala, Abhinav Maurya, Kevin Murphy, Oleg Mürk, Roman Novak, Augustus Q. Odena, Simon Pavlik, Karl Pichotta, Eddie Pierce, Kari Pulli, Roussel Rahman, Tapani Raiko, Anurag Ranjan, Johannes Roith, Mihaela Rosca, Halis Sak, César Salgado, Grigory Sapunov, Yoshinori Sasaki, Mike Schuster, Julian Serban, Nir Shabat, Ken Shirriff, Andre Simpelo, David Slate, Scott Stanley, David Sussillo, Ilya Sutskever, Carles Gelada Sáez, Graham Taylor, Valentin Tolmer, Massimiliano Tomassoli, An Tran, Shubhendu Trivedi, Alexey Umnov, Vincent Vanhoucke, Marco Visentini-Scarzanella, Martin Vita, David Warde-Farley, Dustin Webb, Kelvin Xu, Wei Xue, Ke Yang, Li Yao, Zygmunt Zając ve Ozan Çağlayan.

Bölümler ile ilgili çok faydalı geri bildirimlerde bulunan kişilere de teşekkür etmek istiyoruz:

- Notasyon: Zhang Yuanhang J.B. Lee.

- Bölüm 1, Giriş: Yusuf Akgul, Sebastien Bratieres, Samira Ebrahimi, Charlie Gorichanaz, Brendan Loudermilk, Eric Morris, Cosmin Pârvulescu ve Alfredo Solano.
- Bölüm 2, Doğrusal Cebir: Amjad Almahairi, Nikola Banić, Kevin Bennett, Philippe Castonguay, Oscar Chang, Eric Fosler-Lussier, Andrey Khalyavin, Sergey Oreshkov, István Petrás, Dennis Prangle, Thomas Rohée, Gitanjali Gulve Sehgal, Colby Toland, Alessandro Vitale ve Bob Welland.
- Bölüm 3, Olasılık ve Bilgi Kuramı: John Philip Anderson, Kai Arulkumaran, Vincent Dumoulin, Rui Fa, Stephan Gouws, Artem Oboturov, Antti Rasmus, Alexey Surkov ve Volker Tresp.
- Bölüm 4, Sayısal Hesaplama: Tran Lam AnIan Fischer ve Hu Yuhuang.
- Bölüm 5, Makine Öğrenmesinin Temelleri: Dzmitry Bahdanau, Nikhil Garg, Justin Domingue, Makoto Otsuka, Bob Pepin, Philip Popien, Bharat Prabhakar, Emmanuel Rayner, Peter Shepard, Kee-Bong Song, Zheng Sun ve Andy Wu.
- Bölüm 6, Derin İleri Besleme Ağları: Uriel Berdugo, Fabrizio Bottarel, Elizabeth Burl, Ishan Durugkar, Jeff Hlywa, Jong Wook Kim, David Krueger, Aditya Kumar Praharaj ve Sten Sootla.
- Bölüm 7, Derin Öğrenmede Düzenlileştirme: Morten Kolbæk, Kshitij Lauria, Inkyu Lee, Sunil Mohan, Hai Phong Phan ve Joshua Salisbury.
- Bölüm 8, Derin Modellerin Eğitiminde Eniyileme: Marcel Ackermann, Peter Armitage, Rowel Atienza, Andrew Brock, Tegan Maharaj, James Martens, Mostafa Nategh, Kashif Rasul, Klaus Strobl ve Nicholas Turner.
- Bölüm 9, Evrişimsel Ağlar: Martín Arjovsky, Eugene Brevdo, Konstantin Divilov, Eric Jensen, Mehdi Mirza, Alex Paino, Marjorie Sayer, Ryan Stout ve Wentao Wu.
- Bölüm 10, Sıralı Veri Modelleme: Yinelemeli ve Özyinelemeli Ağlar: Gökçen Eraslan, Steven Hickson, Razvan Pascanu, Lorenzo von Ritter, Rui Rodrigues, Dmitriy Serdyuk, Dongyu Shi ve Kaiyu Yang.
- Bölüm 11, Kullanışlı Metodolojiler: Daniel Beckstein.
- Bölüm 12, Uygulamalar: George Dahl, Vladimir Nekrasov ve Ribana Roscher.
- Bölüm 13, Doğrusal Faktör Modelleri: Jayanth Koushik.

- Bölüm 15, Gösterim Öğrenme: Kunal Ghosh, Rodney Melchers, Gudmundur Einarsson.
- Bölüm 16, Derin Öğrenme için Yapılandırılmış Olasılıksal Modeller: Minh Lê ve Anton Varfolom.
- Bölüm 18, Bölüntü Fonksiyonunu İnceleme: Sam Bowman.
- Bölüm 19, Yaklaşık Çıkarım: Yujia Bao.
- Bölüm 20, Derin Üretken Modeller: Nicolas Chapados, Daniel Galvez, Wenming Ma, Fady Medhat, Shakir Mohamed ve Grégoire Montavon.
- Kaynakça: Lukas Michelbacher ve Leslie N. Smith.

Yayınlarından görselleri, şekilleri veya verileri tekrar üretmemize izin verenlere de teşekkür etmek istiyoruz. Bu kişilerin katkılarını tüm metin boyunca şekil açıklamalarında belirtmiş durumdayız.

Bunların yanında, kitabın internet versiyonunu oluşturabilmek için kullandığımız pdf2htmlEX'i yazan ve ortaya çıkan HTML kalitesinin arttırılması ile ilgili destekte bulunan Lu Wang'e teşekkür ediyoruz.

Ian'ın eşi Daniela Flori Goodfellow'a kitabın yazımında Ian'ı sabırlı bir şekilde desteklediği ve kitap üzerinde düzeltmeler yaptığı için teşekkür ediyoruz.

Ian'ın bu kitabı yazmaya ve meslektaşlarımızdan geri bildirim ve rehberlik almaya bolca zaman ayırabildiği entelektüel bir ortam sağladıkları için Google Brain ekibine teşekkür etmek istiyoruz. Ian'ın eski müdürü Greg Corrado ve mevcut müdürü Samy Bengio'ya bu projeyi destekledikleri için özellikle teşekkür ediyoruz. Son olarak, yazmanın zor geldiği zamanlardaki teşvikleri için Geoffrey Hinton'a teşekkür ediyoruz.

Çevirmenlerin Ön Sözü

1990'lı yıllarda yaşanan bilişim devrimi ile bilgisayarlar hayatımızın her kesitini değiştirerek büyük bir sosyal ve ekonomik dönüşüme neden oldu. Bu dönüşümden çok daha etkili ve hızlı bir devrimi ise son birkaç yıldır, Yapay Zekâ teknolojilerinde kaydedilen büyük bir atılımla yaşamaya başladık.

Derin Öğrenme adı verilen bu yeni teknolojiler insan beynindeki sinir ağlarından esinlenerek geliştirildi. İnsan gibi hatta zaman zaman insandan daha iyi algılayan, gören, duyan, konuşan ve öğrenen bu teknolojiler sayesinde, doğadaki akıllı organizmaları taklit edebiliyoruz. Hatta daha da ileri giderek doğada bulunmayan karmaşık öğrenme yöntemleri de geliştirebiliyoruz. "Akıllı sistemler", bir canlı gibi çevreyi algılıyor ve anlamlandırabiliyor. Bunun da ötesinde, insanın göremediğini görebilen, duyamadığını duyabilen, algılayamadığını algılayan ve tanıyan makineler geliştirebiliyoruz. Örneğin; manyetik rezonans görüntüleme teknikleri ile insanın çıplak gözle göremediği organları gören ve hastalık teşhisinde doktorlara yardımcı olan karar destek sistemleri tasarlayabiliyoruz. Ya da örümceklerin ağ kurma yöntemlerini taklit ederek örümceğin yapamadığı sağlamlıkta yapılar oluşturabiliyoruz.

Derin Öğrenme yöntemleri, her gün kullandığımız cihazların daha kullanışlı ve verimli hâle gelmesini sağladı. Sürücüsüz arabalar, kendi kendine hareket eden insansız hava ve kara araçları, akıllı evler, akıllı şehirler ve akıllı fabrikaların oluşturulmasına olanak sağladı. Tam otomasyonlu üretimi sağlaması beklenen akıllı fabrikaların geliştirileceği Endüstri 4.0 çağını tetikledi.

Derin Öğrenme yöntemlerinin henüz tüm dünyada gelişme aşamasında olmasından dolayı, gelişmiş ve gelişmekte olan ülkeler yarışa aynı noktadan başlamaktadır. Bu durum, Türkiye gibi sınırlı sayıda da olsa yetişmiş insan gücüne sahip bir ülke için büyük bir fırsat sunmaktadır. Bu kitap, ülkemizin derin öğrenme alanında daha iyi yerlere gelmesini sağlamak için gerekli insan gücünü artırmayı hedeflemektedir. Bu vizyon doğrultusunda, kitabın çevirisi büyük bir titizlik, özveri ve emek ile tamamlanmıştır.

Dikkat edilmesi gereken birkaç hususun belirtilmesi yararlı olacaktır:

- Çeviri esnasında, olabildiğince TÜBA Bilim Terimleri Sözlüğü ve TBD Bilişim Terimleri Sözlüğü dikkate alınmıştır. Ancak, Derin Öğrenme yöntemlerinin ve yazılımın baş döndüren bir hızla ilerlemesinin bir sonucu olarak, bu alandaki terimlerin bir kısmının Türkçe karşılığının olmadığı, birden fazla sayıda Türkçe karşılığının olduğu, önerilmiş Türkçe karşılıkların anlaşılır olmadığı veya sözlüklerde geçen karşılıklarının metnin takibini zorlaştırdığı tespit edilmiştir. Bu tür terimlerin Türkçe karşılıkları belirlenirken, metnin kolay anlaşılır ve takip edilir olması hedeflenmiştir. Bunu kolaylaştırmak için metin içerisinde ilk geçtikleri yerlerde önemli terimlerin İngilizce karşılıkları (çoğunlukla ek açıklamalar ile) verilmiştir.

- Denklemler, İngilizce yayın ve programlama kaynaklarının takibini kolaylaştırmak maksadıyla çevirilmemiştir ve bu maksat doğrultusunda, denklemlerde geçen bazı terimler metin içerisinde Türkçeleştirilmemiştir.

- Algoritmaları içeren sözde kodlar da okuyucunun programlama dillerindeki İngilizce komutları bildiği varsayımı ile çevrilmemiştir.

Bu çeviri Ortadoğu Teknik Üniversitesi Bilgisayar Mühendisliği'ndeki pek çok öğretim üyesi ve araştırma görevlisinin katkısı ile gerçekleştirilmiştir. Makine Öğrenmesi alanındaki terimlerin Türkçe karşılıklarını oluşturmak için bize destek veren tüm arkadaşlarımıza teşekkür ederiz.

Derin Öğrenme Sözlüğü

a

absolute value verification: mutlak değer doğrulaması
accuracy: doğruluk
activation function: aktifleştirme fonksiyonu
active constraint: aktif kısıt
adaptive linear element: uyarlanır doğrusal eleman
adversarial example: çekişmeli örnek
adversarial training: çekişmeli eğitme
affine: ilgin
almost everywhere: hemen hemen her yerde
almost sure convergence: hemen hemen kesin yakınsama
ancestral sampling: geçmiş örnekleme
annealed importance sampling: tavlı önem örnekleme
approximation: yaklaşıklık
approximate bayesian computation: yaklaşık Bayes hesaplaması
approximate inference: yaklaşık çıkarım
array: dizi
artificial intelligence: yapay zekâ
artificial neural network: yapay sinir ağı
asymptotically unbiased: asimptotik önyargısız
asynchronize: asenkron
attribute: nitelik
audio: ses
autoencoder: otokodlayıcı
automatic speech recognition: konuşma tanıma
autoregressive networks: otobağlayıcı ağlar

b

back-propagation: geri yayılım
back-propagation through time: zamanla geri yayılım
bag of words: sözcük çuvalı
bagging: istifleme
batch: yığın
batch normalization: yığın normalleştirme
Bayes error: Bayes hatası
Bayes rule: Bayes kuralı
Bayesian hyperparameter optimization: Bayesçi hiperparametre eniyileme
Bayesian network: Bayes ağı
Bayesian probability: Bayes olasılığı
Bayesian statistics: Bayes istatistiği
belief network: inanç ağı
benchmark: denek taşı
Bernoulli distribution: Bernoulli dağılımı
bias: önyargı
bias parameter: önyargı parametresi
biased importance sampling: önyargılı önem örnekleme
binary relation: ikili bağıntı
block gibbs sampling: blok Gibbs örnekleme
boltzmann distribution: Boltzmann dağılımı
boltzmann machine: Boltzmann makinesi
broadcasting: yayımlama
brute force: kaba kuvvet
burn in: alıştırma

c

calculus: analiz
calculus of variations: değişim analizi
cascade: aşamalı
categorical distribution: kategorik dağılım
centering trick: ortalama hilesi
central limit theorem: merkezi limit teoremi
chain rule: zincir kuralı
chain rule of productivity: olasılığın zincir kuralı
chess: satranç
chord: kiriş

chordal graph: kirişli çizge
class-based language models: sınıf tabanlı dil modelleri
classical dynamic system: klasik dinamik sistem
classification: sınıflandırma
clique: klik
clique potential: klik potansiyeli
collaborative filtering: dayanışmalı filtreleme
collider: çarpıştırıcı
color images: renkli görüntüler
complex cell: karmaşık hücre
computational graph: hesaplamalı çizge
computer vision: bilgisayarlı görü
concept drift: kavram sapması
condition number: matrisin sağlamlık sayısı
conditional computation: koşullu hesaplama
conditional independence: koşullu bağımsızlık
conditional probability: koşullu olasılık
conditional RBM: koşullu RBM
connectionism: bağlantıcılık
connectionist temporal classification: bağlantıcı zamansal sınıflandırma
consistency: tutarlılık
constancy: sabitlik
constrained optimization: kısıtlı eniyileme
content-based addressing: içerik-tabanlı adresleme
content-based recommender systems: içerik-tabanlı öneri sistemleri
context: bağlam
context specific independence: bağlama özgü bağımsızlık
contextual bandits: bağlamsal yol kesmeler
continuation methods: sürdürme yöntemleri
continuous: sürekli
contractive autoencoder: daraltan otokodlayıcı
contrast: karşıtlık
contrastive divergence: karşıtsal ıraksama
convex optimization: dışbükey eniyileme
convolution: evrişim
convolutional network: evrişimsel ağ
convolutional neural network: evrişimsel sinir ağı
coordinate descent: koordinat inişi
coordinate ascent: koordinat çıkışı

correlation: korelasyon
cost function: maliyet fonksiyonu
covariance: eşdeğişirlik
covariance matrix: eşdeğişirlik matrisi
coverage: kapsama
critical temperature: kritik sıcaklık
cross-correlation: çapraz korelasyon
cross-entropy: çapraz entropi
cross-validation: çapraz sağlama
curriculum learning: müfredat öğrenme
curse of dimensionality: boyut laneti

d

d-seperation: d-ayırma
damping: söndürme
data generating distribution: veri üretim dağılımı
data generating process: veri üretim işlemi
data parallelism: veri paralelleştirme
dataset: veri kümesi
dataset augmentation: veri kümesi çeşitleme
debugging: hata ayıklama
decision tree: karar ağacı
decoder: kodçözücü
deep belief network: derin inanç ağı
deep Boltzmann machine: derin Boltzmann makinesi
deep feedforward network: derin ileri besleme ağı
deep learning: derin öğrenme
degenerate: yozlaşmış
denoising autoencoder: arıtan otokodlayıcı
denoising score matching: arıtan skor eşleştirme
density estimation: yoğunluk kestirimi
derivative: türev
design matrix: açıklayıcı değişkenler matrisi
detector layer: dedektör katman
diagonal matrix: köşegen matris
differential entropy: diferansiyel entropi
Dirac delta function: Dirac delta fonksiyonu
directed acyclic graph: yönlü çevrimsiz çizgesel

directed graph model: yönlü çizgesel model
directional derivative: yönlü türev
discriminative fine-tuning: ayrımcı hassas ayar
discriminative RBM: ayrımcı RBM
distributed representation: dağıtık gösterim
domain adaptation: alan uyarlama
dot product: iç çarpım
double backprop: çift geri yayılım
doubly block circulant matrix: çift bloklu dairesel matris
dream sleep: rüya uykusu
dropconnect: bağ sönümü
dropout: iletim sönümü
dynamic structure: dinamik yapı

e
e-step: e-adım
early stopping: erken durdurma
echo state network: yankı durum ağı
effective capacity: etkin kapasite
eigendecomposition: özayrışma
eigenvalue: özdeğer
eigenvector: özvektör
elementwise product: eleman çarpımı
embedding: gömülme
empirical distribution: deneysel dağılım
emprical risk: deneysel risk
emprical risk minimization: deneysel risk enküçültmesi
encoder: kodlayıcı
energy function: enerji fonksiyonu
energy-based model: enerji tabanlı model
enhanced gradient: gelişmiş gradyan
ensemble methods: topluluk yöntemleri
epoch: epok
equality constraint: eşitlik kısıtı
equivariance: eşit değişirlik
error function: hata fonksiyonu
estimation: kestirim
Euclidean norm: Euclid normu

Euclidean space: Euclid uzayı
Euler-Lagrange equation: Euler-Lagrange eşitliği
evidence lower bound: kanıt alt sınırı
example: örnek
expectation: beklenti
expectation maximization: beklenti enbüyütme
expected value: beklenti değeri
explaining away: savuşturma
exploitation: faydalanma
exploration: keşif
exponential: üstel
exponential distribution: üstel dağılım

f
f-score: f-skor
factorial: çarpınım
factor: faktör
factor analysis: faktör analizi
factor graph: faktörler çizgesi
factors of variation: değişirlik faktörleri
feature: öznitelik
feature selection: öznitelik seçme
feedforward neural network: ileri beslemeli sinir ağı
fine-tuning: hassas ayar
finite differences: sonlu farklar
forget gate: unutma kapısı
forward propagation: ileri yayılım
Fourier transform: Fourier dönüşümü
foveation: foveasyon
free energy: serbest enerji
frequency: frekans
frequentist probability: deneysel olasılık
frequentist statistics: deneysel istatistik
Frobenius norm: Frobenius normu
fully visible Bayes network: tamamen görünür Bayes ağı
function: fonksiyon
functional derivatives: fonksiyonel türevler

g

Gabor function: Gabor fonksiyonu
gated recurrent unit: kapılı yinelemeli ünite
Gaussian distribution: Gauss dağılımı
Gaussian kernel: Gauss kerneli
Gaussian mixture: Gauss karışımı
generalization: genelleştirme
generalized Lagrange function: genelleştirilmiş Lagrange fonksiyonu
generalized lagrange: genelleştirilmiş Lagrange
generative adversarial networks: üretken çekişmeli ağlar
generative moment matching networks: üretken moment eşleyen ağlar
generator network: üretici ağ
Gibbs distribution: Gibbs dağılımı
Gibbs sampling: Gibbs örneklemesi
global contrast normalization: global karşıt normalleştirme
GPU: grafik işlem birimi
gradient: gradyan
gradient clipping: gradyan kırpma
gradient ascent: gradyan çıkışı
gradient descent: gradyan inişi
graph: çizge
graph embedding: çizge gömülmesi
graphical model: çizgesel model
graphics processing unit: grafik işleme birimi
graph theory: çizge kuramı
greedy algorithm: hırslı algoritma
greedy layer-wise unsupervised pretraining: katmanların denetimsiz hırslı öneğitilmesi
greedy supervised pretraining: hırslı denetimli öneğitme
grid search: ızgara arama

h

Hadamard product: Hadamard çarpımı
harmonium: harmonyum
harmony theory: harmoni kuramı
Helmholtz free energy: Helmholtz serbest enerjisi
Hessian matrix: Hesse matrisi
heteroscedastic: ayrı değişirlikli

hidden layer: gizli katman
hidden unit: gizli birim
hill climbing: tepe tırmanışı
hyperparameter optimization: hiperparametre eniyilemesi
hyperparameters: hiperparametreler
hypothesis space: hipotez uzayı

i

i.i.d. assumptions: i.i.d. varsayımları
identity matrix: birim matris
image: görüntü
ImageNet Large Scale Visual Recognition Challange: ImageNet Geniş Ölçekli Tanıma Yarışması
immorality: aykırılık
implementation: gerçekleştirme
importance sampling: önem örnekleme
importance weighted autoencoder: önem ağırlıklı otokodlayıcı
independence: bağımsızlık
independent and identically distributed: bağımsız özdeşçe dağılmış
independent component analysis: bağımsız bileşenler analizi
independent subspace analysis: bağımsız altuzaylar analizi
inequality constraint: eşitsizlik kısıtı
inference: çıkarım, çıkarsama
intercept: kesim noktası
inferotemporal cortex: şakak altı korteksi
information retrieval: bilgi getirimi
initialization: ilk değer atama
interpolate: aradeğeri bulmak
intractable: kolay çözümlenemez
invariance: değişmezlik
isotropic: izotrop

j

Jacobian matrix: Jacob matrisi
joint probability: ortak olasılık

k

k-means: k-ortalama
k-nearest neighbor: k-en yakın komşu
Karush-Kuhn-Tucker conditions: Karush-Kuhn-Tucker koşulları
kernel machine: kernel makinesi
kernel trick: kernel hilesi
KKT conditions: KKT koşulları
KL divergence: KL ıraksaması
knowledge base: işlenmiş bilgi tabanı
Krylov methods: Krylov yöntemleri
Kullback-Leibler divergence: Kullback-Leibler ıraksaması

l

label: etiket
label smoothing: etiket düzleştirme
ladder network: basamaklı ağ
Lagrange multipliers: Lagrange çarpanları
Lagrangian: Lagrange fonksiyonu
laplace distribution: Laplace dağılımı
latent: saklı
latent variable: saklı değişken
layer: katman
leaky ReLU: sızıntılı ReLU
leaky units: sızıntılı birimler
learnin grate: öğrenme oranı
likelihood: olabilirlik
line search: doğru üzerinde arama
linear combination: doğrusal bileşim
linear dependence: doğrusal bağımlılık
linear factor model: doğrusal faktör modeli
linear regression: doğrusal bağlanım
link prediction: bağlantı öngörüsü
Lipschitz constant: Lipschitz sabiti
Lipschitz continuous: Lipschitz süreklisi
liquid state machine: Likit durum makinesi
local conditional probability distribution: yerel koşullu olasılık dağılımı
local contrast normalization: yerel karşıtlık normalleştirmesi
logistic regression: lojistik bağlanım

logistic sigmoid: lojistik sigmoid
long short-term memory: uzun ömürlü kısa-dönem belleği
look-up table: başvuru çizelgesi
loop: döngü
loopy belief propagation: döngüsel inanç yayılımı
loss function: kayıp fonksiyonu

m
m-step: m-adım
machine learning: makine öğrenmesi
machine translation: makine tercümesi
main diagonal: ana köşegen
manifold hypothesis: manifold hipotezi
manifold learning: manifold öğrenme
manifold tangent classifier: manifold tanjant sınıflandırıcısı
MAP approximation: MAP yaklaştırması
marginal probability: marjinal olasılık
Markov chain: Markov zinciri
Markov chain Monte Carlo: Markov zincirli Monte Carlo
Markov network: Markov ağı
Markov random field: Markov rastgele alanı
matrix: matris
matrix inverse: matris tersi
matrix product: matris çarpımı
max norm: en büyük norm
max pooling: en büyükleri biriktirme
maximize: enbüyütme
maximum aposteriori: en büyük ardıl
maximum likelihood: en büyük olabilirlik
maxout: büyük-çıktı
mean field: orta alan
mean squared error: ortalama karesel hata
measure theory: ölçme kuramı
measure zero: sıfır ölçüm
medial temporal lobe: orta şakak lobu
memory network: bellek ağı
method of steepest descent: en dik iniş yöntemi
minibatch: miniyığın

minimize: enküçültme
missing inputs: eksik girdiler
mixing: karıştırma
mixture density networks: karma yoğunluk ağları
mixture distribution: karma dağılım
mixture model: karma model
mixture of experts: uzman karması
model averaging: model ortalama
model compression: model sıkıştırması
model identifiability: model tanımlanabilirliği
model parallelism: model benzeşmesi
moment matching: moment eşleştirme
Moore-Penrose pseudoinverse: Moore-Penrose tersimsisi
moralized graph: düzeltilmiş çizge
multi-modal learning: çok-doruklu öğrenme
multi-prediction DBM: çok-öngörülü DBM
multi-task learning: çoklu-görev öğrenme
multilayer perception: çok katmanlı algılama
multilayer perceptron: çok katmanlı perseptron
multinomial distribution: çok terimli dağılım
multinoulli distribution: çoklu Bernoulli dağılımı

n
naive Bayes: naif Bayes
natural image: doğal görüntü
natural language processing: doğal dil işleme
nearest neighbor regression: en yakın komşu bağlanımı
negative definite: negatif tanımlı
negative phase: negatif faz
Nesterov momentum: Nesterov momentumu
Netflix Grand Prize: Netflix Büyük Ödülü
neural language model: sinirsel dil modeli
neural network: sinir ağı
neural turing machine: sinirsel Turing Makinesi
neural machine translation: sinirsel makine çevirisi
neuroscience: sinirbilim
Newton's method: Newton yöntemi
no free lunch theorem: "bedaya yemek yok" teoremi
noise-contrastive estimation: gürültüye karşıt kestirim

nonlinear: eğrisel
nonlinearity: eğrisellik
nonparametric model: parametrik olmayan model
normal distribution: normal dağılım
normal equations: normal denklemler
normalized initialization: normalleştirilmiş ilk değer atama
numerical differentiation: sayısal farklar

o
object detection: nesne saptama
object recognition: nesne tanıma
objective function: amaç fonksiyonu
one-hot representation: bir-elemanı-bir olan gösterim
one-hot vector: bir-elemanı-bir olan vektör
one-shot learning: tek seferde öğrenme
operation: işlem
optimization: eniyileme
orthodox statistics: Ortodoks istatistiği
orthogonal matching pursuit: ortogonal eşleştirme takibi
orthogonal matrix: ortogonal matris
orthogonality: ortogonallik
output layer: çıktı katmanı

p
padding: doldurma
parallel distributed processing: paralel dağıtık işleme
parameter initialization: parametrelere ilk değer ataması
parameter sharing: parametre paylaşımı
parameter tying: parametre eşleme
parametric model: parametrik model
parametric ReLU: parametrik ReLU
parent: üst
parsing: Sözdizimsel ayrıştırma
partial derivative: kısmi türev
partition function: bölüntü fonksiyonu
perceptron: perseptron
persistent contrastive divergence: kalıcı karşıtsal ıraksama

perturbation analysis: pertürbasyon analizi
point estimator: nokta kestiricisi
policy: politika
pooling: biriktirme
positive definite: pozitif tanımlı
positive phase: pozitif faz
posterior: ardıl
posterior probability: ardıl olasılık
posterior distribution: ardıl dağılım
precision: kesinlik
predictive sparse decomposition: öngörüsel seyrek ayrışım
preprocessing: önişleme
presentation: sunum
pretraining: öneğitme
primary visual cortex: birincil görsel korteks
principal component analysis: ana bileşenler analizi
prior: öncül
prior probability: öncül olasılık
prior probability distribution: öncül olasılık dağılımı
probabilistic PCA: olasılıksal PCA
probabilistic model: olasılıksal model
probability density function: olasılık yoğunluk fonksiyonu
probability distribution: olasılık dağılımı
probability mass function: olasılık kütle fonksiyonu
probability mass function estimation: olasılık kütle fonksiyonu kestirimi
product of experts: uzman çarpımı
product rule of probability: olasılığın çarpım kuralı
pseudolikelihood: yarı benzerlik

q
quadrature pair: dört evreli çift
quasi-Newton methods: Newton-vari yöntemler

r
radial basis function: dairesel baz fonksiyonu
random search: rastgele arama
random variable: rastgele değişken
ratio matching: oran eşleştirme

recall: duyarlılık
receptive field: alım alanı
recommender systems: önerici sistemler
rectified linear unit: düzeltilmiş doğrusal birim
recurrent network: yinelemeli ağ
recurrent neural network: yinelemeli sinir ağı
regression: bağlanım
regularization: düzenlileştirme
regularizer: düzenlileştirici
reinforcement learning: pekiştirmeli öğrenme
relational database: ilişkisel veri tabanı
relations: bağıntılar
reparametrization trick: yeniden parametreleştirme hilesi
representation: gösterim
representation learning: gösterim öğrenme
representational capacity: gösterimsel kapasite
restricted Boltzmann machine: kısıtlandırılmış Boltzmann makinesi
Ridge regression: Ridge bağlanımı

s

saddle points: eyer noktaları
sample mean: örnekleme ortalaması
scalar: skaler
score matching: skor eşleştirme
second derivative: ikinci türev
second derivative test: ikinci türev testi
self-information: özbilgi
semantic hashing: anlamsal adresleme
semi-supervised learning: yarı denetimli öğrenme
separable convolution: ayrılabilir evrişim
separation: ayırma
sequence: sıra (sıralı dizi)
set: küme
Shannon entropy: Shannon entropisi
sigmoid belief network: sigmoid inanç ağı
simple cell: yalın hücre
simultaneous: eşzamanlı
singular value: tekil değer

singular value decomposition: tekil değer ayrışımı
singular vector: tekil vektör
slow feature analysis: yavaş öznitelik analizi
smoothness: pürüzsüzlük
spam detection: spam saptama
sparse coding: seyrek kodlama
spike and slab sparse coding: sivri ve kalın seyrek kodlama
sparse initialization: seyrek ilk değer atama
sparse representation: seyrek gösterim
spectral radius: spektral yarıçap
speech recognition: konuşma tanıma
sphering: küre şeklini verme
square matrix: kare matris
standard deviation: standart sapma
standard error: standart hata
standard error of the mean: ortalamanın standart hatası
statistic: istatistik
statistical learning theory: istatistiksel öğrenme kuramı
steepest descent: en dik iniş yöntemi
stochastic back-propagation: stokastik geri yayılım
stochastic gradient ascent: stokastik gradyan çıkışı
stochastic gradient descent: stokastik gradyan inişi
stochastic maximum likelihood: stokastik en büyük olabilirlik
stochastic pooling: stokastik biriktirme
structure learning: yapısal öğrenme
structured output: yapılandırılmış çıktı
structured probabilistic model: yapılandırılmış olasılık modeli
sum rule of probability: olasılığın toplam kuralı
sum product network: toplam-çarpım ağı
supervised: denetimli
supervised fine-tuning: denetimli hassas ayar
supervised learning: denetimli öğrenme
support vector machine: destek vektör makinesi
surrogate loss function: vekil kayıp fonksiyonu
symmetric matrix: simetrik matris
synchronize: senkron

t

tangent distance: tanjant mesafesi
tangent plane: tanjant düzlemi
tangent prop: tanjant yayılımı
teacher forcing: öğretmen zoru
tempering: kızdırma
template matching: şablon eşleştirme
tensor: tensör
test set: test kümesi
Tikhonov regularization: Tikhonov düzenlileştirmesi
tiled convolution: döşeli evrişim
time-delay neural network: zaman-gecikmeli sinir ağı
Toeplitz matrix: Toeplitz matrisi
topographic ICA: topografik ICA
trace operator: iz operatörü
tractable: kolay çözümlenebilir
training: eğitme
training error: eğitim hatası
transcription: transkripsiyon
transfer learning: aktarım öğrenme
transpose: transpoz
triangle inequality: üçgen eşitsizliği
triangulated graph: üçgenlemeli çizge
tuple: çokuzlu

u

unbiased: önyargısız
underdetermined: eksik belirtilmiş
underflow: küçümenlik
undirected graphical model: yönsüz çizgesel model
undirected model: yönsüz model
uniform distribution: düzgün dağılım
unimodal distribution: tek tepeli dağılım
unit norm: birim norm
unit vector: birim vektör
universal approximation theorem: evrensel yaklaşıklık teoremi
universal approximator: evrensel yakınlaştırıcı
unnormalized probability distribution: normalleştirilmemiş olasılık dağılımı

unpooling: dağıtım
unsupervised learning: denetimsiz öğrenme
unsupervised pretraining: denetimsiz öneğitme

v
v-structure: v-yapısı
Vapnik-Chervonenkis dimension: Vapnik-Chervonenkis boyutu
variable: değişken
variance: değişirlik
variational: değişimsel
variational autoencoder: değişimsel otokodlayıcı
variational derivatives: değişken türevler
variational free energy: değişken serbest enerji
variational inference: değişimsel çıkarım
VC dimension: VC boyutu
vector: vektör
virtual adversarial examples: sanal çekişmeli örnekler
visible layer: görünür katman
visible variable: görünür değişken
volumetric data: hacimsel veri

w
Wake-Sleep: Uyandırma-Uyutma
weight decay: ağırlık azalımı
weight space symmetry: ağırlık uzayı simetrisi
whitening: beyazlaştırma
word embedding: kelime gömülmesi

z
zero-data learning: verisiz öğrenme
zero-shot learning: eksik etiketli öğrenme
0-1 loss: 0-1 kayıp

Notasyon

Bu bölümde kitapta kullandığımız notasyonun bir özetini sunuyoruz. Eğer buradaki matematiksel kavramlara aşina değilseniz, çoğunun açıklamalarını Bölüm 2–4'te bulabilirsiniz.

Sayılar ve Diziler

a	Bir skaler (tamsayı veya gerçek sayı)
$\boldsymbol{a}$	Bir vektör
$\boldsymbol{A}$	Bir matris
A	Bir tensör
$\boldsymbol{I}_n$	n satırı ve n sütunu olan bir birim matris
$\boldsymbol{I}$	Boyutu içinde bulunduğu bağlama göre değişen bir birim matris
$\boldsymbol{e}^{(i)}$	i pozisyonunda 1 olan standart bir baz vektörü, $[0, \ldots, 0, 1, 0, \ldots, 0]$
$\mathrm{diag}(\boldsymbol{a})$	Köşegenindeki girdileri $\boldsymbol{a}$ ile belirtilen, karesel, köşegenel bir matris.
a	Bir skaler rastgele değişken
$\mathbf{a}$	Vektörel değerli bir rastgele değişken
$\mathbf{A}$	Matris değerli bir rastgele değişken

Kümeler ve Çizgeler

$\mathbb{A}$	Bir küme
$\mathbb{R}$	Gerçek sayılar kümesi
$\{0, 1\}$	0 ve 1 içeren küme
$\{0, 1, \ldots, n\}$	0 ve n arasındaki tüm tamsayıları içeren küme
$[a, b]$	a ve b'yi de içeren gerçek sayılar aralığı
$(a, b]$	a'nın dahil olmadığı, fakat b'nin dahil olduğu gerçek sayılar aralığı
$\mathbb{A}\backslash\mathbb{B}$	Küme farkı, yani $\mathbb{A}$'nın $\mathbb{B}$'de olmayan tüm elemanlarının oluşturduğu küme
$\mathcal{G}$	Bir çizge
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	x_i'nin $\mathcal{G}$'deki ebeveynleri

İndeksleme

a_i	$\boldsymbol{a}$ vektörünün i elemanı, burada dizinleme 1 ile başlamaktadır
a_{-i}	$\boldsymbol{a}$ vektörünün i haricindeki tüm elemanları
$A_{i,j}$	$\boldsymbol{A}$ matrisinin i, j elemanı
$\boldsymbol{A}_{i,:}$	$\boldsymbol{A}$ matrisinin i satırı
$\boldsymbol{A}_{:,i}$	$\boldsymbol{A}$ matrisinin i sütunu
$\mathsf{A}_{i,j,k}$	3 Boyutlu A tensörünün (i, j, k) elemanı
$\mathsf{A}_{:,:,i}$	3 Boyutlu tensörün 2 Boyutlu kesiti
a_i	$\mathbf{a}$ rastgele vektörünün i elemanı

Doğrusal Cebir İşlemleri

$\boldsymbol{A}^\top$	$\boldsymbol{A}$ matrisinin transpozu
$\boldsymbol{A}^+$	$\boldsymbol{A}$'nın Moore-Penrose tersimsisi
$\boldsymbol{A} \odot \boldsymbol{B}$	$\boldsymbol{A}$ ve $\boldsymbol{B}$'nin eleman bazında (Hadamard) çarpımı
$\det(\boldsymbol{A})$	$\boldsymbol{A}$'nın determinantı

Kalkülüs

$\dfrac{dy}{dx}$	y'nin x'e göre türevi
$\dfrac{\partial y}{\partial x}$	y'nin x'e göre kısmi türevi
$\nabla_{\boldsymbol{x}} y$	y'nin x'e göre gradyanı
$\nabla_{\boldsymbol{X}} y$	y'nin $\boldsymbol{X}$'e göre matris türevi
$\nabla_{\mathbf{X}} y$	y'nin $\mathbf{X}$ 'e göre türevini içeren tensör
$\dfrac{\partial f}{\partial \boldsymbol{x}}$	Jacob matrisi $\boldsymbol{J} \in \mathbb{R}^{m\times n}$'nin $f : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ veya $\boldsymbol{H}(f)(\boldsymbol{x})$	f'in $\boldsymbol{x}$ girdi noktasındaki Hesse matrisi
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Tüm $\boldsymbol{x}$ alanında tanımlı olan belirli integral
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	$\mathbb{S}$ kümesinde, $\boldsymbol{x}$'e göre alınan belirli integral

Olasılık ve Bilgi Teorisi

$\mathrm{a} \perp \mathrm{b}$	a ve b rastgele değişkenleri bağımsızdır
$\mathrm{a} \perp \mathrm{b} \mid \mathrm{c}$	a ve b, c verildiğinde koşullu olarak bağımsızdır
$P(\mathrm{a})$	Ayrık bir değişken üzerindeki olasılık dağılımı
$p(\mathrm{a})$	Sürekli bir değişken üzerindeki veya tipi belirtilmemiş bir değişken üzerindeki olasılık dağılımı
$\mathrm{a} \sim P$	Rastgele değişken a'nın dağılımı P'dir
$\mathbb{E}_{\mathrm{x}\sim P}[f(x)]$ or $\mathbb{E} f(x)$	$P(\mathrm{x})$'ye göre $f(x)$'in beklentisi
$\mathrm{Var}(f(x))$	$P(\mathrm{x})$ altında $f(x)$'in değişirliği
$\mathrm{Cov}(f(x), g(x))$	$P(\mathrm{x})$ altında, $f(x)$ ve $g(x)$'in eşdeğişirliği
$H(\mathrm{x})$	x rastgele değişkeninin Shannon entropisi
$D_{\mathrm{KL}}(P\|Q)$	P ve Q'nun Kullback-Liebler ıraksaması
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\boldsymbol{x}$ üzerindeki Gauss dağılımı, ortalaması μ ve eşdeğişirliği $\boldsymbol{\Sigma}$'dir

Fonksiyonlar

$f : \mathbb{A} \to \mathbb{B}$	Tanım kümesi A ve değer kümesi B olan fonksiyon
$f \circ g$	f ve g fonksiyonlarının bileşimi
$f(\boldsymbol{x}; \boldsymbol{\theta})$	$\boldsymbol{\theta}$ ile parametrelendirilmiş olan $\boldsymbol{x}$'in bir fonksiyonu. (Gösterimi hafifletmek amacıyla bazen $f(\boldsymbol{x})$ yazıp $\boldsymbol{\theta}$'yı gösterime katmayız)
$\log x$	x'in doğal logaritması
$\sigma(x)$	Lojistic sigmoid, $\dfrac{1}{1+\exp(-x)}$
$\zeta(x)$	Softplus, $\log(1+\exp(x))$
$\|\|\boldsymbol{x}\|\|_p$	$\boldsymbol{x}$'in L^p normu
$\|\|\boldsymbol{x}\|\|$	$\boldsymbol{x}$'in L^2 normu
x^+	x'in pozitif kısmı, yani $\max(0, x)$
$\mathbf{1}_{\text{condition}}$	Koşul (condition) doğru ise 1, aksi takdirde 0 verir.

Bazen argümanı skaler olan bir f fonksiyonunu alıp, bu fonksiyonu bir vektöre, bir matrise veya bir tensöre uygularız: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, or $f(\mathsf{X})$. Bu, f fonksiyonunun eleman bazında uygulandığını göstermektedir. Örneğin, eğer $\mathsf{C} = \sigma(\mathsf{X})$ olursa, o zaman geçerli tüm i, j and k için $\mathsf{C}_{i,j,k} = \sigma(\mathsf{X}_{i,j,k})$ olur.

Veri Kümeleri ve Dağılımlar

p_{data}	Veri üretim dağılımı
$\hat{p}_{\text{data}}$	Eğitim kümesi tarafından tanımlanan deneysel dağılım
$\mathbb{X}$	Eğitim örnekleri kümesi
$\boldsymbol{x}^{(i)}$	Veri kümesinin i-inci örneği (girdisi)
$y^{(i)}$ veya $\boldsymbol{y}^{(i)}$	Denetimli öğrenmede $\boldsymbol{x}^{(i)}$ ile ilişkilendirilen hedef
$\boldsymbol{X}$	$\boldsymbol{X}_{i,:}$ satırında $\boldsymbol{x}^{(i)}$ girdi örneğini barındıran $m \times n$ boyutlu matris

1

Giriş

Düşünebilen makineler yapabilmek, uzun zamandan beri mucitlerin hayallerinde yer almıştır. Bu hayalleri en az antik Yunan'a kadar gitmektedir. Belki, Pygmalion, Daedalus ve Hephaestus gibi mitik figürler, efsanevi mucitler olarak görülebilir. Galatea, Talos ve Pandora yapay hayat olarak nitelendirilebilir. (Ovid ve Martin, 2004; Sparkes, 1996; Tandy, 1997).

Programlanabilen bilgisayarlar yüzyılı aşkın bir süre önce ilk geliştirildiklerinde, insanlar bu makinelerin zekâya sahip olup olamayacaklarını merak ettiler (Lovelace, 1842). Bugün ise **yapay zekâ** (YZ), birçok uygulaması ve aktif olarak araştırılan çokça konusu bulunan gelişmekte olan bir bilim dalına dönüşmüştür. Günümüzde otomasyondan, konuşmaya, görüntü anlamaya, tıbbi tanıdan temel bilim araştırmlarına kadar birçok alanda akıllı yazılımlardan yardım almaktayız.

Yapay zekânın erken dönemlerinde, insanların güçlükle çözebildiği ama bilgisayarlar için nispeten sıradan, yani biçimsel ya da matematiksel kurallarla tanımlanabilen problemler hızla çözüldü. Ancak gerçek sorunun, insanlar için kolay ama biçimsel ya da matematiksel olarak tanımlanması zor olan, insan yüzü ya da konuşma tanıma gibi günlük hayatta sürekli çözdüğümüz problemler olduğu ortaya çıktı.

Bu kitap göreceli olarak daha sezgisel problemleri çözmek için önerilen bir yaklaşım üzerinedir. Bu yaklaşım bilgisayarların her kavramı, daha temel kavramlarla ilişkilendirerek, bir kavramlar hiyerarşisi öğrenmesini sağlar. Böylece, bilgisayarların gereksinim duyduğu biçimsel kuralların insan eliyle girilmesine ihtiyaç kalmaz. Kavramlar hiyerarşisi bilgisayarın karmaşık kavramları daha basit kavramlardan yola çıkarak kurmasına olanak sağlar. Bu kavramların birbiri üzerine nasıl inşa edildiğini gösteren bir çizge oluşturursak, bu çizge çok katmanlı, bir başka deyişle derin bir çizge olur. İşte bu yüzden biz, bu yaklaşıma **derin öğrenme** diyoruz.

Başlangıçta yapay zekâ algoritmalarının birçok başarısı, bilgisayarların gerçek problemlerle ilgili olmayan kısmen daha steril ve düzenli ortamlarda elde edildi. Örneğin, IBM'nin Deep Blue adındaki satranç oynama sistemi dünya şampiyonu Garry Kasparov'u 1997 yılında mağlup etti (Hsu, 2002). Elbette satranç yalnızca 64 kare ve 32 parçadan oluşan ve parçaların sıkı kurallara bağlı olarak hareket edebildiği oldukça basit bir oyundur. Başarılı bir satranç stratejisi oluşturmak önemli bir başarıdır ancak bu problemin zorluğu, satranç taşlarını ve geçerli hamleleri bilgisayara anlatmakta yatmamaktadır. Satranç kuralları, programcı tarafından bilgisayara biçimsel kurallardan oluşan kısa bir liste olarak kolayca verilebilir.

İlginç bir şekilde, insanlar için en zor zihinsel aktivitelerden biri olan soyut ve biçimsel problemler bir bilgisayar için en kolay işler arasında yer alır. Bilgisayarlar uzun zamandır en iyi satranç oyuncularını yenebilmektedir ancak yakın bir zamanda insanların nesneleri tanıma veya konuşma yeteneklerine erişebilmiştir. Günlük hayatta bir insan dünya hakkında çok fazla bilgiye ihtiyaç duyar. Bu bilginin çoğu öznel ve sezgiseldir ve bu nedenle biçimsel olarak ifade edilmesi güçtür. Akıllı bilgisayarlar geliştirebilmek için bu tür bilgilerin bilgisayarlara yüklenmesi gerekir. Yapay zekâdaki en önemli güçlüklerden biri de biçimsel olmayan bu bilginin bilgisayara nasıl aktarılacağıdır.

Geçmişte çeşitli yapay zekâ projelerinde gerçek hayat bilgileri, biçimsel dillerde kodlanmaya çalışıldı. Teorik olarak, bir bilgisayarın mantıksal çıkarım kuralları kullanarak bu biçimsel dillerde otomatik olarak akıl yürütmesi mümkündü. Bu yapay zekâ alanında **knowledge base** yaklaşımı olarak bilinmektedir. Ancak bu projelerden hiçbiri büyük bir başarı yakalayamadı. Bu projelerin en ünlülerinden biri olan Cyc Cyc (Lenat ve Guha, 1989) veri bankasında CyCL dilini kullanılarak yazılmış ifadeler içeren bir çıkarım motoruydu. Veri tabanındaki ifadeler araştırmacılar tarafından giriliyordu ve bu problemli bir süreçti. Dünyayı doğru bir şekilde tarif etmek için yeterli karmaşıklıkta biçimsel kurallar üretmek zorluydu. Örneğin Cyc, Fred adındaki bir adamın sabahları traş olması ile ilgili bir öyküyü anlamayamıştı. Çıkarım motoru hikâyede çelişki buluyordu. Cyc insanların elektriksel parçaları olmadığını biliyordu fakat Fred elektrikli traş makinesini tuttuğunda, Cyc "traş olan Fred" nesnesinin elektriksel parçalar içerdiği kanısına vardı. Sonuç olarak traş olan Fred'in traş sırasında hâlâ insan olup olmadığından emin değildi.

Sabit kodlu bilgiye dayanan sistemlerin karşılaştığı zorluklar, YZ sistemlerinin ham veriden örüntüler çıkararak kendi bilgilerini elde etme kabiliyetine ihtiyaç duyduklarını ortaya koymaktadır. Bu kabiliyet **makine öğrenmesi** olarak bilinir. Makine öğrenmesi, gerçek dünyadaki bilgileri içeren problemlerle baş etmeye ve öznel görünen kararlar vermeye olanak sağladı. Örneğin, basit bir makine öğrenme

algoritması olan **lojistik bağlanım** sezaryen ameliyat yapılıp yapılmaması konusunda karar verebilir. Bir diğer basit makine öğrenme algoritması olan **Naif Bayes**, gerçek e-maillerle spamları ayırt edebilir.

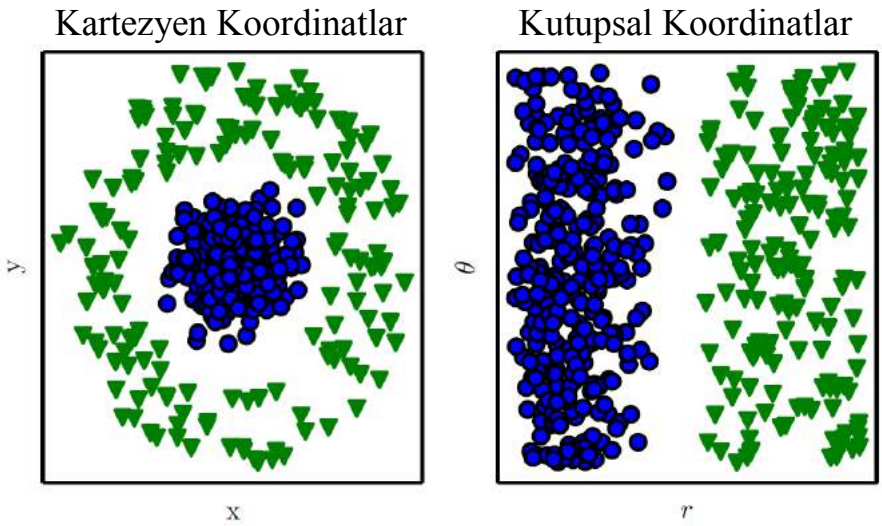
Basit makine öğrenme algoritmalarının performansı girdi verilerinin **gösteriminin** kalitesinden büyük ölçüde etkilenir. Örneğin, sezaryen doğumlara karar vermek için lojistik bağlanım kullanıldığında, yapay zekâ hastayı muayene etmez. Bunun yerine doktorlar sisteme rahimde yara olup olmadığı gibi annenin durumuyla ilgili birtakım bilgiler iletir. Hastanın gösteriminde yer alan her bilgi parçası birer **öznitelik** olarak bilinir. Lojistik bağlanım, hastanın bu özelliklerinin her birinin çeşitli sonuçlar ile nasıl bağlantılı olduğunu öğrenir. Bununla birlikte model, özniteliklerin nasıl tanımlandığını herhangi bir şekilde belirtemez. Lojistik bağlanıma doktorun hazırladığı biçimsel rapor yerine bir MR taraması verilseydi, modelin bu görüntülerden herhangi mantıklı birer tahmine varması mümkün olmazdı. Doğum sırasında oluşabilecek komplikasyonlarla MR üzerindeki tek bir piksel arasındaki bağlantı ise önemsenmeyecek kadar düşüktür.

Gösterimlere olan bu bağımlılık, bilgisayar bilimleri ve hatta günlük yaşam boyunca ortaya çıkan genel bir olgudur. Bilgisayar bilimlerinde, bir veri kolleksiyonunu aramak gibi işlemler, koleksiyon akıllıca yapılandırılmış ve endekslenmişse, kat kat daha hızlı gerçekleştirilebilir. İnsanlar kolayca Arap rakamları üzerinde aritmetik işlem yapabilir, ancak Roma rakamları üzerinde aritmetik işlem yapmak çok daha zaman alıcıdır. Öznitelik seçiminin, makine öğrenme algoritmalarının performansı üzerinde çok büyük bir etkisi olması şaşırtıcı değildir. Görsel bir örnek için Şekil 1.1'e bakınız.

Pek çok yapay zekâ problemi, bu problem için gereken doğru öznitelik kümesini tasarlayarak ve daha sonra bu öznitelikleri basit bir makine öğrenme algoritmasına ileterek çözülebilir. Örneğin, konuşmacının ses yolunun uzunluğu, konuşan kişiyi tanımak için kullanışlı bir özniteliktir. Bu öznitelik, konuşmacının erkek mi, kadın mı ya da çocuk mu olduğuna dair güçlü bir ipucu verir.

Bununla birlikte, birçok problem için hangi özniteliklerin toplanması gerektiğini bilmek güçtür. Örneğin, fotoğraflarda araba bulmak için bir program yazmak istediğimizi varsayalım. Arabaların tekerlekleri olduğunu biliyoruz, bu nedenle tekerleklerin varlığını bir özellik olarak kullanmak isteyebiliriz. Ne yazık ki, bir tekerleğin piksel değerleri açısından tam olarak neye benzediğini açıklamak zordur. Bir tekerlek basit bir geometrik şekle sahiptir, ancak tekerleğin görüntüsü tekerleğe düşen gölgeler, tekerleğin metal parçalarından yansıyan güneş, arabanın çamurluğu veya tekerleğin önünü kapatan nesneler vb. gibi sebeplerden oldukça karmaşık olabilir.

Bu sorunun bir çözümü, yalnızca gösterimden çıktıya eşlemenin değil, gösterim-



Şekil 1.1: Farklı türde gösterimlerin bir örneği: Bir dağılım grafiğinde gösterilen veriyi bir çizgi çizerek iki kategoriye ayırmaya çalıştığımızı düşünelim. Soldaki çizitte bir veriyi Kartezyen koordinat sistemi kullanarak gösteriyoruz. Böyle bir problemi çözmemiz imkânsızdır. Sağdaki çizitte veriyi kutupsal koordinatlarla gösteriyoruz. Bu durumda problem basit dikey bir doğruyla kolayca çözülebilir hâle geliyor.(Çizge David Warde-Farley ile birlikte üretilmiştir.)

den kendisinin de elde edilmesi için makine öğrenimini kullanmaktır. Bu yaklaşım, **gösterim öğrenme** olarak bilinmektedir. Öğrenilen gösterimler genellikle elle tasarlanmış gösterimlerle elde edilebileceğinden daha iyi performans gösterirler. Aynı zamanda, YZ sistemlerinin en az insan müdahalesi ile yeni görevlere hızla adapte olmasını sağlarlar. Bir gösterim öğrenme algoritması, basit bir görev için iyi bir dizi özelliği basit bir problem için birkaç dakika içinde veya daha karmaşık bir problem için birkaç ayda keşfedebilir. Karmaşık bir problem için özellikleri elle tasarlamak, çok fazla zaman ve çaba gerektirir; böyle bir problem bir araştırma grubunun onlarca yılını alabilir.

Otokodlayıcılar gösterim öğrenmenin özünü anlatan bir örnek oluştururlar. Otokodlayıcılar, girilen veriyi farklı bir gösterime dönüştüren bir **kodlayıcı** fonksiyonuyla, bu dönüştürülmüş gösterimi tekrar orijinal hâline geri çeviren **çözücü**den oluşurlar. Otokodlayıcılar verilen girdiyle ilgili olabilecek en fazla bilgiyi korumak için eğitilirler, ancak bunun yanında oluşturduklar yeni gösterimin belli özellikleri

sağlamasını da dikkate alırlar. Farklı otokodlayıcılar farklı özellikleri elde etmeye çalışır.

Öznitelik algoritmaları tasarlarken, amacımız genellikle gözlemlenen verileri açıklayan **varyasyon etkenlerini** gruplamaktır. Bu bağlamda, "etkenler" kelimesini basitçe ayrı etkileşim kaynaklarını belirtmek için kullanıyoruz; buradaki faktörler genellikle çarpılarak birleştirilmezler. Bu tür etkenler nadiren doğrudan gözlemlenebilen niceliklerden oluşur.

Bu etkenler, fiziksel dünyada gözlemlenebilen değerleri etkileyen gözlemlenmemiş nesne ya da etkilere karşı gelebilir, ya da insan zihnindeki gözlemlerin basit açıklaması veya nedeni hakkında çıkarsama yapılmasını sağlayan yapılar olabilirler. Bu etkenler, verilerdeki değişikliği anlamamıza yardım eden kavram ve soyutlamalara da karşı gelebilir. Varyasyon etkenleri bir konuşma kaydında konuşmacıların yaşını, cinsiyetini, vurgularını ve kullandıkları kelimeleri içerirler. Örneğin, bir otomobil görüntüsünde, aracın rengini, parlaklığını ve ışığın açısını içermektedir.

Yapay zekâ uygulamalarının gerçek hayattaki zorluklarının büyük bir kısmı varyasyon etkenlerinin çoğunun gözlemleyebildiğimiz her veri noktasını etkilemesidir. Kırmızı bir araba resminin piksel değerleri geceleri siyah renge yakın olabilir. Arabanın silueti bakılan açıya bağlıdır. Çoğu uygulama bu etkenleri birbirinden ayırmamızı ve işimize yaramayanları elden çıkarmamızı gerektirir.

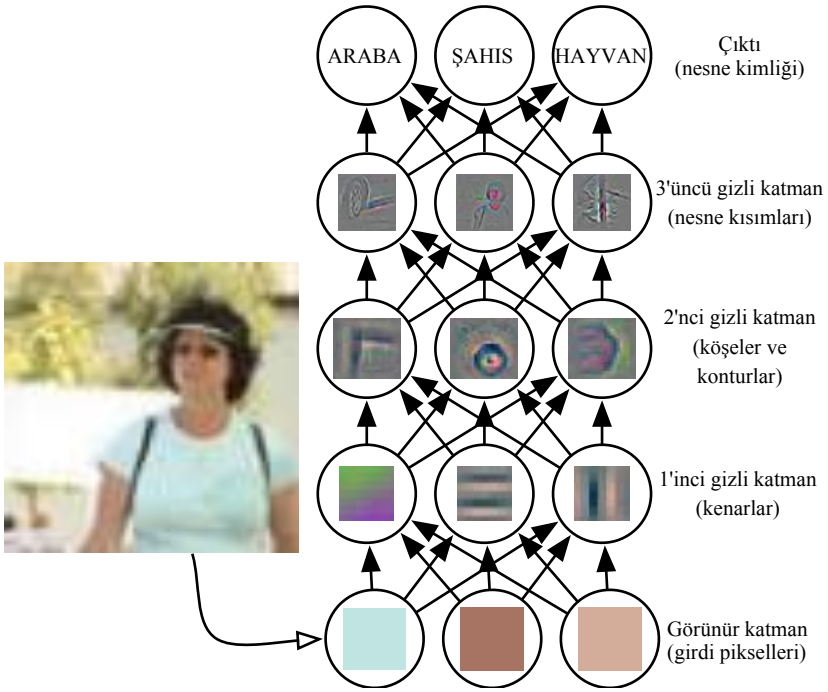
Elbette bu kadar yüksek seviyeli ve soyut öznitelikleri ham veriden elde etmek zordur. Bu varyasyon etkenlerinden çoğu, bir konuşmacının aksanı gibi yalnızca insan seviyesinde bir anlama yeteneği ile tespit edilebilir. Gösterimleri elde etmenin neredeyse orijinal problemi çözmek kadar zor olduğu göz önünde bulundurulduğunda, öğrenme yöntemi pek de yardımcı olmuyor gibi gözükebilir.

İşte **derin öğrenme** gösterim öğrenmedeki ana problemi gösterimleri daha temel gösterimlerle ifade ederek çözer. Derin öğrenme bilgisayarların basit kavramlardan daha karmaşık kavramlar kurmasına olanak sağlar.

Şekil 1.2'de bir derin öğrenme sisteminin kenarlardan, köşe ve kontur gibi basit kavramları, bu kavramları da birleştirerek daha karmaşık bir insan resminin gösterimini oluşturması gösterilmektedir.

İleri beslemeli derin ağlar ya da **çok katmanlı perseptron (multilayer perceptron, MLP)** derin öğrenmenin özünü anlatan bir modeldir. Çok katmanlı perseptron, girdileri çıktılara eşleyen matematiksel bir fonksiyondan ibarettir. Bu fonksiyon birden çok basit fonksiyonun birleşmesiyle oluşturulur. Farklı fonksiyonların her uygulanışı girdinin yeni bir gösterimi olarak düşünülebilir.

Veriler için en iyi gösterimi öğrenme fikri, bize derin öğrenmeyle ilgili bir bakış



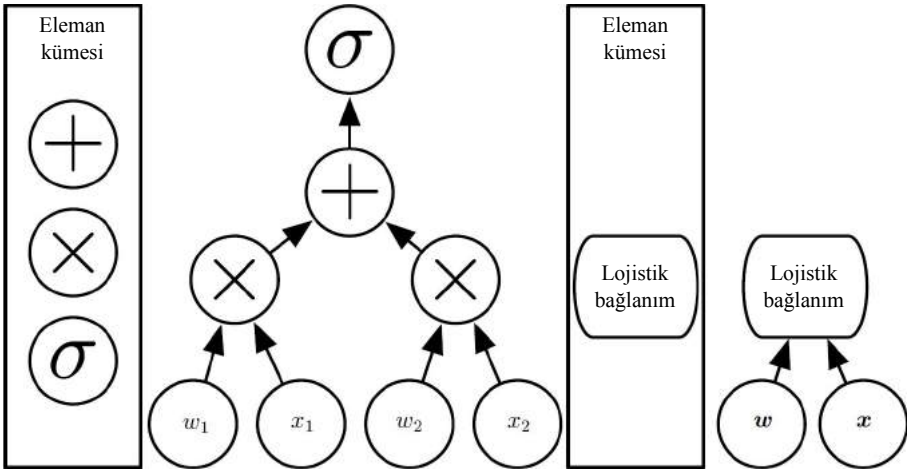
Şekil 1.2: Bir derin öğrenme modelinin gösterimi: Bir görüntünün piksel değerleri gibi ham sensör verisinin anlaşılması bilgisayarlar için zordur. Piksel değerlerini nesnenin kimliğine eşleyen fonksiyon oldukça karmaşıktır. Bu eşlemeyi doğrudan öğrenmek ya da değerlendirmek imkânsız gözükmektedir. Derin öğrenme bu karmaşık eşlemeyi her biri farklı katman olarak tanımlanmış iç içe geçmiş ve daha basit eşlemeye ayırarak çözer. Girdi modele **görünür katman** adı verilir. Bu katmanın görülebilir katman olarak isimlendirilmesinin sebebi içinde gözlemleyebildiğimiz değişkenler tutmasıdır. Daha sonra bir dizi **gizli katman** görüntüden giderek daha da soyutlaşan öznitelikler çıkarırlar. Bu katmanların "gizli" olarak isimlendirilmesinin sebebiyse bu katmanların değerlerinin veri tarafından verilmemesidir. Bunun yerine model, hangi kavramların verinin içindeki ilişkileri açıklamada daha kullanışlı olduğuna karar vermelidir. Buradaki görüntüler her bir gizli birim tarafından gösterilen öznitelik türünün görselleştirilmesidir. Pikseller verildiğinde, birinci katman komşu piksellerin parlaklıklarını karşılaştırarak kenarları kolayca tanımlayabilir. İlk gizli katmanın tanımlaması verildiğinde ise ikinci gizli katman, köşeleri ve konturları, yani kenarların oluşturduğu kümeleri kolayca arayabilir. İkinci gizli katmanın çıktısı verildiğinde, üçüncü katman buradaki köşe ve kontur kümelerini kullanarak nesnelerin daha büyük parçalarını tanımlayabilir. Son olarak resmin içindeki nesne parçalarını içeren bu tanım, görüntü içindeki nesneleri tanımak için kullanılabilir. Buradaki görüntüler Zeiler ve Fergus (2014)'un izniyle çoğaltılmıştır.

açısı sunar. Derin öğrenme üzerine bir diğer bakış açısı ise derinliğin bilgisayarın

çok basamaklı bir bilgisayar programı öğrenmesini sağlamasıdır. Gösterimin her katmanı, bir dizi talimatı paralel olarak yerine getiren bir bilgisayarın bellek durumu olarak düşünülebilir. Daha derin ağlar, art arda daha fazla talimat yürütebilir. Art arda gelen talimatlar modele büyük bir güç sunar. Çünkü daha sonraki talimatlar kendinden önceki talimatların sonuçlarına geri dönebilir veya onları kullanabilirler.

Derin öğrenme bakış açısına göre bir katmanın aktifleşmelerindeki bilginin tamamı, girdiyi açıklayan varyasyon faktörlerini kodlamaz. Buradaki gösterimler aynı zamanda girdiden anlam çıkarmaya olanak sağlayan bir programın yürütülmesine yardımcı olan durum bilgisini de depolar. Durum bilgisi, geleneksel bir bilgisayar programındaki bir sayaç veya işaretçiye benzeyebilir. Bu bilginin girdi içeriğiyle özel olarak bir ilgisi yoktur, ancak modelin işlemlerini organize etmesine yardımcı olur.

Bir modelin derinliğini ölçmenin iki temel yöntemi vardır. İlki model mimarisinin değerlendirilmesi için gereken sıralı işlemlerin sayısına dayalıdır. Bu yöntemi bir modelin çıktılarını nasıl hesaplayacağını gösteren bir akış çizelgesindeki en uzun



Şekil 1.3: Her düğüm noktasının bir işlemi gösterdiği, bir girdiyi bir çıktıya eşleyen bir çizgelerin gösterimi. Derinlik girdiden çıktıya olan en uzun yolun mesafesi olarak tanımlanır. Ancak burada neyin bir işlem basamağı olarak kabul edildiği önemlidir. Bu görüntüde gösterilen hesaplama $\sigma(\boldsymbol{w}^T\boldsymbol{x})$ olarak verilen bir lojistik bağlanım modelinin çıktısıdır. σ burada lojistik sigmoid fonksiyonudur. Eğer toplama, çarpma ve lojistik sigmoidi programlama dilimizin temel elemanları olarak kullanırsak, bu modelin derinliği üç birimdir. Eğer lojistik bağlanımı tek bir eleman olarak kabul edersek modelin derinliği bu durumda bir birimdir.

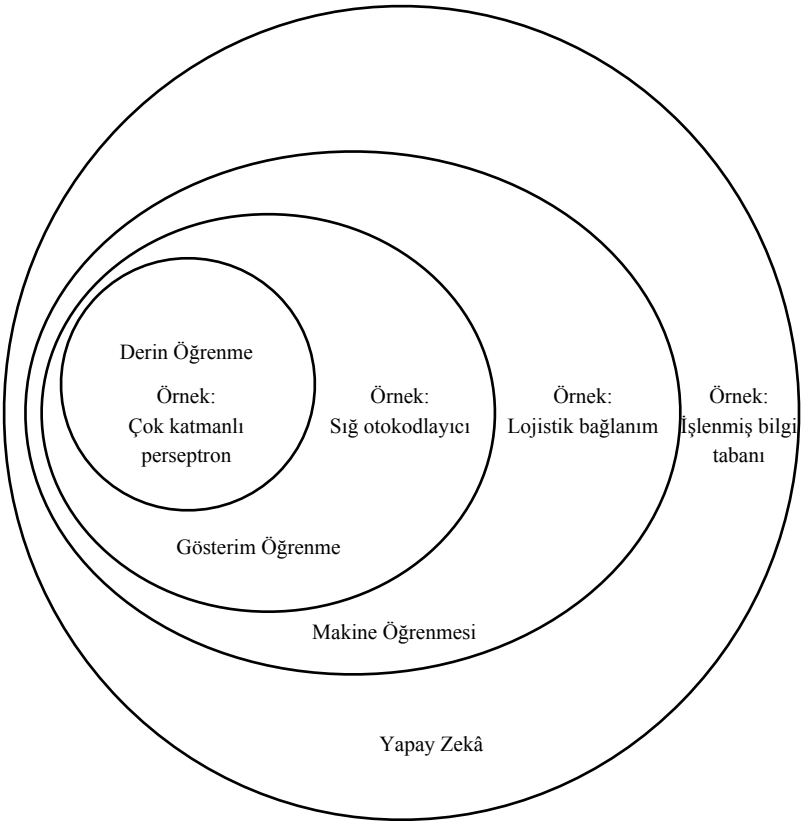
yolun boyunu bulmak olarak da düşünebiliriz. Nasıl ki farklı dillerde yazılan iki eş değer program farklı uzunluklara sahip oluyorsa, bu akış çizelgesinde de aynı fonksiyonu, işlem adımlarına bağlı olarak farklı derinliklerde çizebiliriz. Şekil 1.3 farklı diller kullanmanın aynı mimari üzerinde nasıl farklı ölçümlere yol açabileceğini göstermektedir.

Derin olasılıksal modeller tarafından kullanılan başka bir yaklaşım, model derinliğini hesaplama çizgesinin derinliği yerine kavramların birbirine olan ilişkileri gösteren bir çizgenin derinliği olarak kabul eder. Bu durumda, her kavramı hesaplamak için gerekli olan işlemleri içeren akış çizelgesi, kavramları ilişkilendiren çizelgeden çok daha derin olabilir. Bunun nedeni sistemin daha temel kavramlar hakkındaki anlayışının, daha karmaşık kavramlarla ilgili bilgileri kullanarak artırılabilmesidir. Örneğin, bir gözünün gölge altında olduğu bir yüz görüntüsünü inceleyen bir YZ sistemi başlangıçta yalnızca tek bir göz görebilir. Yüzün varlığını belirledikten sonra, sistem bu bilgiyi kullanarak, ikinci gözün de büyük olasılıkla orada olduğunu çıkarabilir. Bu durumda kavramlar çizgesi, gözler ve yüz için olmak üzere yalnızca iki katmandan oluşur. Ancak hesaplama çizgesini, her kavram için n adet olarak yeniden hesaplarsak, $2n$ adet katman içerecektir.

Bu iki yaklaşımdan hangisinin daha uygun olduğu her zaman açık değildir. Bir bilgisayar programının uzunluğunu belirleyen tek bir yanıt olmadığı gibi bir mimarinin de derinliği için doğru bir cevap yoktur. Zira, farklı kişiler çizgeleri oluştururken farklı temel elemanlar seçebilirler.

Farklı kişilerin çizelgelerini oluştururken farklı temel elemanlar seçmesinden dolayı nasıl ki bir bilgisayar programının uzunluğu için tek bir doğru cevap yoksa, verilen bir mimarinin derinliği için de doğru bir cevap yoktur. Bir modelin "derin" olarak kabul edilmesi için ne kadar derin olması gerektiğine dair de bir fikir birliği bulunmamaktadır. Yine de, derin öğrenme, geleneksel makine öğrenimi algoritmalarından daha fazla fonksiyon ya da kavram içeren modellerin incelenmesi olarak kabul edilebilir.

Özetle, bu kitabın konusu olan derin öğrenme, yapay zekâ yaklaşımlarından biridir. Özellikle bilgisayarların veri ve deneyimlerden yararlanarak kendilerini geliştirdikleri bir makine öğrenmesi türüdür. Biz bu tür makine öğrenmesinin, gerçek dünya problemlerini çözebilecek bir yapay zekâ inşa etmenin tek gerçekleştirilebilen yegane yolu olduğunu savunuyoruz. Derin öğrenme, dünyayı iç içe geçmiş kavramlar hiyerarşisi olarak göstererek büyük güce ve esnekliğe sahip bir makine öğrenmesi yaklaşımıdır. Bu yaklaşımda, her kavram kendisinden daha basit olan kavramlara olan ilişkleriyle tanımlanır ve daha soyut gösterimler kendilerinden daha somut gösterimler kullanılarak hesaplanır. Şekil 1.4 farklı YZ disiplinleri arasındaki ilişkiyi göstermektedir. Şekil 1.5 ise bu disiplinlerin nasıl çalıştığına dair genel bir görüntü

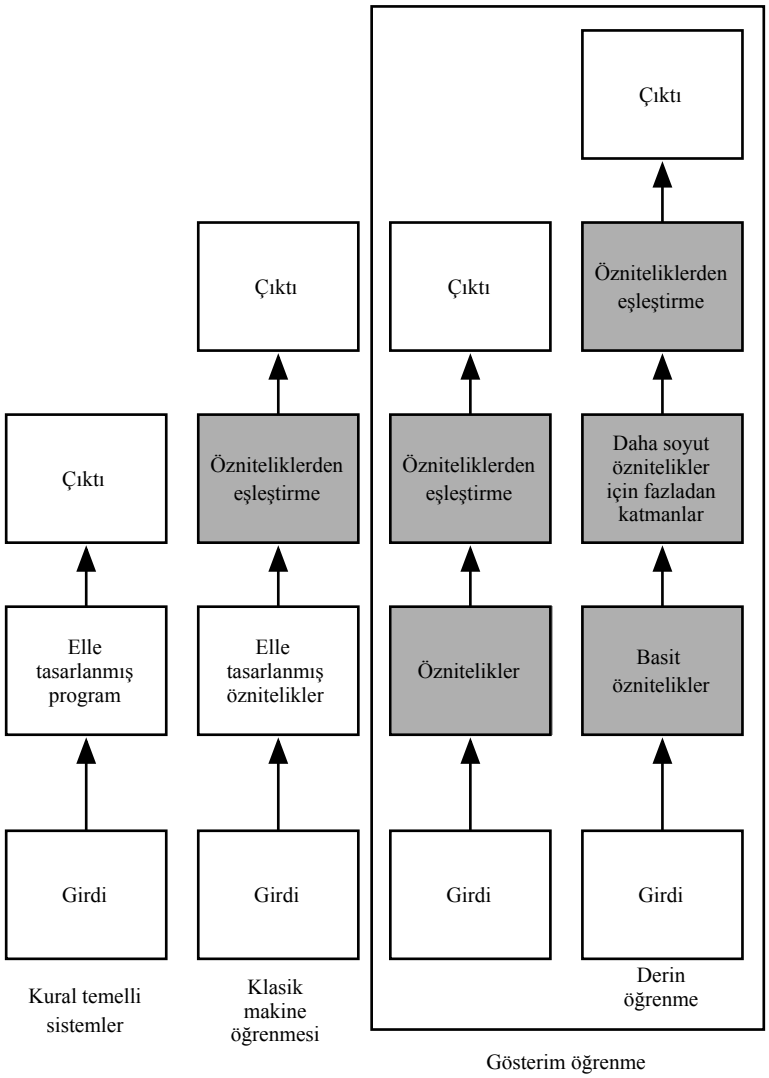


Şekil 1.4: Derin öğrenmenin gösterim öğrenmenin bir türü olduğunu gösteren Venn şeması. Şema aynı zamanda derin öğrenmenin yapay zekânın birçok yaklaşımında kullanılan makine öğrenmesinin bir türü olduğunu göstermektedir. Venn şemasının her bölümü YZ teknolojilerinden bir örneği içermektedir.

vermektedir.

1.1 Bu Kitabı Kimler Okumalı?

Bu kitap farklı türdeki okuyucular için kullanışlı olabilir. Ancak biz bu kitabı temelde iki hedef kitle için yazdık. Bu hedef kitlelerden ilki, makine öğrenmesi çalışan ya da derin öğrenme ve yapay zekâ araştırmasında bir kariyere başlayacak üniversite öğrencileridir(lisans veya yüksek lisans). Diğer hedef kitlesiyse, makine öğrenmesi ya da istatistik arka planına sahip olmayan ancak derin öğrenmeyi kendi platformlarında kullanmak isteyen yazılım mühendisleridir. Derin öğrenme



Şekil 1.5: Farklı YZ disiplinlerindeki YZ sistemlerinin birbirleriyle olan ilişkilerini gösteren bir akış şeması. Gölgelendirilmiş kutular, veriden öğrenebilen parçaları göstermektedir.

hâlihazırda, bilgisayarlı görü, konuşma ve ses işleme, doğal dil işleme, robotik, biyoenformatik, kimya, bilgisayar oyunları, arama motorları, çevrimiçi reklam ve finans gibi birçok yazılım disiplininde kendini ispatlamıştır.

Kitap farklı türde okuyuculara hitap etmek üzere 3 bölüme ayrılmıştır. Kısım I, makine öğrenmesiyle ilgili kavramlar ve temel matematiksel araçları kapsamaktadır.

Kısım II, iyi bilinen ve anlaşılmış yani, temelinde çözülmüş derin öğrenme algoritmalarını içermektedir. Son bölüm olan Kısım III ise derin öğrenme araştırmasında önemli yer tuttukları düşünülen daha kuramsal ve yeni fikirlerin toplandığı bir bölümdür.

Okuyucular, kendi ilgi veya akademik geçmişleriyle alakalı olmadığını düşündükleri kısımları atlamada bir sakınca görmemelidirler. Örneğin, lineer cebir, olasılık ve temel makine öğrenmesi kavramlarına hâlihazırda aşina olan okuyucular, Kısım I'i atlayabilirler. Yalnızca çalışan bir derin öğrenme sistemi kurmak isteyenlerin ise Kısım II'ye kadar okumaları yeteri olacaktır.

Bütün okuyucuların bilgisayar mühendisliği ya da bilgisayar bilimleriyle ilgili bir arka plana sahip olduklarını varsayıyoruz. Aynı zamanda, programlama, hesaplama performansıyla ilgili problemler, karmaşıklık kuramı, giriş seviyesinde matematik ve çizge kuramına ait birkaç terime hâkim olduklarını göz önünde bulunduruyoruz.

1.2 Derin Öğrenmede Tarihsel Eğilimler

Derin öğrenmeyi anlamanın en iyi yollarından biri tarihsel süreci göz önüne almaktır. Burada detaylı bir derin öğrenme tarihi vermektense, derin öğrenmedeki önemli eğilimleri tanımlıyoruz:

- Derin öğrenme uzun ve zengin bir tarihe sahiptir, ancak isim olarak tarih boyunca farklı felsefi bakış açılarına göre değişmiş, popülerliği artıp azalmıştır.
- Derin öğrenme kullanılabilir eğitim verisinin artmasıyla daha kullanışlı bir hâle gelmiştir.
- Derin öğrenme, model boyutları arttıkça ve derin öğrenme donanımları ve yazılımları geliştikçe yaygınlaşmıştır.
- Derin öğrenme, geçen zamanla birlikte daha karmaşık problemleri daha isabetli olarak çözebilmeye başlamıştır.

1.2.1 Sinir Ağlarının Sayısız İsmi ve Değişen Kaderleri

Birçok okur derin öğrenmenin yeni ortaya çıkan ilginç bir teknoloji olduğunu düşünüp, bu yüzden bir kitabın içinde bir derin öğrenme "tarihi" bölümü olmasına şaşırcaklardır. Gerçekte derin öğrenme 1940'lı yıllardan beri var olan bir alandır. Derin öğrenmenin yeni olarak algılanmasının sebebi hem birkaç yıl öncesine kadar popüler olmayan bir metot olması hem de "derin öğrenme" ismini almadan önce

farklı isimlerle anılmış olmasıdır. Bu alan, farklı araştırmacıların ve bakış açılarının etkisinde tarih boyunca yeniden isimlendirilmiştir.

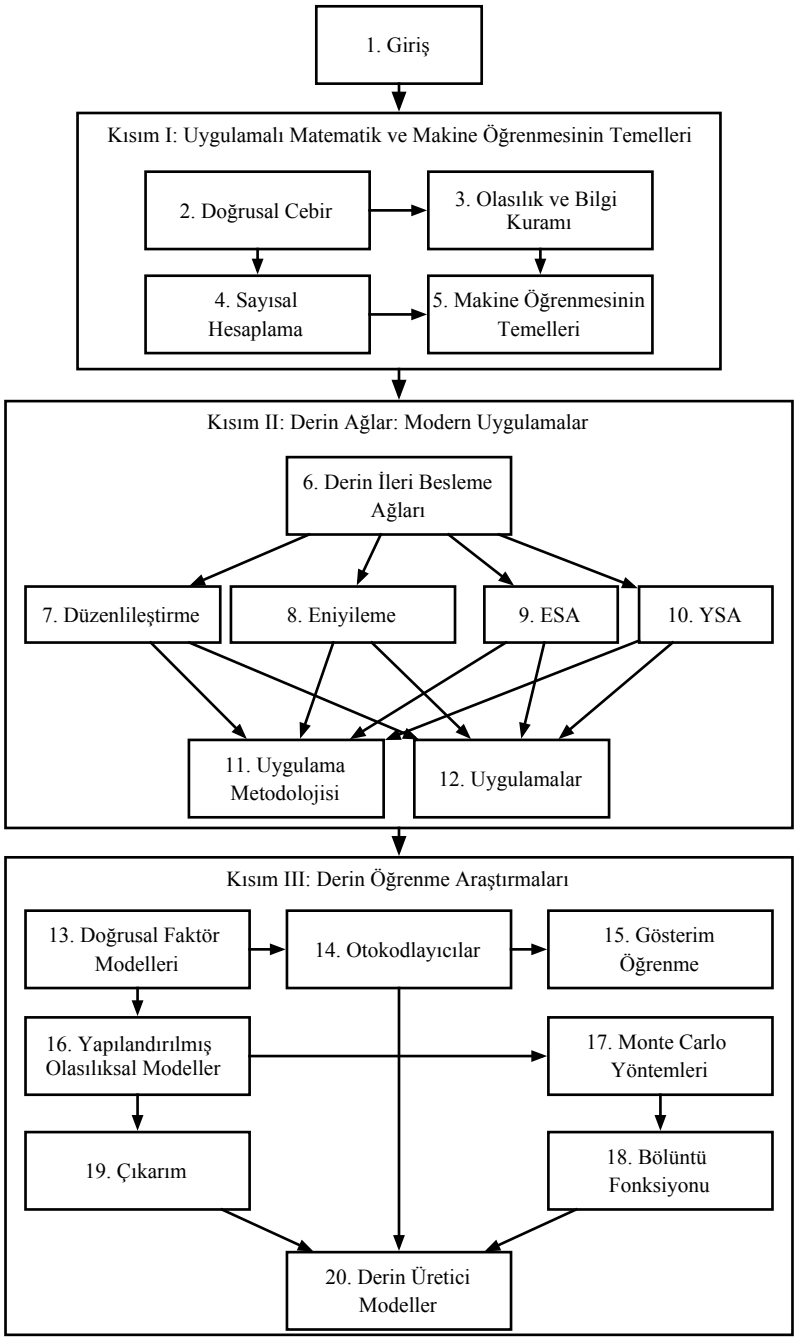
Derin öğrenmenin kapsamlı tarihi bu kitabın amacının ötesindedir, ancak temel seviyede bir açıklama derin öğrenmeyi anlamamız için önemlidir. Genel olarak değerlendirildiğinde, derin öğrenmenin 3 dalgadan oluşan bir gelişme dönemi vardır. Bunlar, **sibernetik** adıyla bilinen ve 1940'lı yıllardan 1960'lı yılları kapsayan, **bağlantıcılık** olarak bilinen ve 1980'li yıllardan 1990'lı yılları kapsayan, ve son olarak da 2006'da ortaya çıkan ve derin öğrenme dediğimiz dönemlerdir. Bu dönemler Şekil 1.7'de gösterilmiştir.

Bugün kabul ettiğimiz birçok erken öğrenme algoritması, aslında temel biyolojik öğrenme modelleri olarak geliştirilmiştir. Bu modellern amacı öğrenmenin nasıl gerçekleştiği ya da beyinde gerçekleşip gerçekleşmediğini açıklamaktır. Bu nedenle derin öğrenmenin isimlerinden biri de **yapay sinir ağları** (YSA'lar)dır. Derin öğrenme modellerinde buna karşılık gelen bakış açısı, bu modellerin biyolojik beyinlerden (insan ya da hayvan beyni) esinlenilerek geliştirilmiş sistemler olduğudur. Sinir ağları bazı durumlarda beyin fonksiyonlarını anlamak için kullanılsa da (Hinton ve Shallice, 1991), genelde biyolojik fonksiyonları gerçekçi olarak modellemek için tasarlanmamıştır. Derin öğrenmedeki sinirsel bakış açısı iki ana fikre dayanır. Fikirlerden biri, beynin zeki davranışlara dair bir örnek gösterdiği ve zekâ yaratmanın kavramsal olarak en basit yolunun beyni analiz ederek onun işlevselliğini birebir kopyalamak olduğudur. Diğer fikir ise makine öğrenmesi modellerinin mühendislik uygulamalarının yanı sıra, beynin ve zekânın altında yatan prensiplerin anlaşılması için gereken temel bilim problemlerini aydınlatabileceği düşüncesidir.

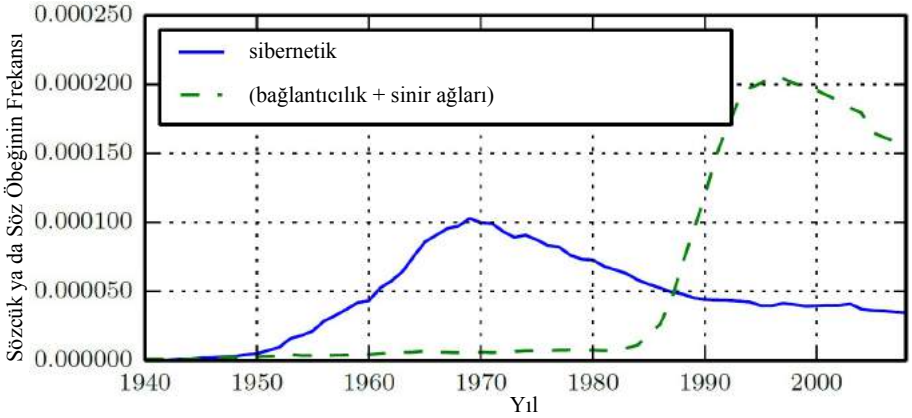
Günümüzdeki "derin öğrenme" sinirbilimin yeni nesil makine öğrenme modellerine olan bakış açısının ötesindedir ve öğrenmeyle ilgili daha genel bir prensibin *çoklu bileşim seviyeleri* peşindedir. Bu prensip, sinirlerden esinlenilmemiş makine öğrenme yöntemlerine de uygulanabilir.

Derin öğrenmenin ataları, sinirbilimsel bakış açısından yola çıkılarak geliştirilmiş basit doğrusal modellerdir. Bu modeller $x_1, \ldots, x_n$ olarak n adet girdi alıp bunları y çıktısıyla ilişkilendirirler. Bunun için $w_1, \ldots, w_n$ olarak verilen ağırlıkları öğrenip çıktıyı $f(\boldsymbol{x}, \boldsymbol{w}) = x_1 w_1 + \cdots + x_n w_n$ formülünü kullanarak hesaplarlar. Şekil 1.7'de gösterilen bu ilk dalga sinir ağları sibernetik olarak bilinirler.

McCulloch-Pitts sinir hücresi (McCulloch ve Pitts, 1943) beyin fonksiyonunun ilkel bir modelidir. Bu doğrusal model $f(\boldsymbol{x}, \boldsymbol{w})$'nin pozitif ya da negatif olduğuna bakarak iki farklı sınıfı birbirinden ayırabilir. Elbette modelin doğru şekilde sınıflandırma yapması için ağırlıklarının ayarlanmış olması gerekmektedir. Bu ağırlıklar bir insan tarafından ayarlanabilir. 1950 yılında, perseptron (Rosenblatt, 1958, 1962)



Şekil 1.6: Kitabın özetlenmiş bir gösterimi. İki bölüm arasındaki ok, ilk bölümün ikincisi için ön koşul olduğunu belirtmektedir.



Şekil 1.7: Yapay zekâ araştırmalarındaki üç dalgadan ikisinin Google Books'taki "sibernetik" ve "bağlantıcılık" terimlerinin frekansları üzerinden gösterimi. (Üçüncü dalga burada geçmek için çok yenidir). İlk dalga olan sibernetik 1940 ve 1960 yılları arasında, biyolojik öğrenme teorilerinin(McCulloch ve Pitts, 1943; Hebb, 1949) gelişmesiyle ve tek nöronların eğitilmesini mümkün kılan perseptron gibi (Rosenblatt, 1958) modellerin ilk defa geliştirilmesiyle birlikte başladı. Tek ya da iki katmanlı sinir ağlarının geri yayılımla (Rumelhart vd., 1986a) eğitildiği ikinci dalga 1980 ve 1995 yılları arasındaki bağlantıcılık akımıyla birlikte başladı. Şimdi içinde bulunduğumuz üçüncü dalga olan derin öğrenme ise 2006 yılında (Hinton vd., 2006; Bengio vd., 2007; Ranzato vd., 2007a) başladı ve 2016'da daha yeni olarak kitaplarda yer almaya başladı. İlk iki dalganın kitap hâlinde yer alması da dalgaları başlatan bilimsel gelişmelerden çok sonra gerçekleşmiştir.

her sınıftan verilen örnekleri kullanarak ağırlıklarını öğrenebilen ilk model oldu. Yaklaşık aynı tarihlerde ortaya çıkan **adaptif doğrusal eleman** (ADALINE) ise basitçe $f(\boldsymbol{x})$'nin değerini kullanarak bir reel sayıyı tahmin ediyordu. ADALINE tahminlerini yaparken veri kullanmayı öğrenebiliyordu (Widrow ve Hoff, 1960).

Bu basit öğrenme algoritmaları, modern makine öğrenmesinin gelişimini çok etkilemiştir. ADALINE'ın ağırlıklarını öğrenmek için kullanılan algoritma **stokastik gradyan inişi** olarak bilinen bir algoritmanın özelleştirilmiş bir hâlidir. Stokastik gradyan inişinin farklı versiyonları, bugün derin öğrenme modelleri içinde hâlâ en yaygın algoritma olarak kullanılmaktadır.

Perceptron ve ADALINE tarafından kullanılan $f(\boldsymbol{x}, \boldsymbol{w})$ üzerine temellendirilmiş modellere **doğrusal modeller** adı verilir. Bu modeller hâlâ en yaygın olarak kullanılan makine öğrenmesi modellerini oluştururlar. Ancak birçok durumda, orijinal modellere göre farklı şekilde *eğitilirler*.

Doğrusal modellerin birçok kısıtlaması bulunmaktadır. Bunlardan en çok bilineni, $f([0,1], \boldsymbol{w}) = 1$, $f([1,0], \boldsymbol{w}) = 1$, $f([1,1], \boldsymbol{w}) = 0$ ve $f([0,0], \boldsymbol{w}) = 0$ şekilde

tanımlanan ÖZEL VEYA fonksiyonudur. Bu kısıtlamayı gözlemleyen eleştirmenler, genel olarak biyolojik temelli öğrenmeye karşı kötü bir algı oluşturmuştur. Bu sinir ağlarının popülerliğindeki ilk büyük çöküş olmuştur.

Günümüzde sinirbilim, derin öğrenme araştırmacıları için hâlâ önemli bir ilham kaynağıdır. Ancak derin öğrenme için artık eskisi kadar baskın bir rehberlik teşkil etmemektedir.

Günümüzde sinirbilimin derin öğrenme üzerindeki rolünün azalmasının başlıca sebebi beyinde neler olduğuyla ilgili yeterince bilgiye sahip olmamamızdır. Beyindeki algoritmaları gerçekten anlayabilmemiz için beyindeki (en azından) binlerce sinir hücresinin aktivitesini takip edebiliyor olmamız gerekmektedir. Bunu yapamadığımız için beynin en basit ve en çok araştırılmış bölgelerinin bile nasıl çalıştığını anlamaktan oldukça uzaktayız (Olshausen ve Field, 2005).

Sinirbilim, tek bir derin öğrenme algoritmasının birçok farklı problemi çözebileceğine inanmamız için bir neden sağlamıştır. Sinirbilimciler, dağ gelinciklerinin beyinlerinin görsel sinyallerini duyma bölgesine yönlendirdiklerinde, beyinlerinin duyma bölgesiyle "görebildiklerini" bulmuşlardır (Von Melchner vd., 2000). Bu bulgu, memeli beyinlerinin büyük bir bölümünün birçok problemi çözmek için tek bir algoritma kullandığı yönünde bir kanı oluşturmuştur. Bu hipotezden önce makine öğrenmesindeki araştırma doğal dil işleme, görü, hareket planlama ve konuşma tanıma gibi araştırma toplulukları arasında oldukça bölünmüş bir hâldeydi. Günümüzde bu araştırma toplulukları hâlâ ayrıdır ancak derin öğrenme çalışan araştırma grupları bu alanlardan birçoğu hatta hepsi üzerine aynı anda araştırma yürütebilmektedirler.

Sinirbilimden esinlenerek kendimize kabaca bir yol çizebilmekteyiz. Birçok hesaplama ünitesinin yalnızca birbirleriyle iletişim kurarak zekâ oluşturması fikrine beyinden yola çıkılarak erişilmiştir. Neocognitron (Fukushima, 1980), görüntü işlemede memeli beynindeki görsel sistemden esinlenilmiş çok güçlü bir model mimarisi ortaya koymuştur. Bu sistem daha sonra Bölüm 9.10'da da göreceğimiz gibi çağdaş evrişimsel ağlar (LeCun vd., 1998b) için bir temel oluşturmuştur. Günümüzdeki çoğu sinir ağı **düzeltilmiş doğrusal birim** (rectified linear unit) adı verilen bir sinir hücresi modeline dayanır. İlk cognitron(Fukushima, 1975) modeli beyin fonksiyonu hakkındaki bilgimizden esinlenen daha karmaşık bir sinir hücresi kullanmaktaydı. Bu basitleştirililmiş model birçok farklı noktadan yola çıkılarak geliştirildi. Nair ve Hinton (2010) sinirbilimsel bir etkilenmeden, Jarrett vd. (2009) ise daha mühendislik odaklı bir etkilenmeden bahseder. Sinirbilim her ne kadar önemli bir esin kaynağı olsa da, kalıplaşmış bir rehber olarak kullanılmamalıdır. Gerçek sinir hücrelerinin aslında doğrusal doğrultulmuş ünitelerden çok farklı fonksiyonlar hesapladığını biliyoruz. Ancak daha gerçekçi sinir modelleri kullanmak

makine öğrenme performansında henüz bir ilerlemeye yol açmamıştır. Ayrıca, sinirbilim birçok sinir ağı *mimarisine* esin kaynağı olsa da, bu modelleri daha iyi *eğitebilecek yöntemler* geliştirmek için henüz biyolojik öğrenmeyle ilgili yeterince bilgiye sahip değiliz.

Medya genelde derin öğrenme ve beynin benzerliğine vurgu yapar. Derin öğrenme araştırmacılarının Bayes istatistiği gibi diğer makine öğrenme alanlarında çalışanlara göre beyne daha fazla atıf yaptığı doğrudur. Ancak bu derin öğrenmenin beyni simüle etme çabası olarak görülmemelidir. Çağdaş derin öğrenme özellikle lineer cebir, olasılık, bilgi kuramı gibi birçok alandan esinlenmektedir. Bazı araştırmacılar sinirbilimi çalışmaları için önemli bir esin kayağı olarak gösterirken, bazı araştırmacılar herhangi bir şekilde sinirbilimle ilgilenmezler.

Bilinmelidir ki beynin algoritmasal seviyede nasıl çalıştığına dair uğraşlar hâlâ devam etmektedir. "Hesaplamalı Sinirbilim" olarak bilinen bu çaba, derin öğrenmeden farklı bir bilim alanıdır. Araştırmacıların bu iki alan arasında gidip gelmeleri oldukça sık rastlanan bir durumdur. Derin öğrenme, zekâ gerektiren problemleri çözmek için bilgisayar sistemlerinin nasıl oluşturulabileceğiyle ilgilenirken, hesaplamalı sinirbilim beynin nasıl çalıştığına dair gerçeğe daha yakın modeller geliştirmeyle uğraşır.

1980'li yıllarda **bağlantıcılık** ya da **paralel dağıtık işleme** (Rumelhart vd., 1986c; McClelland vd., 1995) adı altında sinir ağı araştırmasının ikinci dalgası ortaya çıkmıştır. Bağlantıcılık bilişsel bilimin bağlamı içinde ortaya çıkmış, zihni anlamak amacıyla birçok farklı açıklamayı birleştiren disiplinler arası bir alandır. 1980'li yılların başlarında çoğu bilişsel bilimci sembolik akıl yürütme modellerini incelemekteydi. Popüler olmalarına rağmen, sembolik modellerin beyin tarafından sinir hücereleri kullanılarak nasıl ortaya çıktığını açıklamak zordu. Bağlantıcılar, psikolog Donald Hebb'in 1940'larda ortaya attığı fikirleri (Hebb, 1949) kullanarak, sinirsel yapılara (Touretzky ve Minton, 1985) dayanan bilişsel modelleri incelemeye başladılar.

Birçok basit hesaplama ünitesinin bir ağ oluşturduğunda zeki davranış göstermesi bağlantıcılığın ana fikridir. Bu anlayış hesaplama modellerinde uygulanabildiği kadar biyolojik sinir sistemlerine de uygulanabilir.

1980'li yıllarda bağlantıcılıkla birlikte ortaya çıkan birçok kavram günümüzdeki derin öğrenme yöntemlerinin merkezinde yatmaktadır.

Bu kavramlardan biri dağıtık gösterimlerdir (Hinton vd., 1986). Dağıtık gösterim, her girdinin birçok öznitelik tarafından sunumunu ve aynı zamanda her özniteliğin oldukça fazla girdinin gösteriminde kullanılması olarak özetlenebilir. Örneğin arabaları, kamyonları ve kuşları ayırt edebilen bir görü sistemimiz oldu-

ğunu varsayalım. Bu nesneler aynı zamanda, kırmızı, yeşil ya da mavi olabilir. Nesneleri göstermenin bir yolu, kırmızı araba, kırmızı kamyon, kırmızı kuş gibi olası dokuz kombinasyonun her biri için ayrı ateşlenen sinir hücreleri kullanmaktır. Bu yaklaşım, dokuz farklı sinir hücresi gerektirir ve ayrıca her bir sinir hücresi renkleri ve nesneleri ayrı ayrı öğrenmek durumundadır. Bu durumu iyileştirmenin bir yolu dağıtık gösterim kullanımıdır. Her bir renk ve nesne tipi için üçer sinir hücresi gerekir ve bu durumda toplamda kullanılan hücre sayısı altıya düşer. Kırmızılığı gösteren sinir hücresi bu durumda kırmızılığa dair bilgiyi yalnızca ona özgü kategoriden ziyade araba, kamyon ve kuş görüntülerinin hepsinden öğrenebilir hâle gelir. Dağıtık gösterim kavramı bu kitabın merkezindedir ve Bölüm 15'te daha detaylı olarak işlenecektir.

Bağlantıcılık akımının önemli başarımlarından bir diğeri de içsel gösterimlere sahip olan derin sinir ağlarının geri yayılım kullanılarak eğitilmesi ve geri yayılım algoritmasının popülerleştirilmesidir (Rumelhart vd., 1986a; LeCun, 1987). Bu algoritmanın popülerliği tarih boyunca artıp azalmıştır. Ancak bu kitabın yazıldığı dönemde derin modelleri eğitmede kullanılan başlıca yöntemdir. 1990'lı yıllarda araştırmacılar dizilim modellemede önemli ilerlemeler kaydettiler. Hochreiter (1991) ve Bengio vd. (1994), Bölüm 10.7'de bahsi geçen dizilim modellemedeki bazı temel matematiksel zorlukları keşfettiler. Hochreiter ve Schmidhuber (1997), bu sorunların bazılarını çözmek amacıyla uzun ömürlü kısa-dönem belleği (Long short-term memory | LSTM) geliştirdiler. Günümüzde LSTM, Google'ın doğal dil işleme problemi gibi birçok dizilim modelleme probleminde yaygın olarak kullanılmaktadır.

Sinir ağı araştırmalarının ikinci dalgası 1990'ların ortalarına kadar sürmüştür. Sinir ağları ve diğer yapay zekâ teknolojilerine dayanan birçok girişim, yatırım arama sürecinde gerçeklikten giderek uzaklaşan iddialarda bulunmaya başladılar. YZ araştırmaları, makul olmayan beklentileri karşılayamadıkları zaman yatırımcılar büyük hayal kırıklığına uğradı. Aynı dönemde makine öğrenmesinin diğer alanları gelişmeye devam ediyordu. Kernel makineleri (Boser vd., 1992; Cortes ve Vapnik, 1995; Schölkopf vd., 1999) ve çizgesel modeller (Jordan, 1998) birçok önemli problemde iyi sonuçlar elde ettiler. Bu iki sebepten dolayı sinir ağlarının popüleritesi 2007 yılına kadar sürecek olan bir düşüşe girdi.

Bu dönem boyunca sinir ağları bazı problemlerde etkileyici sonuçlar elde etmeye devam etti (LeCun vd., 1998b; Bengio vd., 2001). Kanada İleri Araştırmalar Enstitüsü (CIFAR), Sinirsel Hesaplama ve Adaptif Algı(NCAP) programıyla sinir ağı araştırmalarını canlı tutmada yardımcı oldu. Bu program, birçok makine öğrenmesi araştırma grubunu Toronto Üniversitesi'nde Geoffrey Hinton, Montreal Üniversitesi'nde Yoshua Bengio ve New York Üniversitesi'nde Yann LeCun önderliğinde birleştirdi. CIFAR NCAP girişimi sinirbilim uzmanları, insan ve bilgisayar görüsü

üzerine çalışan uzmanları da içermekteydi.

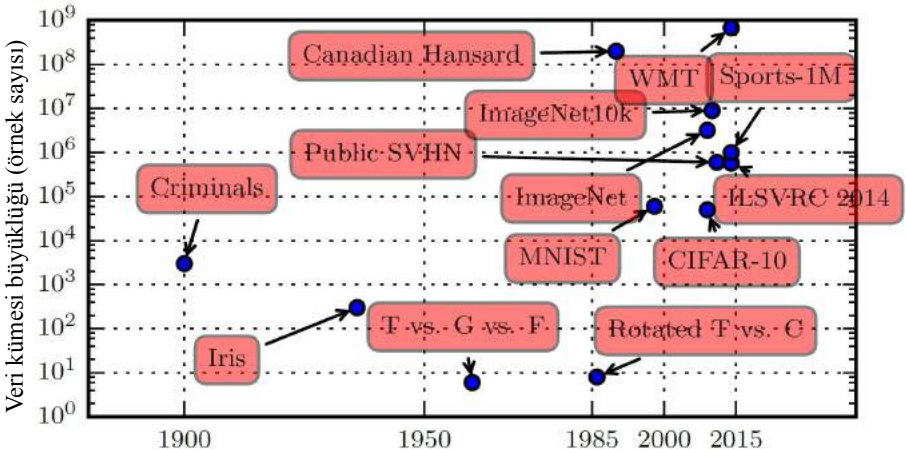
O zamanlar, derin ağların eğitilmesi genellikle çok zor olarak görülüyordu. 1980'li yıllardan beri var olan algoritmaların iyi çalıştıklarını günümüzde biliyoruz ancak bu 2006 yılında o kadar da açık değildi. Zira o dönemdeki bilgisayar donanımları bu algoritmaları yeterli hızda yürütebilecek kadar güçlü değildi. Sinir ağı araştırmalarının üçüncü dalgası 2006 yılında çığır açan bir araştırmayla başladı. Geoffrey Hinton derin inanç ağları olarak bilinen bir tür sinir ağının, katmanların denetimsiz hırslı önyetiştirme metodunu kullanarak eğitilebileceğini gösterdi (Hinton vd., 2006). Bu tür ağlardan Bölüm 15.1'de daha detaylı bahsedeceğiz. Diğer CIFAR bağlantılı araştırma grupları aynı metodu kullanarak birçok farklı derin ağın eğitilebileceğini gösterdiler (Bengio vd., 2007; Ranzato vd., 2007a). Aynı zamanda test örneklerindeki genelleştirme performansının iyileşmesine sistematik olarak yardım ettiler. Sinir ağlarının bu dalgası, araştırmacıların artık eskisinden daha derin sinir ağlarını eğitebileceğini vurgulamak ve derinliğin kuramsal önemini belirtmek amacıyla "derin öğrenme" terimini yaygınlaştırdı (Bengio ve LeCun, 2007; Delalleau ve Bengio, 2011; Pascanu *vd.*, 2014a; Montufar vd., 2014). Derin sinir ağları diğer makine öğrenmesi teknolojilerine dayalı YZ sistemlerini ve elle tasarlanmış sistemleri geçmeye başladılar. Derin öğrenme araştırmasının odağı bu dönem içinde oldukça değişmesine rağmen, sinir ağlarının üçüncü dalgayla birlikte gelen popülerliği şu ana kadar korunmaktadır. Üçüncü dalga, yeni gözetimsiz öğrenme teknikleri ve küçük veri kümelerini kullanarak iyi genelleme yapabilen ağlar üzerine odaklanarak başladı. Ama günümüzde bu ilgi daha çok gözetimli öğrenme algoritmaları ve büyük etiketli veri kümelerinden destek alan modeller üzerine kaymıştır.

1.2.2 Artan Veri Kümesi Boyutları

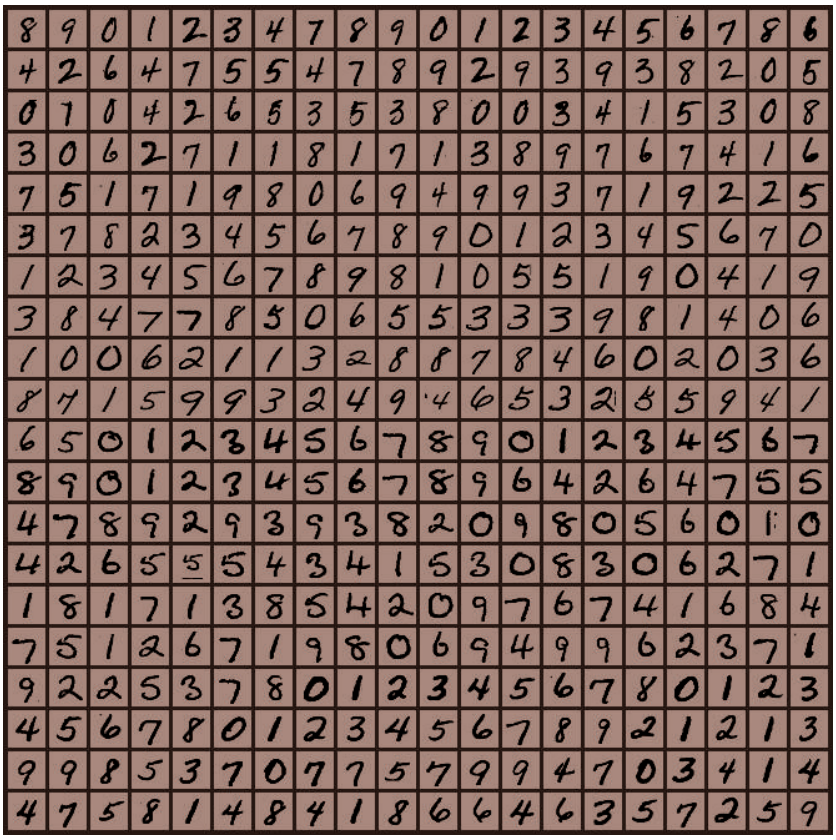
Yapay sinir ağlarıyla ilgili ilk deneylerin 1950'lerde başladığı göz önüne alınırsa, derin öğrenmenin neden çok yakın zamanlara kadar önemli bir teknoloji olarak kabul görmediği merak edilebilir. Derin öğrenme, 1990'lardan beri ticari uygulamalarda başarıyla kullanılmıştır. Ancak yakın zamana kadar bir teknolojiden ziyade sadece uzmanların kullanabildiği bir sanat olarak görülmüştür. Bir derin öğrenme algoritmasından yüksek performans elde etmek için uzmanlık gerektiği doğrudur. Neyse ki, eğitim verisi arttıkça gereken yetenek miktarı azalmaktadır. Bugün karmaşık problemlerde insan performansına erişen öğrenme algoritmaları 1980'lerde kolay problemleri çözmekte zorlanan algoritmalarla neredeyse aynıdır. Tek farklılık, derin mimarilerin eğitilmesini kolaylaştırmak için modellerde yapılan değişikliklerdir. En büyük gelişmeyse, algoritmaların başarılı olmaları için gereken kaynakları sunabilecek hâle gelmemizdir. Şekil 1.8 bize, test veri kümelerinin büyükleklerinin

zamanla ne kadar çok büyüdüğünü göstermektedir. Bu eğilim, toplumun artan dijitalleşmesi sayesinde mümkün olmuştur. Aktivitelerimizin gittikçe daha büyük kısmı bilgisayarlar üstünde gerçekleştikçe gittikçe daha büyük kısmı kaydedilmeye başlanmıştır. Bilgisayarlarımız birbirlerine daha da çok bağlandıkça, bu kayıtları merkezi hâle getirip onları makine öğrenmesi uygulamalarında kullanılacak birer veri kümesine çevirmek kolaylaşmıştır.

"Büyük veri" çağı, makine öğrenmesini oldukça kolaylaştırmıştır. Bunun sebebi,



Şekil 1.8: Zamanla artan veri kümesi. 1900'lerin başlarında, istatistikçiler elle derlenen yüzlerce hatta binlerce ölçümü kullanarak hazırlanmış veri kümeleri üzerinde araştırmalarını devam ettirdiler. (Garson, 1900; Gosset, 1908; Anderson, 1935; Fisher, 1936). 1950'lerden 1980'lere kadar biyolojiden esinlenen makine öğrenmesinin öncüleri, genelde küçük yapay veri kümeleri ile çalıştılar. Buna örnek olarak düşük hesaplama maliyeti ile belirli fonksiyonların öğrenilebileceğini gösteren, karakterlerin düşük çözünürlüklü bit haritaları gösterilebilir. (Widrow ve Hoff, 1960; Rumelhart vd., 1986b). 1980'lerde ve 1990'larda, makine öğrenmesi daha istatistiksel bir hâl aldı. Elle yazılmış sayıların taramalarından oluşan MNIST veri kümesi (Şekil 1.9'da gösterildiği gibi) gibi on binlerce örnek içeren daha büyük veri kümeleri sayesinde güçlenmeye başladı (LeCun vd., 1998b). 2000'lerin başlarında, aynı boyuttaki daha karmaşık CIFAR-10 gibi veri kümeleri üretilmeye başlandı. 2010'a yaklaşıldıkça, milyonlarca örnek içeren daha büyük veri kümeleri, derin öğrenmenin sınırlarını değiştirmeye başladılar. Bu veri kümeleri kamusal Street View House Numbers veri kümesini (Netzer vd., 2011), ImageNet veri kümesinin çeşitli versiyonlarını (Deng vd., 2009, 2010a; Russakovsky vd., 2014a) ve Sports-1M veri kümesini (Karpathy vd., 2014) de içeriyordu. Çizgenin üst kısmında, Kanada Parlemantosu'nun tartışmalarından oluşturulmuş IBM'nin veri kümesi (Brown vd., 1990) ya da WMT İngilizceden Fransızcaya veri kümesi gibi (Schwenk, 2014) çevrilmiş veri kümelerinin diğer veri kümelerinden çok daha büyük olduğunu görebiliriz.



Şekil 1.9: MNIST veri kümesinden örnek girdiler. "NIST"'in açılımı Ulusal Standartlar ve Teknoloji Enstitüsü(National Institue of Standards and Technology)'dür ve veriyi toplayan kurumun adıdır. "M" değiştirilmiş(modified) anlamındadır çünkü veri, makine öğrenme algoritmalarında kullanılması için önişlemeden geçirilmiştir. MNIST veri kümesi, elle yazılmış rakamların taramalarının ve hangi görüntüde hangi rakamın olduğunu belirten etiketlerin olduğu bir veri kümesidir. Bu basit sınıflandırma problemi derin öğrenme araştırmalarında en sık kullanılan testlerden biridir. Günümüz teknikleri tarafından kolayca çözülmesine rağmen hâlâ popülerliğini korumaktadır. Geoffrey Hinton bu veri kümesini, biyologların sıkça meyve sineklerini araştırmalarına benzer şekilde makine öğrenmesi araştırmacılarının da kontrollü bir şekilde deney yapabilmesine olanak sağladığından "makine öğrenmesinin *drozofili*" olarak tanımlamıştır.

istatistiksel kestirimin asıl zorluğu olan küçük bir miktar veriden, yeni verilere genelleme büyük oranda kolaylaşmasıdır. 2016 yılından beri, göz kararı bir kural, gözetimli bir derin öğrenme algoritmasının, kategori başına 5.000 etiketli örnekten sonra kabul edilebilir bir performansa sahip olacağıdır. 10 milyon etiketli örnekle

insan performansına eşit hatta insan performansını aşan modeller eğitilebilir. Başta büyük miktarda etiketsiz veriden nasıl yararlanılabileceği olmak üzere, gözetimsiz ve yarı gözetimli öğrenme gibi daha küçük veri kümeleriyle çalışmak önemli araştırma alanlarıdır.

1.2.3 Artan Model Boyutları

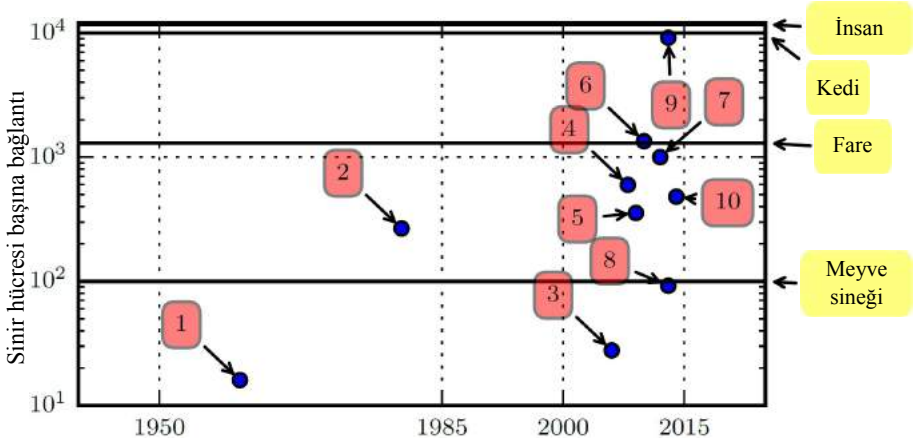
Sinir ağlarının oldukça başarılı olmasının önemli sebeplerinden biri de günümüzde çok daha büyük hesaplama kaynaklarına sahip olmamızdır. Bağlantıcılığın ana fikirlerinden biri de hayvanların birçok sinirinin aynı anda çalıştıklarında zekâya sahip olduğudur. Bir ya da birkaç sinir hücresinden oluşan bir topluluk kendi başlarına yararlı değillerdir.

Biyolojik sinir hücrelerinin birbirlerine olan bağlantıları nispeten seyrektir. Şekil 1.10'da görüldüğü üzere makine öğrenme modellerinin sinir başına bağlantı sayısı, memeli beyinlerinden bile kat kat fazladır.

Sinir ağları, Şekil 1.11'de de gösterildiği üzere yakın zamana kadar toplam sinir hücresi sayısı bakımından şaşırtıcı derecede küçüktü. Zamanla artan sinir ağı boyutları gizli ünitelerin kullanılması ile birlikte, yapay sinir ağları yaklaşık her 2,4 yılda büyüklük olarak ikiye katlandı. Bu büyüme daha büyük hafızaya sahip daha hızlı bilgisayarlar ve daha büyük veri kümeleri sayesinde mümkündür. Daha büyük ağlar, daha karmaşık problemlerde daha isabetli sonuçlar elde etmeyi başardılar. Bu eğilimin onlarca yıl daha süreceği beklenmektedir. Yeni teknolojiler daha hızlı bir şekilde büyümeyi mümkün kılmazsa, yapay sinir ağlarının 2050 yılına kadar insanlarla aynı sayıda sinir hücresine ulaşmaları mümkün olmayacaktır. Biyolojik sinir ağları günümüzdeki yapay sinirlerden çok daha karmaşık fonksiyonları gösterebilmektedir. Dolayısıyla, biyolojik sinir ağları gerçekte, bu çizimin gösterdiğinden çok daha büyük olabilir.

Geçmişe baktığımızda, bir sülükten daha az sinir hücresine sahip sinir ağlarının karmaşık yapay zekâ problemlerini çözememesi o kadar da şaşırtıcı değildir. Günümüzde hesaplama sistemleri bakımından oldukça büyük sinir ağları bile kurbağa gibi kısmen ilkel olan omurgalı hayvanlardan daha az sinir hücresine sahiptir.

Model büyüklüğü, sürekli gelişen MİB, GPU ağ bağlantısı ve dağıtık hesaplama yöntemleri içeren yayılım altyapısı sebebiyle sürekli artmaktadır (Bölüm 12.1.2'de de bahsedildiği gibi). Derin öğrenmenin tarihsel sürecinde çok önemli olan bu gidişin gelecekte de devam etmesi beklenmektedir.



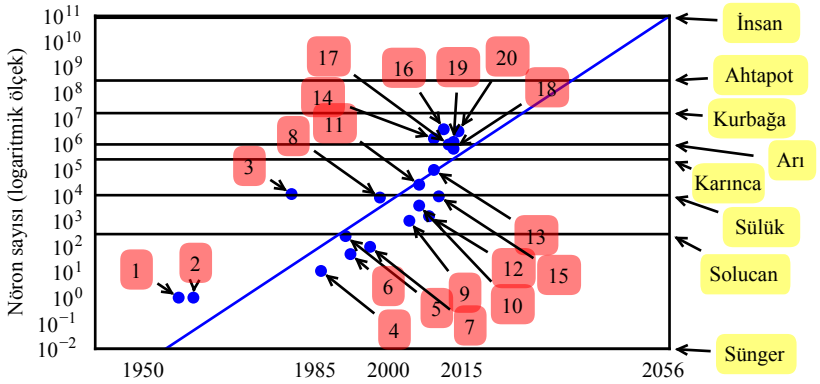
Şekil 1.10: Zaman içinde sinir hücresi başına düşen bağlantı sayısının değişimi. Başlangıçta sinir hücreleri arasındaki bağlantı sayısı donanımla sınırlıydı. Günümüzde sinir hücreleri arasındaki bağlantı sayısı ağı tasarlarken seçtiğimiz bir özelliktir. Bazı yapay sinir ağları, bir kedideki kadar fazla sayıda sinir başına bağlantıya sahiptir. Diğer sinir ağlarının da daha küçük memelilerdeki kadar sinir başına bağlantıya sahip olması sık karşılaşılan bir durumdur. İnsan türünün bile sinir hücresi başına düşen bağlantı sayısı çok da büyük değildir. Biyolojik sinir ağlarının sayısı Wikipedia (2015) isimli kaynaktan alınmıştır.

1. Uyarlanır doğrusal eleman (Widrow ve Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU ile hızlandırılmış evrişimsel ağ (Chellapilla vd., 2006)
4. Derin Boltzmann makinesi (Salakhutdinov ve Hinton, 2009a)
5. Gözetimsiz evrişimsel ağ (Jarrett vd., 2009)
6. GPU ile hızlandırılmış çok katmanlı perseptron (Ciresan vd., 2010)
7. Dağıtık otokodlayıcı (Le vd., 2012)
8. Çoklu GPU evrişimsel ağ (Krizhevsky vd., 2012)
9. COTS HPC gözetimsiz evrişimsel ağ (Coates vd., 2013)
10. GoogLeNet (Szegedy vd., 2014a)

1.2.4 İsabetliliği, Karmaşıklığı ve Gerçek Dünyaya Olan Etkisini Arttırmak

1980'li yıllardan beri, derin öğrenme sürekli olarak tanıma ve öngörmedeki kabiliyetini tutarlı bir şekilde artırdı. Dahası, derin öğrenme daha geniş alanlarda başarıyla uygulandı.

İlk derin öğrenme modelleri, sıkıca kırpılmış ve oldukça küçük görüntülerdeki tekil nesneleri tanıyabiliyordu (Rumelhart vd., 1986a). O zamandan beri sinir



Şekil 1.11: Zamanla artan sinir ağlarının boyutları. Gizli ünitelerin kullanılmasından itibaren yapay sinir ağları yaklaşık olarak 2,4 yılda bir büyüklük olarak ikiye katlanmıştır. Biyolojik sinir ağlarının sayısı Wikipedia (2015) isimli kaynaktan alınmıştır.

1. Perseptron (Rosenblatt, 1958, 1962)
2. Uyarlanır doğrusal eleman (Widrow ve Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Erken geri yayılım ağı (Rumelhart vd., 1986b)
5. Konuşma tanıma için yinelemeli sinir ağı (Robinson ve Fallside, 1991)
6. Konuşma tanıma için çok katmanlı perseptron (Bengio vd., 1991)
7. Ortalama alan sigmoid inanç ağı (Saul vd., 1996)
8. LeNet-5 (LeCun vd., 1998b)
9. Yankı durum ağı (Jaeger ve Haas, 2004)
10. Derin inanç ağı (Hinton vd., 2006)
11. GPU ile hızlandırılmış evrişimsel ağ (Chellapilla vd., 2006)
12. Derin Boltzmann makinesi (Salakhutdinov ve Hinton, 2009a)
13. GPU ile hızlandırılmış derin inanç ağı (Raina vd., 2009)
14. Gözetimsiz evrişimsel ağ (Jarrett vd., 2009)
15. GPU ile hızlandırılmış çok katmanlı perseptron (Ciresan vd., 2010)
16. OMP-1 ağı (Coates ve Ng, 2011)
17. Dağıtık otokodlayıcı (Le vd., 2012)
18. Çoklu GPU evrişimsel ağı (Krizhevsky vd., 2012)
19. COTS HPC gözetimsiz evrişimsel ağ (Coates vd., 2013)
20. GoogLeNet (Szegedy vd., 2014a)

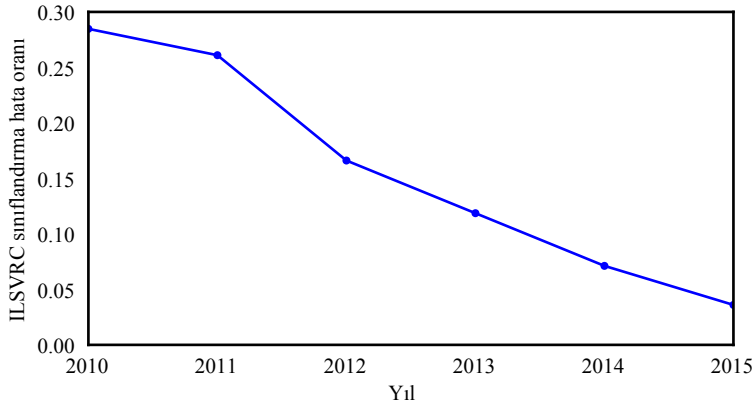
ağlarının işleyebildiği görüntülerin büyüklüğü kademeli bir şekilde arttı. Çağdaş nesne tanıma ağları, yüksek çözünürlüklü fotoğrafları işleyebilirler. Aynı zamanda fotoğrafın tanınacak nesne etrafında kırpılmış olmasını gerektirmezler (Krizhevsky vd., 2012). Benzer şekilde ilk sinir ağları yalnızca iki tür nesneyi ayırabilirken, (hatta bazı durumlarda tek bir nesenin var olup olmadığını) çağdaş sinir ağları genelde en az 1000 farklı sınıfı ayırt edip tanıyabilirler. En büyük nesne tanıma yarışması, her yıl yapılan ImageNet Büyük Ölçekli Görsel Tanıma Yarışması (ImageNet Large

Scale Visual Recognition Challenge, ILSVRC)'dır. Derin öğrenmenin yükselişindeki dramatik anlardan biri, evrişimsel bir ağın bu yarışmayı büyük bir farkla kazanmasıdır. Bu, her modelin ürettiği en olası 5 kategorinin doğru kategoriyi içerip içermediğine bağlı olan en-iyi-5 hatasını yüzde 26.1'den yüzde 15.3'e düşürmüştür (Krizhevsky vd., 2012). Bu andan itibaren bu yarışmalar düzenli bir şekilde derin evrişimsel ağlar tarafından kazanılmıştır. Derin öğrenmedeki ilerlemeler sayesinde, günümüzde en-iyi-5 hatası şekil Şekil 1.12'de gösterildiği üzere yüzde 3.6'ya kadar düşmüştür.

Derin öğrenmenin konuşma tanımada da büyük etkileri olmuştur. 1990'lı yıllar boyunca arttıktan sonra konuşma tanımadaki hata miktarları 2000'lerde duraksamaya başlamıştır. Derin öğrenmenin konuşma tanımada kullanılmasıyla (Dahl vd., 2010; Deng vd., 2010b; Seide vd., 2011; Hinton vd., 2012a) birlikte hata oranlarında büyük düşüşler elde edilmiş, hatta bazı hata oranları yarıya inmiştir. Derin öğrenme ve konuşma tanımanın tarihini Bölüm 12.3'te daha detaylı şekilde işleyeceğiz.

Derin ağlar aynı zamanda yaya saptama ve görüntü bölütleme (Sermanet vd., 2013; Farabet vd., 2013; Couprie vd., 2013) alanlarında da müthiş başarılar elde etmiştir. Trafik işareti tanıma probleminde ise insanüstü performansa ulaşmıştır (Ciresan vd., 2012).

Derin ağların boyutları ve isabetliliği arttıkça, çözebildikleri problemlerin karmaşıklıkları da arttı. Goodfellow vd. (2014d) bize sinir ağlarının tek nesnelerden ziyade bir görüntüdeki bütün harfleri üretebileceğini gösterdi. Daha önce bu tür bir öğrenme için bütün elemanların etiketlenmesi gerektiği düşünülüyordu (Gülçehre



Şekil 1.12: Zamanla azalan hata oranları. Derin ağlar, ILSVRC'de yarışacak ölçeğe ulaştıklarından beri yarışmaları gittikçe azalan hata oranlarıyla düzenli bir şekilde kazanmışlardır. Veriler, Russakovsky vd. (2014b) ve He vd. (2015) isimli kaynaklardan alınmıştır.

ve Bengio, 2013). LSTM gibi yinelemeli sinir ağları günümüzde sabit büyüklükte girdiler yerine *sıralı dizilerle* başka *sıralı dizilerin* arasındaki ilişkileri modellemek için kullanılmaktadır. Diziden-diziye öğrenme, makine çevirisi alanında devrim getirecek gibi gözükmektedir (Sutskever vd., 2014; Bahdanau vd., 2015).

Artan karmaşıklığa olan eğilim derin öğrenmeyi mantıksal bir yönteme doğru itti. Bunun sonucunda bellek hücrelerinden okuyup keyfi bellek hücrelerine yazabilen sinirsel Turing makineleri ortaya çıktı(Graves vd., 2014a). Bu tür sinir ağları, istenen davranışa ait örnekleri kullanarak basit programları öğrenebildiler. Örneğin, karıştırılmış sırada verilen sayı örneklerinden sayıları sıralamayı öğrenebildiler. Kendi kendini programlayan bu teknoloji hâlâ yeni olsa da, teorik olarak gelecekte bütün problemlere uygulanabilecek bir yaklaşımdır.

Derin öğrenmenin diğer parlak başarılarından biriyse **pekiştirmeli öğrenme**deki kullanımlarıdır. Pekiştirmeli öğrenmede özerk bir ajan, bir görevi deneme yanılma yöntemiyle yerine getirmeye çalışmaktadır. DeepMind, derin öğrenme tabanlı bir pekiştirmeli öğrenme sisteminin, Atari oyunlarını oynayabileceğini ve birçok görevde insan seviyesine ulaşabileceğini gösterdi(Mnih vd., 2015). Derin öğrenme aynı zamanda robotikte kullanılan pekiştirmeli öğrenmeyi de kayda değer miktarda iyileştirdi(Finn vd., 2015).

Birçok derin öğrenme uygulaması oldukça kârlıdır. Derin öğrenme günümüzde Google, Microsoft, Facebook, IBM, Baidu, Apple, Adobe, Netflix, NVIDIA ve NEC gibi birçok büyük teknoloji şirketi tarafından kullanılmaktadır.

Derin öğrenmedeki ilerlemeler yazılım altyapısına da güçlü bir şekilde bağlıdır. Theano (Bergstra vd., 2010; Bastien vd., 2012), PyLearn2 (Goodfellow vd., 2013c), Torch (Collobert vd., 2011b), DistBelief (Dean vd., 2012), Caffe (Jia, 2013), MXNet (Chen vd., 2015) ve TensorFlow (Abadi vd., 2015) gibi yazılım kütüphaneleri önemli araştırma projelerinin ve ticari ürünlerin geliştirilmesine destek olmuştur.

Derin öğrenme aynı zamanda diğer bilimlere de katkı sağlamıştır. Nesne tanımak için kullanılan çağdaş evrişimsel ağlar, sinirbilimcilerin üstünde çalışabileceği bir görüntü işleme modeli sunar. Derin öğrenme, aynı zamanda büyük miktarlardaki veriyi işlemede ve bilim alanlarında öngörüler yapmada kullanışlı araçlar sunar. İlaç yapımında moleküllerin birbirleriyle nasıl etkileşeceklerini öngörmede başarıyla kullanılmıştır (Dahl vd., 2014). Atom altı parçacıkları aramak (Baldi vd., 2014) ve mikroskop görüntülerini otomatik olarak işleyerek insan beyninin 3B bir haritasını çıkarmakta da başarılı olmuştur (Knowles-Barley vd., 2014). Derin öğrenmenin gittikçe daha fazla bilim alanında yer alabileceğini öngörüyoruz.

Özetle, derin öğrenme geçtiğimiz on yıllar içinde geliştirilmiş, insan beyni, istatistik ve uygulamalı matematik bilgimizden yola çıkılarak geliştirilmiş bir makine öğrenmesi yaklaşımıdır. Geçtiğimiz yıllarda derin öğrenme, daha güçlü bilgisayarlar

daha büyük veri kümeleri ve derin ağları eğitmek için yeni tekniklerle birlikte hem popülerlik hem de kullanışlılık açısından gelişme göstermiştir. Önümüzdeki yıllar derin öğrenmeyi daha da geliştirmek için aşmamız gereken engeller ve fırsatlarla doludur.

I

Uygulamalı Matematik ve Makine Öğrenmesinin Temelleri

Kitabın bu kısmı, derin öğrenmeyi anlamak için gerekli olan temel matematiksel kavramları tanıtır. Uygulamalı matematikteki çok değişkenli fonksiyonları tanımlamamızı ve bu fonksiyonların en yüksek ve en alçak noktaları bulmamızı sağlayan genel fikirlerinden başlayacağız.

Sonra makine öğrenmesinin temel amaçlarını tanıtacağız. Çeşitleri inançları temsil eden bir model tanımlayarak, bu inançlarımızın gerçek dünyayla uygunluğunu ölçen bir maliyet fonksiyonu tasarlayarak ve bu maliyet fonksiyonunu enküçülterek bu amaçlara nasıl ulaşacağımızı anlatacağız.

Bu çerçeve, derin olmayan makine öğrenmesi dahil olmak üzere birçok makine öğrenmesi algoritmasının temelini oluşturur. Kitabın ileriki bölümlerindeki derin öğrenme algoritmalarıyla ilgili bilgiyi bu çerçeve dahilinde geliştireceğiz.

2

Doğrusal Cebir

Doğrusal cebir bilimde ve mühendislikte geniş bir kullanım alanı bulan bir matematik dalıdır. Doğrusal cebirin ayrık matematikten ziyade sürekli bir matematik dalı olması birçok bilgisayar bilimcisinin doğrusal cebir hakkındaki tecrübesini sınırlamıştır. Doğrusal cebirin iyi bir şekilde anlaşılması, özellikle derin öğrenme olmak üzere birçok makine öğrenmesi algoritmasını anlamak ve kullanmak için gereklidir. Dolasıyla derin öğrenmeye giriş yapmadan önce temel doğrusal cebir kavramlarından bahsetmek istiyoruz.

Eğer doğrusal cebir konusuna hâkim olduğunuzu düşünüyorsanız, bu bölümü geçebilirsiniz. Bu konularla daha önceden tecrübeniz mevcut ama önemli formüllere erişebileceğiniz detaylı bir referans arıyorsanız, *The Matrix Cookbook* (Petersen ve Pedersen, 2006) isimli kitabı öneriyoruz. Doğrusal cebir hakkında daha önceden hiçbir tecrübeniz yoksa bu bölüm kitabın geri kalanı için yeterli olacaktır. Yinede Shilov (1977) gibi sadece doğrusal cebire odaklanan ayrı bir kaynak kullanmanızı öneriyoruz.

Bu bölüm derin öğrenmeyi anlamak için gerekli olmayan bütün doğrusal cebir konularını tamamen atlamaktadır.

2.1 Skalerler, Vektörler, Matrisler ve Tensörler

Doğrusal cebir birçok matematiksel nesne içermektedir:

- **Skalerler**: Doğrusal cebirde karşımıza çıkan birçok sayıdan oluşan dizilerden farklı olarak skalerler, tek bir sayıdan oluşurlar. Kitap boyunca skalerleri italik olarak belirteceğiz ve genellikle küçük harflerden oluşan isimler kullanacağız.

Bir skaleri tanımlarken nasıl bir tür sayı olduğunu belirteceğiz. Örneğin bir gerçel sayıyı tanımlarken "$s \in \mathbb{R}$ bir doğrunun eğimi olsun" gibi bir ifade kullanacağız. Benzer bir şekilde doğal bir sayı tanımlarken "$n \in \mathbb{N}$ yapay ağdaki ünitelerin sayısı olsun" ifadesini kullanacağız.

- **Vektörler**: Bir vektör bir sayı dizisidir. Bu sayıların her biri belli bir düzen içindedir ve bu sayılara bu düzen içindeki sıralamasından yararlanarak erişebiliriz. Genel olarak vektörleri $\boldsymbol{x}$ gibi kalın yazıyla belirteceğiz. Bu vektörün elemanlarını vektörün ismini ve bir altindisi kullanarak tanımlayabiliriz. Örneğin $\boldsymbol{x}$ vektörünün ilk elemanını x_1, ikinci elemanını x_2 vb. şeklinde tanımlayacağız. Bu vektörün içinde ne tür sayıların saklandığını da belirtmemiz gerekmektedir. Eğer vektörün her elemanı $\mathbb{R}$ kümesinin bir elemanı ise ve bu vektörün n adet elemanı varsa, bu vektör $\mathbb{R}$ ve n sayısının kartezyen çarpımından ortaya çıkan ve $\mathbb{R}^n$ şekline gösterilen kümede yer almaktadır. Bir vektörün elemanlarını teker teker ayırt etmek istediğimizdeyse onları köşeli parentez içine alınmış bir sütun olarak gösterebiliriz:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.1}$$

  Vektörleri, her bir elemanının farklı bir eksen üzerindeki koordinatını belirttiği birer nokta olarak düşünebiliriz.

  Bazı durumlarda vektörlerin elemanlarından oluşan bir kümeyi indekslememiz gerekebilir. Bu durumda, bu indislerden oluşan bir küme tanımlayıp, bu kümeyi altindiste belirterek bu elemanlara erişebiliriz. Örneğin bir kümedeki x_1, x_3 ve x_6 elemanlarına erişmek için önce bir $S = \{1, 3, 6\}$ kümesini tanımlayıp daha sonra $\boldsymbol{x}_S$ yazabiliriz. $-$ sembolünü bir kümenin tümleyinini belirtmek için kullanacağız. Örneğin $\boldsymbol{x}_{-1}$ vektörü, $\boldsymbol{x}$ vektörünün x_1 hariç bütün elemanlarını içeren kümeyi temsil eder. $\boldsymbol{x}_{-S}$ vektörü ise yine $\boldsymbol{x}$ vektörünün x_1, x_3 ve x_6 elemanlarının çıkartılmış hâlini gösterir.

- **Matrisler**: Matrisler iki boyutlu sayı dizileridir ve her elemanları bir yerine iki indisle temsil edilir. Kitap boyunca matrisler için genellikle $\boldsymbol{A}$ gibi kalın yazı tipine sahip büyük harfli isimler kullanacağız. Eğer gerçel değerlerden oluşan bir $\boldsymbol{A}$ matrisinin uzunluğu m genişliği n ise bu matris $\mathbb{R}^{m \times n}$ kümesinin içindedir. Genelde matrisin elemanlarını kalın olmayan italik yazı tipiyle belirteceğiz. İndisleri ise virgülle ayıracağız. Örneğin $A_{1,1}$ matrisin en üst ve soldaki elemanını belirtirken $A_{m,n}$ ise en alt ve en sağdaki elemanını

belirmektedir. i numaralı düşey koordinattaki bütün elemanlara yatay koordinatta ":" operatörünü kullanarak erişebiliriz. Örneğin $\boldsymbol{A}_{i,:}$, $\boldsymbol{A}$ matrisinin düşey koordinatının i olduğu yatay bir kesitini ifade eder. Bu $\boldsymbol{A}$ matrisinin i'inci **satırı** olarak bilinir. Benzer bir şekilde $\boldsymbol{A}_{:,i}$, matrisin i'inci sütununu belirtir. Bir matrisi açık olarak göstermek istediğimizde köşeli parantez içine yazılı birer dizi olarak yazabiliriz:

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix}. \tag{2.2}$$

Bazı durumlarda birden fazla harften oluşan matrisli ifadeleri indekslememiz gerekebilir. Bu durumda ifadeden sonra altindis kullanacağız, ancak herhangi bir ifadeyi küçük harfe çevirmeyeceğiz. Örneğin, $f(\boldsymbol{A})_{i,j}$ ifadesi f fonksiyonunu $\boldsymbol{A}$ matrisine uyguladıktan sonra ortaya çıkan matrisin (i, j) indisli elemanını ifade etmektedir.

- **Tensörler**: Bazı durumlarda ikiden fazla ekseni olan dizilere ihtiyaç duyabiliriz. En genel tanımında, düzenli bir ızgara üzerine yerleştirilmiş ve değişken sayıda ekseni bulunan diziler tensör olarak adlandırılır. Biz "A" isimli bir tensörü A yazı tipini kullanarak belirteceğiz. A tensörünün (i, j, k) koordinatlarındaki bir elemanını $\mathsf{A}_{i,j,k}$ ifadesiyle göstereceğiz.

Matrisler için önemli olan operasyonlardan biri **transpoz** operasyonudur. Bir matrisin transpozu, o matrisin **ana köşegen** çizgisi etrafında aynalanmış hâlidir. Ana köşegen matrisin üst sol köşesinden sağ alt köşesine uzanır. Transpoz operasyonunun bir gösterimi için Şekil 2.1'e bakınız. $\boldsymbol{A}$ matrisinin transpozunu $\boldsymbol{A}^\top$ ifadesini kullanarak gösteriyoruz. Bu operasyon

$$(\boldsymbol{A}^\top)_{i,j} = A_{j,i}. \tag{2.3}$$

ifadesiyle tanımlıdır.

Vektörler tek bir sütun içeren matrisler olarak düşünülebilir. Dolayısıyla bir vektörün transpozu tek bir satır içeren bir matristir. Bir vektörü satır içinde bir

$$\boldsymbol{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \Rightarrow \boldsymbol{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix}$$

Şekil 2.1: Bir matrisin transpozu, matrisin ana köşegeni etrafında aynalanmış görüntüsü olarak düşünülebilir.

satır matrisi olarak yazıp daha sonra transpoz operatörüyle standart bir sütun vektörü olarak kullanmaktayız. Örneğin, $\boldsymbol{x} = [x_1, x_2, x_3]^\top$.

Bir skaler tek elemanlı bir matris olarak düşünelebilir. Bu sebeple bir skaler kendisinin transpozudur: $a = a^\top$.

Aynı şekile sahip oldukları sürece, iki matrisi birbiriyle toplayalabiliriz. Bunu karşılıklı elemanlarını toplayarak yaparız: $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$ öyleki $C_{i,j} = A_{i,j} + B_{i,j}$.

Aynı zamanda bir matrise skaler ekleyebiliriz ya da bir matrisi skalerle çarpabiliriz. Bu bahsi geçen işlemi verilen matrisin her elemanına uygulamaya eş değerdir: $\boldsymbol{D} = a \cdot \boldsymbol{B} + c$ öyleki $D_{i,j} = a \cdot B_{i,j} + c$.

Derin öğrenme bağlamında alışılagelmiş gösterimlerin dışında gösterimler kullanmaktayız. Bir matris ve bir vektörün toplamını, sonucu bir matris olacak şekilde tanımlıyoruz: $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{b}$, öyleki $C_{i,j} = A_{i,j} + b_j$. Burada $\boldsymbol{b}$ vektörü matrisin her bir satırına eklenmiştir. Bu kısa gösterim toplama işleminden önce $\boldsymbol{b}$ vektörünün her bir satırına kopyalandığı yardımcı bir matris tanımlama gereksinimini ortadan kaldırır. $\boldsymbol{b}$ vektörü üstü kapalı kopyalama işlemine **yayımlama** denir.

2.2 Matris ve Vektörleri Çarpmak

Matrislerle ilgili en önemli işlemlerden biri iki matrisin çarpım işlemidir. $\boldsymbol{A}$ matrisi ve $\boldsymbol{B}$ matrisinin **matris çarpımı** $\boldsymbol{C}$ matrisidir. Bu çarpımın tanımlı olabilmesi için $\boldsymbol{A}$ matrisinin sütun sayısıyla $\boldsymbol{B}$ matrisinin satır sayısı eşit olmalıdır. Eğer $\boldsymbol{A}$ matrisinin şekli $m \times n$ ve $\boldsymbol{B}$ matrisinin şekli $n \times p$ ise $\boldsymbol{C}$ matrisinin şekli de $m \times p$ kadardır. Matris çarpımını iki matrisi yanyana koyarak yazabiliriz. Örneğin,

$$\boldsymbol{C} = \boldsymbol{AB} \tag{2.4}$$

.

Çarpım işlemi

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}. \tag{2.5}$$

ifadesiyle tanımlanmıştır.

Burada önemli olan nokta, matris çarpımının yalnızca matris elemanlarının birebir çarpımına eşit *olmadığıdır*. Bu işleme **eleman çarpımı** ya da **Hadamard çarpımı** denir ve $\boldsymbol{A} \odot \boldsymbol{B}$ şeklinde gösterilir.

İki eşit boyutlu vektör $\boldsymbol{x}$ ve $\boldsymbol{y}$ arasındaki **iç çarpım**, $\boldsymbol{x}^\top \boldsymbol{y}$ ifadesiyle belirtilen matris çarpımına eşittir. $\boldsymbol{C} = \boldsymbol{AB}$ şeklindeki matris çarpımını, $C_{i,j}$ ifadesini $\boldsymbol{A}$ matrisinin i'inci satırının ve $\boldsymbol{B}$ matrisinin j'inci sütununun iç çarpımı olarak hesaplayarak gösterebiliriz.

Matris çarpımı matrislerin matematiksel analizini kolaylaştıran birçok yararlı özelliğe sahiptir. Örneğin, matris çarpımı dağılma özelliğine sahiptir:

$$\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C}) = \boldsymbol{AB}+\boldsymbol{AC}. \tag{2.6}$$

Aynı zamanda birleşme özelliğine de sahiptir:

$$\boldsymbol{A}(\boldsymbol{BC}) = (\boldsymbol{AB})\boldsymbol{C}. \tag{2.7}$$

Matris çarpımı, skaler çarpımın aksine değişme özelliğine sahip *değildir* (yani, $\boldsymbol{AB} = \boldsymbol{BA}$ eşitliği her zaman geçerli değildir). Bununla birlikte, vektörler arasındaki iç çarpım değişme özelliğine sahiptir:

$$\boldsymbol{x}^\top \boldsymbol{y} = \boldsymbol{y}^\top \boldsymbol{x}. \tag{2.8}$$

Bir matris çarpımının transpozu basit olarak

$$(\boldsymbol{AB})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top \tag{2.9}$$

şeklinde ifade edilebilir. Bu çarpmının sonucu skaler olduğu ve dolayısıyla kendi transpozuna eşit olduğu için Denklem 2.8 eşitliğini ispatlar:

$$\boldsymbol{x}^\top \boldsymbol{y} = \left(\boldsymbol{x}^\top \boldsymbol{y}\right)^\top = \boldsymbol{y}^\top \boldsymbol{x}. \tag{2.10}$$

Bu kitabın asıl odak noktası doğrusal cebir olmadığı için matris çarpımlarının kullanışlı özelliklerini sıraladığımız etraflı bir listesini oluşturmayacağız. Ancak okuyucu birçok böyle özelliğin olduğunu unutmamalıdır.

Artık bir doğrusal denklemler sistemi yazabilecek doğrusal cebir notasyonu bilgisine sahibiz:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{2.11}$$

Bu eşitlikte $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ bilinen bir matris, $\boldsymbol{b} \in \mathbb{R}^m$ bilinen bir vektör ve $\boldsymbol{x} \in \mathbb{R}^n$ değerlerini bilmediğimiz ve çözmemiz gereken bir vektördür. $\boldsymbol{x}$ vektörünün her elemanı x_i, bu bilinmeyen değişkenlerden biridir. $\boldsymbol{A}$ matrisinin her satırı ve $\boldsymbol{b}$ vektörünün her elemanı, sistem için yeni bir kısıtlamayı ifade eder. Denklem 2.11

$$\boldsymbol{A}_{1,:}\boldsymbol{x} = b_1 \tag{2.12}$$

$$\boldsymbol{A}_{2,:}\boldsymbol{x} = b_2 \tag{2.13}$$

$$\dots \tag{2.14}$$

$$\boldsymbol{A}_{m,:}\boldsymbol{x} = b_m \tag{2.15}$$

şeklinde yazılabilir. Hatta bunu daha da detaylı olarak,

$$\boldsymbol{A}_{1,1}x_1 + \boldsymbol{A}_{1,2}x_2 + \cdots + \boldsymbol{A}_{1,n}x_n = b_1 \tag{2.16}$$

$$\boldsymbol{A}_{2,1}x_1 + \boldsymbol{A}_{2,2}x_2 + \cdots + \boldsymbol{A}_{2,n}x_n = b_2 \tag{2.17}$$

$$\cdots \tag{2.18}$$

$$\boldsymbol{A}_{m,1}x_1 + \boldsymbol{A}_{m,2}x_2 + \cdots + \boldsymbol{A}_{m,n}x_n = b_m. \tag{2.19}$$

şeklinde ifade edebiliriz.

Matris vektör çarpımı notasyonu bu tür eşitlikler için daha kısa bir gösterim sağlar.

2.3 Birim ve Ters Matrisler

Doğrusal cebir, Denklem 2.11 eşitliğini $\boldsymbol{A}$ matrisinin birçok değeri için analitik olarak çözebilmemizi sağlayan **matris evirme** olarak bilinen güçlü bir araç içerir.

Matris evirmeyi tanımlayabilmemiz için öncelikle **birim matris** kavramını tanımlamamız gerekmektedir. Birim matrisler, bir vektör ile çarpıldıklarında o vektör üzerinde hiçbir değişiklik yapmayan matrislerdir. n boyutlu vektörler üzerinde işlem yapan birim matrisleri $\boldsymbol{I}_n$ şeklinde belirtiyoruz. Biçimsel olarak bunu, $\boldsymbol{I}_n \in \mathbb{R}^{n\times n}$ ve

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{I}_n\boldsymbol{x} = \boldsymbol{x}. \tag{2.20}$$

şeklinde belirtiyoruz.

Birim matrislerin yapısı oldukça basittir: Ana köşegen üzerindeki bütün elemanların değeri bir, geriye kalan bütün elamanların değeri ise sıfırdır. Bir örnek için Şekil 2.2 içerisine bakılabilir.

$\boldsymbol{A}$ matrisinin tersi, $\boldsymbol{A}^{-1}$ şeklinde gösterilir ve

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_n. \tag{2.21}$$

olacak şekilde tanımlıdır.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Şekil 2.2: Örnek birim matris: $\boldsymbol{I}_3$.